CHAPTER 7

ライセンス インベントリ管理関数

この章では、次のライセンス インベントリ管理関数に関する情報を示します。

asyncAnnotateLicenses

構文

```
String asyncAnnotateLicenses()(UserToken token,String jobGroup,String[] lic_ids, String[]
annotation,IDStatusListener listener) throws RemoteException;
```

説明

この関数を使用すると、ライセンスに注釈を付けることができます。

この関数はノンブロッキングであり、呼び出し元に即座に要求 ID を返します。この関数を呼び出している間、クライアント プログラムは StatusListener インターフェイスを実装するリスナー オブジェクトを提供します。操作が完了すると、リスナー オブジェクトの onStatus() メソッドが呼び出されます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
jobGroup	—	—	この非同期ジョブが属するグループ。null に設定した場合は、返されたジョブ ID がジョブ グループ名として使用されます。
lic_ids	String の配列、必須	—	License オブジェクトの ID。
annotation	String の配列、必須	最大 99 文字のテキスト文字列	各ライセンスの注釈テキスト。Cisco License Manager は annotation パラメータの長さをチェックしません。文字列制限を超える場合は、Cisco IOS ソフトウェアがエラーを返します。
listener	IDStatusListener オブジェクト、必須	—	IDStatusListener インターフェイスを実装するオブジェクト。

戻り値

この関数は呼び出し元に要求 ID 文字列を返します。操作が完了すると、onStatus() メソッドの入力パラメータとしてステータスが提供されます。リスナーは、IDStatusItem のリストを含む IDStatus を受信します。各 IDStatusItem には、ライセンス ID、この操作のエラー コード、およびエラー メッセージが含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

IDStatusListener の onStatus() メソッドの入力パラメータには、エラー コードとエラー メッセージが含まれます。IDStatus オブジェクトには複数のエラー コードとエラー メッセージが含まれる場合があります。次の例では、onStatus() メソッドでエラー コードとエラー メッセージを取得しています。

```
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();
        }
    }
```

asyncDeployLicenses

構文

```
String asyncDeployLicenses(UserToken token, String jobGroup,String[]
licIDs,IDStatusListener listener) throws RemoteException;
```

説明

この関数は、指定したライセンスをそのターゲット デバイスに配備します。

この関数は呼び出し元に即座に要求 ID を返します。この関数を呼び出しているとき、クライアント プログラムは StatusListener インターフェイスを実装するリスナー オブジェクトを提供します。操作が完了すると、リスナー オブジェクトの onStatus() メソッドが呼び出されます。

入力パラメータ

パラメータ	タイプ	値	説明
jobgroup	String、必須	—	この非同期ジョブが属するグループ。null に設定した場合は、返されたジョブ ID がジョブ グループ名として使用されます。
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
lic_ids	String の配列、必須	—	ライセンス ID の配列。
listener	IDStatusListener オブジェクト、必須	—	IDStatusListener インターフェイスを実装するオブジェクト。

戻り値

この関数は呼び出し元に要求 ID 文字列を返します。操作が完了すると、onStatus() メソッドの入力パラメータとしてステータスが提供されます。

リスナーは、IDStatusItem のリストを含む IDStatus を受信します。各 IDStatusItem には、ライセンス ID、この操作のエラー コード、およびエラー メッセージが含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

IDStatusListener の onStatus() メソッドの入力パラメータには、エラー コードとエラー メッセージが含まれます。IDStatus オブジェクトには複数のエラー コードとエラー メッセージが含まれる場合があります。次の例では、onStatus() メソッドでエラー コードとエラー メッセージを取得しています。

```
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();
    }
}
```

asyncObtainLicense

構文

```
public String asyncObtainLicense(UserToken token,String jobGroup,LicenseRequest[]
licReq,boolean deploy,IDStatusListener listener) throws RemoteException
```

説明

この関数は、特定の Product Authorization Key（PAK; 製品認証キー）ID に関連付けられた情報を Cisco Product License Registration Portal からダウンロードし、その情報をインベントリに格納します。最初の関数はライセンスの取得だけを行います。2 番目の関数は、ライセンスを取得してそれらを配備します。

この関数はノンブロッキングであり、呼び出し元に即座に要求 ID を返します。この関数を呼び出している間、クライアント プログラムは StatusListener インターフェイスを実装するリスナー オブジェクトを提供します。操作が完了すると、リスナー オブジェクトの onStatus() メソッドが呼び出されます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
jobGroup	—	—	この非同期ジョブが属するグループ。null に設定した場合は、返されたジョブ ID がジョブ グループ名として使用されます。
lic_reqs	LicenseRequest の配列、必須	—	LicenseRequest の配列。
deploy	Boolean、必須	True、False	取得したすべてのライセンスを配備するようサーバに要求する場合は True。
listener	IDStatusListener オブジェクト、必須	—	IDStatusListener インターフェイスを実装するオブジェクト。

戻り値

この関数は呼び出し元に要求 ID 文字列を返します。操作が完了すると、onStatus() メソッドの入力パラメータとしてステータスが提供されます。

リスナーは、IDStatusItem のリストを含む IDStatus を受信します。各 IDStatusItem には、PAKid + "-::-" + SKU 名 + "-::-" + udi、この操作のエラー コード、およびエラー メッセージが含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

IDStatusListener の onStatus() メソッドの入力パラメータには、エラー コードとエラー メッセージが含まれます。IDStatus オブジェクトには複数のエラー コードとエラー メッセージが含まれる場合があります。次の例では、onStatus() メソッドでエラー コードとエラー メッセージを取得しています。

```
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();
```

```
        }
    }
```

deployLicenseByFile

構文

```
public Status deployLicenseByFile(UserToken token,String devID,String
licFileContent)throws RemoteException;
```

説明

この関数は、指定したライセンスの内容をターゲット デバイスに配備します。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
devID	String、必須	—	デバイス ID 文字列。この API でターゲットとして使用できるのは、Application Control Engine（ACE）デバイスだけです。
licFileContent	String、必須	—	ライセンス ファイルの内容。

戻り値

この関数は、操作が成功したか失敗したかを示す Status を呼び出し元に返します。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

getDeviceIdsWithUnDeployedLicenses

構文

```
IDStatus getDeviceIdsWithUnDeployedLicenses(UserToken token, String folder) throws
RemoteException;
```

説明

この関数は、配備されていないライセンスを持つデバイスの ID を返します。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
folder	String、必須	—	デバイス フォルダの名前（グループ化されていないデバイスを検索する場合は null）。

戻り値

この関数は、IDStatusItem 配列を含む IDStatus オブジェクトを返します。各 IDStatusItem にはデバイス ID が含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、Status オブジェクトは SUCCESS 以外のエラー コードとエラー メッセージを返します。

getPAKIdsWithUnDeployedLicenses

構文

```
IDStatus getPAKIdsWithUndeployedLicenses(UserToken token,String group) throws
RemoteException;
```

説明

この関数は、配備されていないライセンスを持つ PAK の ID を返します。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
group	String、必須	—	PAK グループの名前（グループ化されていないデバイスを検索する場合は null）。
folder	String、必須	—	PAK フォルダの名前。フォルダが null の場合は、デフォルト フォルダに含まれる PAK が検索されます。

戻り値

この関数は、IDStatusItem 配列を含む IDStatus オブジェクトを返します。各 IDStatusItem には PAK ID が含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、Status オブジェクトは SUCCESS 以外のエラー コードとエラー メッセージを返します。

getLicensesOnDevice

構文

```
LicenseStatus getLicensesOnDevice(UserToken token, String dev_id) throws RemoteException;
```

説明

この関数は、指定したデバイスに存在するライセンス情報を取得します。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
dev_id	String、必須	—	デバイス ID 文字列。

戻り値

この関数は LicenseStatus オブジェクトを返します。次の例では、ステータスに含まれるエラー コード、エラー メッセージ、および戻りオブジェクトを取得しています。

```
        LicenseStatus status = getLicensesOnDevice(…, …);

        // 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.
        LicenseStatusItem[] items = status.getLicenseStatusItems()

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

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

    // 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();
        }
```

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

入力配列の要素でエラーが発生した場合は、エラー コードとエラー メッセージを含む Status オブジェクトが返されます。

getRehostableSKUsByDevice

構文

```
RehostableSKUStatus getRehostableSKUsByDevice(UserToken token, String dev_id) throws
RemoteException;
```

説明

この関数は、指定した入力デバイスでライセンスの再ホストに使用できる SKU の配列を取得します。また、そのデバイスに配備されているライセンスを含むすべての SKU を問い合せるよう SWIFT に要求します。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
dev_id	String、必須	—	指定したデバイスの ID。

戻り値

この関数は RehostableSKUStatus オブジェクトを返します。指定したデバイスのライセンスが取得されなかった場合は、返された RehostSKUInfoStatus の RehostableSKU フィールドが null に設定されます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合は、エラーの情報を含む Status オブジェクトが返されます。

getRehostInfo

構文

```
RehostInfoStatus getRehostInfo(UserToken token, String[] dev_ids) throws RemoteException;
```

説明

この関数は、指定した各デバイスの RehostInfo を返します。各 RehostInfo には、再ホスト要求と許可チケットまたは再ホスト チケットが含まれます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
dev_ids	String の配列、必須	—	デバイス ID の配列。

戻り値

この関数は RehostInfoStatus オブジェクトを返します。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

操作エラーが発生した場合、RehostInfoStatus オブジェクトには SUCCESS 以外のエラー コードとエラー メッセージが含まれます。それ以外の場合は、RehostInfoStatusItem 配列を反復処理してすべての RehostInfo オブジェクトを取得する必要があります。

initRehostLicense

構文

```
Status initRehostLicense(UserToken token, RehostRequest rehost_req) throws
RemoteException;
```

説明

Cisco Product License Registration Portal からの再ホストには制限があり、デバイスごとに取得される PermissionTicket は、新しいライセンスが取得されるまで 1 つだけに限定されます。これは、PermissionTicket と RehostTicket はそれぞれ、デバイスごとに常に 1 つだけ存在することを意味します。

この関数は、再ホスト プロセスの最初のステップです。再ホスト プロセスは、Cisco Product License Registration Portal からの許可チケットの取得、デバイスからの再ホスト チケットの取得、Cisco Product License Registration Portal への再ホスト チケットの送信によるライセンスの取得、宛先デバイスへのライセンスの配備など、いくつかのステップから成ります。

取得した PermissionTicket はローカル ストレージに格納され、後で転送元デバイスからライセンスを取り消すときに使用されます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
rehost_req	RehostRequest、必須	—	要求を表すオブジェクト。

戻り値

この関数は Status オブジェクトを返します。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

操作が失敗した場合、Status には SUCCESS 以外のエラー コードとエラー メッセージが含まれます。

listAllLicenses

構文

```
public License PagingInfo listAllLicenses(UserToken token, Pagination pageinfo) throws
RemoteException;
```

説明

この関数は、インベントリからすべてのライセンスを取得します。

有効なオフセットと最大数を指定してページ分割オプションを設定できます。次の例は、オフセットと最大数を指定してページ分割を設定する方法を示します。

1 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(0, 10)
```

5 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(50, 10)
```

ページ分割せずにすべてのレコードを取得するには、次のようにします。

```
Pagination p = new Pagination(0, -1)
```

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
pageinfo	DataObject、必須	—	offset と max を含むページ分割オブジェクト。offset は、この関数によって取得する、最初のレコード セットからの相対的なオフセットを指定します。max は、取得するレコードの最大数を指定します。max を -1 に設定すると、値が -1 であるすべてのレコードが返されます。

戻り値

この関数は、ライセンスの配列を含む LicensePagingInfo オブジェクトを返します。offset がレコード数よりも大きい場合は、サイズが 0 の配列が返されます。操作エラーが発生した場合、Status オブジェクトにはエラー コードとエラー メッセージが含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、この関数は null を返します。

listAllLicenseLines

構文

```
public LicenseLinePagingInfo listAllLicenseLines(UserToken token,Pagination pageinfo)throws
RemoteException;
```

説明

この関数は、インベントリからすべてのライセンスを取得します。

有効なオフセットと最大数を指定してページ分割オプションを設定できます。次の例は、オフセットと最大数を指定してページ分割を設定する方法を示します。

1 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(0, 10)
```

5 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(50, 10)
```

ページ分割せずにすべてのレコードを取得するには、次のようにします。

```
Pagination p = new Pagination(0, -1)
```

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
pageinfo	DataObject、必須	—	offset と max を含むページ分割オブジェクト。offset は、この関数によって取得する、最初のレコード セットからの相対的なオフセットを指定します。max は、取得するレコードの最大数を指定します。max を -1 に設定すると、値が -1 であるすべてのレコードが返されます。

戻り値

この関数は、ライセンスの配列を含む LicenseLinePagingInfo オブジェクトを返します。offset がレコード数よりも大きい場合は、サイズが 0 の配列が返されます。操作エラーが発生した場合、Status オブジェクトにはエラー コードとエラー メッセージが含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、この関数は null を返します。

listAllPAKs

構文

```
public PAKPagingInfo listAllPAKs(UserToken token, Pagination pageinfo) throws
RemoteException;
```

説明

この関数は、インベントリからすべての PAK を取得します。

有効なオフセットと最大数を指定してページ分割オプションを設定できます。次の例は、オフセットと最大数を指定してページ分割を設定する方法を示します。

1 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(0, 10)
```

5 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(50, 10)
```

ページ分割せずにすべてのレコードを取得するには、次のようにします。

```
Pagination p = new Pagination(0, -1)
```

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
pageinfo	DataObject、必須	—	offset と max を含むページ分割オブジェクト。offset は、この関数によって取得する、最初のレコード セットからの相対的なオフセットを指定します。max は、取得するレコードの最大数を指定します。max を -1 に設定すると、値が -1 であるすべてのレコードが返されます。

戻り値

この関数は、PAK の配列を含む PAKPagingInfo オブジェクトを返します。offset がレコード数よりも大きい場合は、サイズが 0 の配列が返されます。操作エラーが発生した場合、Status オブジェクトにはエラー コードとエラー メッセージが含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、この関数は null を返します。

listAllLicensesInPAK

構文

```
public IDPagingInfo listAllLicensesInPAK(UserToken token,String pakID, Pagination
pageinfo) throws RemoteException;
```

説明

この関数は、指定した PAK に属するライセンス ID の配列を返します。

有効なオフセットと最大数を指定してページ分割オプションを設定できます。次の例は、オフセットと最大数を指定してページ分割を設定する方法を示します。

1 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(0, 10)
```

5 ページ目を取得するには、次のようにします。

```
Pagination p = new Pagination(50, 10)
```

ページ分割せずにすべてのレコードを取得するには、次のようにします。

```
Pagination p = new Pagination(0, -1)
```

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
pageinfo	DataObject、必須	—	offset と max を含むページ分割オブジェクト。offset は、この関数によって取得する結果セットの、最初のレコードからの相対的なオフセットを指定します。max は、取得するレコードの最大数を指定します。max を -1 に設定すると、すべてのレコードが返されます。

戻り値

この関数は、ライセンス ID の配列を含む IDPagingInfo オブジェクトを返します。offset がレコード数よりも大きい場合は、サイズが 0 の配列が返されます。操作エラーが発生した場合、Status オブジェクトにはエラー コードとエラー メッセージが含まれます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、この関数は null を返します。

obtainLicenseForRehost

構文

```
public IDStatus obtainLicenseForRehost(UserToken token, RehostRequest rehostReq) throws
RemoteException;
```

説明

Cisco Product License Registration Portal からの再ホストには制限があり、デバイスごとに取得される PermissionTicket は、新しいライセンスが取得されるまで 1 つだけに限定されます。これは、PermissionTicket と RehostTicket はそれぞれ、デバイスごとに常に 1 つだけ存在することを意味します。

この関数は、再ホスト プロセスの 3 番目のステップです。再ホスト プロセスは、Cisco Product License Registration Portal からの許可チケットの取得、デバイスからの再ホスト チケットの取得、Cisco Product License Registration Portal への再ホスト チケットの送信によるライセンスの取得、宛先デバイスへのライセンスの配備など、いくつかのステップから成ります。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
rehost_req	RehostRequest、必須	—	要求を表すオブジェクト。

戻り値

この関数は、ID Status オブジェクトと、取得されたライセンス オブジェクトを返します。

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

           // 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.
           LicenseStatusItem[] items = status.getLicenseStatusItems()

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

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

       // 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();
           }
```

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

操作エラーが発生した場合、Status オブジェクトには SUCCESS 以外のエラー コードとエラー メッセージが含まれます。それ以外の場合は、IDStatusItem 配列を反復処理してすべてのライセンス オブジェクトを取得する必要があります。

readLicenses

構文

```
LicenseStatus readLicenses(UserToken token, String[] lic_ids) throws RemoteException;
```

説明

この関数は、指定したデバイス ID を使用して、インベントリからライセンス オブジェクトの配列を取得します。lic_ids パラメータが null の場合は、インベントリからすべてのライセンス オブジェクトが取得されます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
lic_ids	String の配列、必須	—	ライセンス ID の配列。

戻り値

この関数は LicenseStatus オブジェクトを返します。次の例では、ステータスに含まれるエラー コード、エラー メッセージ、および戻りオブジェクトを取得しています。

```
LicenseStatus status = readLicenses(…, …);

// 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.
LicenseStatusItem[] items = status.getLicenseStatusItems()

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

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

    // 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();
}
```

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

入力配列の要素でエラーが発生した場合は、エラーに関する情報を含む Status オブジェクトが返されます。

rehostLicense

構文

```
Status rehostLicense(UserToken token, RehostRequest rehost_req) throws RemoteException;
```

説明

この関数は、あるデバイスから別のデバイスにライセンスを再ホストする要求を送信します。再ホスト プロセスは、Cisco Product License Registration Portal からの許可チケットの取得、デバイスからの再ホスト チケットの取得、Cisco Product License Registration Portal への再ホスト チケットの送信によるライセンスの取得、宛先デバイスへの新しく取得したライセンスの配備など、いくつかのステップから成ります。これらのステップは、この関数によって 1 回の操作としてカプセル化されます。取得したライセンスはローカル ストレージに格納され、後で宛先デバイスにライセンスを配備するときに使用できます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
rehost_req	RehostRequest、必須	—	要求を表すオブジェクト。

戻り値

この関数は、エラー コードとエラー メッセージを含む Status オブジェクトを返します。操作が成功した場合は、ClmErrors.SUCCESS エラー コードが返されます。操作が失敗した場合は、ClmErrors.SUCCESS 以外のエラー コードとエラー メッセージが返されます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、この関数は、エラーを示す Status オブジェクトを返します。

reObtainLicense

構文

```
IDStatus reObtainLicense(UserToken token, String dev_id) throws RemoteException;
```

説明

この関数は、ライセンスを再送するよう Cisco Product License Registration Portal に要求します。ライセンスを受信した後、Cisco License Manager データ ストレージを更新して同期化します。デバイスへのライセンスの配備は行いません。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
dev_id	String	—	ライセンスの再送先デバイスの ID。

戻り値

この関数は IDStatus オブジェクトを返します。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合は、SUCCESS 以外のエラー コードとエラー メッセージを含む IDStatus オブジェクトが返されます。それ以外の場合は、SUCCESS エラー番号を含む IDStatus オブジェクトが返され、IDStatusItem 配列を反復処理して再取得されたすべてのライセンス ライン ID を取得する必要があります。

resendLicense

構文

```
Status resendLicense(UserToken token, String dev_id) throws RemoteException;
```

説明

この関数は、破損したライセンス ファイルを復元するために、ライセンスをデバイスに再送します。この関数は Cisco Product License Registration Portal から取得済みのすべてのライセンスを要求し、それらを License Manager データベースに保存してからデバイスに再配備します。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
dev_id	String、必須	—	ライセンスの再送先デバイスの ID。

戻り値

この関数は、エラー コードとエラー メッセージを含む Status オブジェクトを返します。操作が成功した場合は、ClmError.SUCCESS エラー コードが返されます。

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合、この関数は、エラーを示す Status オブジェクトを返します。

revokeLicenseForRehost

構文

```
Status revokeLicenseForRehost (UserToken token, RehostRequest rehost_req) throws
RemoteException;
```

説明

この関数は、作業の途中で再ホストが失敗したときに使用します。PermissionTicket の取得と Cisco License Manager インベントリへの格納が正常に完了している場合にだけ呼び出します。Cisco Product License Registration Portal からの再ホストには制限があり、デバイスごとに取得される PermissionTicket は、新しいライセンスが取得されるまで 1 つだけに限定されます。これは、PermissionTicket と RehostTicket はそれぞれ、デバイスごとに常に 1 つだけ存在することを意味します。

この関数は、再ホスト プロセスの 2 番目のステップです。再ホスト プロセスは、Cisco Product License Registration Portal からの許可チケットの取得、デバイスからの再ホスト チケットの取得、Cisco Product License Registration Portal への再ホスト チケットの送信によるライセンスの取得、宛先デバイスへのライセンスの配備など、いくつかのステップから成ります。

取得した PermissionTicket はローカル ストレージに格納され、後で転送元デバイスからライセンスを取り消すときに使用されます。取り消し操作が成功した場合は PermissionTicket が削除され、再ホスト プロセスの次のステップのために RehostTicket がローカル ストレージに格納されます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
rehost_req	RehostRequest、必須	—	要求を表すオブジェクト。

戻り値
この関数は Status オブジェクトを返します。

エラーと例外
システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

操作が失敗した場合、Status には SUCCESS 以外のエラー コードとエラー メッセージが含まれます。

transferRMADeviceLicenses

構文
```
Status transferRMADeviceLicenses(UserToken token, String source_dev_udi, String
dest_dev_udi, boolean deploy) throws RemoteException;
```

説明
この関数は、Return Material Authorization（RMA）デバイスから新しいデバイスにライセンスを転送します。Boolean 型の deploy 値を true に設定すると、ライセンスが新しいデバイスに配備されます。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
Source_dev_udi	String	—	RMA デバイスの UDI。
Dest_dev_udi	String	—	ターゲット デバイスの UDI。
deploy	Boolean	—	ライセンスをターゲット デバイスに配備する場合は True。

戻り値
この関数は Status オブジェクトを返します。

エラーと例外
システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

writeLicenses

構文
```
IDStatus writeLicenses(UserToken token, License[] lics) throws RemoteException;
```

説明
この関数は、入力ライセンス オブジェクトをインベントリに書き込みます。入力ライセンス オブジェクトには、関数 readLicenses() によってインベントリから取得された既存のライセンス オブジェクトを指定します。

入力パラメータ

パラメータ	タイプ	値	説明
token	UserToken、必須	—	ユーザの認証パスを表すトークン。これは、ユーザが login 関数を呼び出して、バックエンド サーバがそのユーザを認証した後に取得されます。
lics	License の配列、必須	有効なライセンス ID を持つ License オブジェクトの配列。	License オブジェクトの配列。

戻り値

この関数は IDStatus オブジェクトを返します。次の例では、ステータスに含まれるエラー コード、エラー メッセージ、および戻りオブジェクトを取得しています。

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

// 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();
}
```

エラーと例外

システム エラーによって操作が完了しなかった場合は、RemoteException がスローされます。

エラーが発生した場合は、エラーに関する情報を含む IDStatus オブジェクトが返されます。