CHAPTER 3

SCA BB Service Configuration API を使用したプログラミング

この章では、Cisco SCA BB Service Configuration API（Service Configuration API）の主なクラスおよび方式について説明します。プログラミングに関するガイドラインおよびコードの例も示します。

Service Configuration API パッケージ

Service Configuration API には次のパッケージが含まれています。

- Service Configuration Management API
  - com.cisco.scabb.servconf.mgmt
  - com.pcube.apps.engage
- Service Configuration Editing API
  - com.cisco.scasbb.backend.classification — さまざまなモデルの説明を提供します。
  - com.pcube.apps.engage.common — Policy および Subscriber クラスで使用されるクラスを提供します。
  - com.pcube.apps.engage.policy — サービス コンフィギュレーションを定義するクラスを提供します。

ここでは、com.cisco.scabb.servconf.mgmt についてのみ説明します。その他のパッケージの詳細については、Service Configuration API ディストリビューション（「インストール内容」[p.2-2] を参照）の一部である Javadoc を参照してください。

Package com.cisco.scabb.servconf.mgmt

- SCABB — SCE プラットフォームとの接続方法、動作の適用と取得に使用する方式を提供
- ConnectionApi — SCE プラットフォームとの接続
- ImportExportApi — PQB ファイルに対するサービス コンフィギュレーションの保存方法、読み取り方法を提供。また、CSV ファイルに対するサービス コンフィギュレーションの部分インポート方法、部分エクスポート方法を提供
- ServiceConfigApi — 新しいサービス コンフィギュレーションの作成方法、および SCE プラットフォームに対するサービス コンフィギュレーションの適用方法、取得方法を提供
- ServiceConfig — このクラスのインスタンスは、SCE プラットフォームに適用される場合、サービス コントロール アプリケーションのネットワーク トラフィックに対する分類方法、アカウンティングとレポートの方法、および制御方法を決定するコンフィギュレーション パラメータのコンテナです。

Class SCABB の概要

Class SCABB 方式の概要

ログイン

構文

```
public static ConnectionApi login(String hostName,
String userName,
String password,
byte deviceType)
throws ConnectionFailedException
```

説明

SCE プラットフォームに接続します。この方式は、デバイスに影響を与えるすべての API 方式で使用されるハンドルを返します。すべてのログイン操作は、*logout(ConnectionApi)* 操作で終了する必要があります。

パラメータ

- ***hostName*** — SCE ホストのアドレス
- ***userName*** — ユーザ名
- ***password*** — パスワード
- ***deviceType*** — デバイスのタイプ。デバイス タイプ *Connection.SE_DEVICE* を使用して SCE プラットフォームに接続します。

戻り値

接続、つまりすべての API 方式で渡されるハンドルです。

例外

この方式には、次の例外があります。

- ConnectionFailedException — ログインが失敗した場合。失敗の理由を取得できます。

ログイン

構文

```
public static ConnectionApi login(com.pcube.management.framework.client.SessionObject
sessionObject)
```

説明

既存の *SessionObject* を使用してデバイスに接続します。この方式は、デバイスに影響を与えるすべての API 方式で使用されるハンドルを返します。

すべてのログイン操作は、*logout(ConnectionApi)* 操作で終了する必要があります。

パラメータ

- ***sessionObject*** — デバイスの既存の SessionObject

戻り値

接続、つまりすべての API 方式で渡されるハンドルです。

ログアウト

構文

```
public static void logout(ConnectionApi connectionApi)
```

説明

SCE からの接続解除。この方式が呼び出されたあと、接続は使用できません。

パラメータ

- ***connection*** — SCE への接続用のハンドルを保持する接続

addNotificationListener

構文

```
public static void addNotificationListener(NotificationListener listener)
```

説明

SCABB API で実行されたすべての操作に関する通知を受信するため、指定された SCABB 通知リスナーを承認します。

リスナーは NotificationListener インターフェイスを実装している必要があります。

パラメータ

- ***listener*** — NotificationListener

removeNotificationListener

構文

```
public static void removeNotificationListener(NotificationListener listener)
```

説明

SCABB API 操作に関する通知の受信を SCABB リスナー上で解除します。

受信解除されたリスナーは、その後通知を受け取りません。

パラメータ

- ***listener*** — 削除される NotificationListener

例外

この方式には、次の例外があります。

- IllegalArgumentException — リスナーが SCABB 通知の受信を承認されない場合

getDefaultProtocolFamilies

構文

```
public static InputStream getDefaultProtocolFamilies()
```

説明

このクラス用のデフォルトのプロトコル ファミリを取得します。

戻り値

プロトコル ファミリの InputStream

参照先

Class.getResourceAsStream(java.lang.String)

Class ConnectionApi の概要

Class ConnectionApi は、すべての Service Configuration API 方式および Subscriber API 方式で使用される SCE への接続用ハンドルです。

Class ConnectionApi 方式については、次のセクションで説明します。

Class ConnectionApi 方式の概要

isConnected

構文

```
public boolean isConnected()
```

説明

接続が有効かどうかチェックします。

戻り値

デバイスに接続されている場合は true、それ以外は false

Class ImportExportApi の概要

Class ImportExportApi は、操作のインポートおよびエクスポート用の API です。サービス コンフィギュレーションのエレメントは CSV 形式でインポートおよびエクスポートされます。サービス コンフィギュレーションは XML 形式でインポートおよびエクスポートされます。

Class ImportExportApi 方式については、次のセクションで説明します。

Class ImportExportApi Constructor の概要

ImportExportApi

構文

```
public ImportExportApi()
```

説明

ImportExportApi コンストラクタ

Class ImportExportApi 方式の概要

exportServiceConfiguration

構文

```
public static void exportServiceConfiguration(ServiceConfig servConf,
File f)
throws FileNotFoundException,
ImportExportException
```

説明

サービス コンフィギュレーションをファイルにエクスポートします。

パラメータ

- ***servConf***— エクスポートするサービス コンフィギュレーション
- ***f***— サービス コンフィギュレーションのエクスポート先のファイル

例外

この方式には、次の例外があります。

- FileNotFoundException
- ImportExportException — エクスポート中にエラーが発生した場合

importServiceConfiguration

構文

```
public static ServiceConfig importServiceConfiguration(File f)
throws ImportExportException,
IOException
```

説明

指定されたファイルからサービス コンフィギュレーションをインポートします。

パラメータ

- ***f***— インポートするサービス コンフィギュレーションを格納するファイル

戻り値

インポートされたサービス コンフィギュレーション

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合
- IOException

importFlavors

構文

```
public static void importFlavors(ServiceConfig servConf,
FlavorType flavorType,
File file)
throws ImportExportException
```

説明

指定された CSV ファイルから特定のフレーバ タイプをインポートします。

パラメータ

- ***servConf*** — フレーバのインポート先となるサービス コンフィギュレーション
- ***flavorType*** — インポートされたフレーバのタイプ
- ***file*** — インポートを行う CSV ファイル

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

importFlavors

構文

```
public static void importFlavors(ServiceConfig servConf,
FlavorType flavorType,
InputStream inStream)
throws ImportExportException
```

説明

指定された入力ストリームから特定のフレーバ タイプをインポートします。

パラメータ

- ***servConf*** — フレーバのインポート先となるサービス コンフィギュレーション
- ***flavorType*** — インポートされたフレーバのタイプ
- ***inStream*** — インポートを行う入力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

importZones

構文

```
public static void importZones(ServiceConfig servConf,
File file)
throws ImportExportException
```

説明

指定された CSV ファイルからゾーンをインポートします。

パラメータ

- ***servConf*** — ゾーンのインポート先となるサービス コンフィギュレーション
- ***file*** — インポートを行う CSV ファイル

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

importZones

構文

```
public static void importZones(ServiceConfig servConf,
InputStream inStream)
throws ImportExportException
```

説明

特定の入力ストリームからゾーンをインポートします。

パラメータ

- ***servConf*** — ゾーンのインポート先となるサービス コンフィギュレーション
- ***inStream*** — インポートを行う入力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

importProtocols

構文

```
public static void importProtocols(ServiceConfig servConf,
File file)
throws ImportExportException
```

説明

指定された CSV ファイルからプロトコルをインポートします。

パラメータ

- ***servConf*** — プロトコルのインポート先となるサービス コンフィギュレーション
- ***file*** — インポートを行う CSV ファイル

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

importProtocols

構文

```
public static void importProtocols(ServiceConfig servConf,
InputStream inStream)
throws ImportExportException
```

説明

特定の入力ストリームからプロトコルをインポートします。

パラメータ

- ***servConf*** — プロトコルのインポート先となるサービス コンフィギュレーション
- ***inStream*** — インポートを行う入力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

importServices

構文

```
public static void importServices(ServiceConfig servConf,
File file)
throws ImportExportException
```

説明

指定された CSV ファイルからサービスをインポートします。

パラメータ

- ***servConf*** — サービスのインポート先となるサービス コンフィギュレーション
- ***file*** — インポートを行う CSV ファイル

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

importServices

構文

```
public static void importServices(ServiceConfig servConf,
InputStream inStream)
throws ImportExportException
```

説明

特定の入力ストリームからサービスをインポートします。

パラメータ

- ***servConf*** — サービスのインポート先となるサービス コンフィギュレーション
- ***inStream*** — インポートを行う入力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — インポート中にエラーが発生した場合

exportProtocols

構文

```
public static void exportProtocols(List protocols,
File file)
throws ImportExportException
```

説明

プロトコルを指定されたファイルへ CSV 形式でエクスポートします。

パラメータ

- ***protocols*** — エクスポートするプロトコルのリスト
- ***file*** — エクスポートを行うファイル

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

exportProtocols

構文

```
public static void exportProtocols(List protocols,
OutputStream outStream)
throws ImportExportException
```

説明

プロトコルを指定された出力ストリームへ CSV 形式でエクスポートします。

パラメータ

- ***protocols*** — エクスポートするプロトコルのリスト
- ***outStream*** — エクスポートを行う出力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

exportZones

構文

```
public static void exportZones(List zones,
File file)
throws ImportExportException
```

説明

ゾーンを指定されたファイルへ CSV 形式でエクスポートします。

パラメータ

- ***zones*** — エクスポートするゾーンのリスト
- ***file*** — エクスポートを行うファイル

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

exportZones

構文

```
public static void exportZones(List zones,
OutputStream outStream)
throws ImportExportException
```

説明

ゾーンを指定された出力ストリームへ CSV 形式でエクスポートします。

パラメータ

- ***zones*** — エクスポートするゾーンのリスト
- ***outStream*** — エクスポートを行う出力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

exportFlavors

構文

```
public static void exportFlavors(List flavors,
FlavorType flavorType,
File file)
throws ImportExportException
```

説明

特定のフレーバ タイプを指定されたファイルへ CSV 形式でエクスポートします。

パラメータ

- ***flavors*** — エクスポートするフレーバのリスト
- ***flavorType*** — エクスポートされたフレーバのタイプ
- ***file*** — エクスポートを行うファイル

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

exportFlavors

構文

```
public static void exportFlavors(List flavors,
FlavorType flavorType,
OutputStream outStream)
throws ImportExportException
```

説明

特定のフレーバ タイプを指定された出力ストリームへ CSV 形式でエクスポートします。

パラメータ

- ***flavors*** — エクスポートするフレーバのリスト
- ***flavorType*** — エクスポートされたフレーバのタイプ
- ***outStream*** — エクスポートを行う出力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

exportServices

構文

```
public static void exportServices(ServiceConfig servConf,
File file)
throws ImportExportException
```

説明

サービスをサービス コンフィギュレーションからファイルへ CSV 形式でエクスポートします。

パラメータ

- ***servConf*** — サービスのエクスポート元となるサービス コンフィギュレーション
- ***file*** — エクスポートを行うファイル

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

exportServices

構文

```
public static void exportServices(ServiceConfig servConf,
OutputStream outStream)
throws ImportExportException
```

説明

サービスをサービス コンフィギュレーションから出力ストリームへ CSV 形式でエクスポートします。

パラメータ

- servConf — サービスのエクスポート元となるサービス コンフィギュレーション
- outStream — エクスポートを行う出力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException — エクスポート中にエラーが発生した場合

loadListArray

構文

```
public static void loadListArray(ServiceConfig servConf,
InputStream inStream)
throws ImportExportException
```

説明

Deprecated — 2.57 ホストおよび IP リストでのみ使用します。

- 2.57 CSV ホスト リスト ファイルを 3.0 CSV HTTP URL フレーバ ファイルに変換します。
- 2.57 CSV IP リスト ファイルを 3.0 CSV ゾーン ファイルに変換します。

3.0 CSV ファイルをロードするには、importFlavors（ServiceConfig、FlavorType、File）および importZones（ServiceConfig、File）を使用します。

パラメータ

- ***servConf*** — サービスのインポート先となるサービス コンフィギュレーション
- ***inStream*** — インポートを行う入力ストリーム

例外

この方式には、次の例外があります。

- ImportExportException

Class ServiceConfigApi の概要

Class ServiceConfigApi では、Service Configuration API 方式を公開します。これらのすべての方式では、SCE に格納されたデータの取得または設定を行います。このため、SCE プラットフォームへの接続が必要です。

Class ServiceConfigApi 方式については、次のセクションで説明します。

Class ServiceConfigApi 方式

applyServiceConfiguration

構文

```
public static long applyServiceConfiguration(ConnectionApi connectionApi,
ServiceConfig servConf)
throws ElementManagementException,
ApplyException
```

説明

サービス コンフィギュレーションを指定された SCE に適用します。

パラメータ

- ***connectionApi*** — SCE への接続用ハンドルを保持する ConnectionApi
- ***servConf*** — 適用するサービス コンフィギュレーション

戻り値

操作タイムスタンプ

例外

この方式には、次の例外があります。

- ElementManagementException
- ApplyException

applyServiceConfiguration

構文

```
public static long applyServiceConfiguration(ConnectionApi connectionApi,
ServiceConfig servConf,
Properties applySettings)
throws ElementManagementException,
ApplyException
```

説明

サービス コンフィギュレーションを指定された SCE に適用します。

パラメータ

- connectionApi — SCE への接続用ハンドルを保持する ConnectionApi
- servConf — 適用するサービス コンフィギュレーション
- applySettings — プロパティ

戻り値

操作タイムスタンプ

例外

この方式には、次の例外があります。

- ElementManagementException
- ApplyException

applyServiceConfiguration

構文

```
public static long applyServiceConfiguration(ConnectionApi connectionApi,
ServiceConfig servConf,
boolean updateCm,
Properties cmIpRemap,
int cmUpdateMethod,
int rpcPort)
throws ElementManagementException,
ApplyException
```

説明

サービス コンフィギュレーションを指定された SCE に適用します。

パラメータ

- ***connectionApi*** — SCE への接続用ハンドルを保持する ConnectionApi
- ***servConf*** — 適用するサービス コンフィギュレーション
- ***updateCm*** — 指定された SCE 用の Collection Manager が指定されたサービス コンフィギュレーション値を使用してアップデートされたかどうか
- ***cmIpRemap*** — SCE に設定された Collection Manager の IP アドレスから実際の Collection Manager のアドレスへのマップ
- ***cmUpdateMethod*** — Collection Manager への接続に使用する方式
- ***rpcPort*** — Collection Manager の RPC 接続に使用するポート番号

戻り値

操作タイムスタンプ

例外

この方式には、次の例外があります。

- ElementManagementException
- ApplyException

参照先

PolicyAPI.DC_UPDATE_METHOD_RPC, *PolicyAPI.DC_DEFAULT_RPC_PORT*

retrieveServiceConfiguration

構文

```
public static ServiceConfig retrieveServiceConfiguration(ConnectionApi connectionApi)
throws IOException,
ElementManagementException,
ApplyException
```

説明

指定された SCE にロードされたサービス コンフィギュレーションを取得します。

パラメータ

- ***connectionApi*** — SCE への接続用ハンドルを保持する ConnectionApi

戻り値

SCE プラットフォーム内のサービス コンフィギュレーション

例外

この方式には、次の例外があります。

- IOException
- ElementManagementException
- ApplyException

updateValuesIni

構文

```
public static void updateValuesIni(String cmAddress,
String sceAddress,
ServiceConfig servConf)
throws ApplyException
```

説明

SCE プラットフォームのサービス コンフィギュレーションから取得したデータを使用して Collection Manager をアップデートします。

パラメータ

- ***cmAddress*** — アップデートする Collection Manager のアドレス
- ***sceAddress*** — 特定のサービス コンフィギュレーションを実行する SCE プラットフォームのアドレス
- ***servConf*** — サービス コンフィギュレーション

例外

この方式には、次の例外があります。

- ApplyException

updateValuesIni

構文

```
public static void updateValuesIni(String cmAddress,
String sceAddress,
ServiceConfig servConf,
int cmUpdateMethod,
int rpcPort)
throws ApplyException
```

説明

SCE プラットフォームのサービス コンフィギュレーションから取得したデータを使用して Collection Manager をアップデートします。

パラメータ

- ***cmAddress*** — アップデートする Collection Manager のアドレス
- ***sceAddress*** — 特定のサービス コンフィギュレーションを実行する SCE プラットフォームのアドレス
- ***servConf*** — サービス コンフィギュレーション
- ***cmUpdateMethod***
- ***rpcPort***

例外

この方式には、次の例外があります。

- ApplyException

validateServiceConfiguration

構文

```
public static ArrayList validateServiceConfiguration(ServiceConfig servConf)
```

説明

サービス コンフィギュレーションを検証します。

パラメータ

- servConf — 検証するサービス コンフィギュレーション

戻り値

エラーになる可能性のあるルールに対する警告メッセージを持つベクトル

参照先

PolicyValidator.validatePolicy(com.pcube.apps.engage.policy.Policy)

importServConf

構文

```
public static ServiceConfig importServConf(File pqbfile)
throws ImportExportException,
IOException
```

説明

PQB ファイルからサービス コンフィギュレーションをロードします。

パラメータ

- ***pqbfile*** — ロードする PQB ファイル

戻り値

結果として生じるサービス コンフィギュレーション

例外

この方式には、次の例外があります。

- ImportExportException
- IOException

exportServConf

構文

```
public static void exportServConf(ServiceConfig servConf,
File pqbfile)
throws FileNotFoundException,
ImportExportException
```

説明

サービス コンフィギュレーションを PQB ファイルに保存します。

パラメータ

- ***servConf*** — 保存するサービス コンフィギュレーション
- ***pqbfile*** — サービス コンフィギュレーションの保存先となる PQB ファイル

例外

この方式には、次の例外があります。

- FileNotFoundException
- ImportExportException

importDefaultServConf

構文

```
public static ServiceConfig importDefaultServConf()
throws ImportExportException
```

説明

デフォルトのサービス コンフィギュレーションをロードします。

戻り値

デフォルトのサービス コンフィギュレーション

例外

この方式には、次の例外があります。

- ImportExportException

Class ServiceConfig の概要

Class ServiceConfig は、ISP が定義したリスト、プロトコル、サービス、およびパッケージの包括的なセットです。サービス コンフィギュレーションは ServiceConfig ドメインの SCE プラットフォームに適用されます。また、アプリケーション パラメータを設定して加入者フローの調整を行います。

Class ServiceConfig 方式については、次のセクションで説明します。

- Class ServiceConfig 方式（p.3-25）

Class ServiceConfig 方式

- getCalendarList（p.3-25）
- getClassificationCfg（p.3-26）
- getDynamicSignatureScript（p.3-26）
- getPackageList（p.3-26）
- getPolicySettings（p.3-27）
- getProtocolRedirectIndexNameArray（p.3-27）
- getProtocolRedirectString（p.3-27）
- getRealTimeFrameName（p.3-28）
- getServiceList（p.3-28）
- getSubNotifications（p.3-29）
- getTimeFrameNames（p.3-29）
- getProtocolRedirectString（p.3-30）
- getZoneList（p.3-30）
- isProtocolRedirectable（p.3-31）
- setProtocolRedirectString（p.3-31）
- setProtocolRedirectString（p.3-32）
- setTimeFrameName（p.3-33）
- setTimeFrameName（p.3-33）
- setTimeFrameNames（p.3-34）

getCalendarList

- 構文（p.3-25）
- 説明（p.3-25）
- 戻り値（p.3-25）

構文

```
public CalendarArray getCalendarList()
```

説明

カレンダー リストを取得します。

戻り値

サービス コンフィギュレーション内のカレンダー リスト

getClassificationCfg

構文

```
public ClassificationConfiguration getClassificationCfg()
```

説明

分類設定を取得します。

戻り値

ドメイン内の設定に関する分類

getDynamicSignatureScript

構文

```
public DynamicSignaturesScript getDynamicSignatureScript()
```

説明

ダイナミック シグニチャ設定を取得します。

戻り値

ダイナミック シグニチャのスクリプト

getPackageList

構文

```
public PackageArray getPackageList()
```

説明

サービス コンフィギュレーションのパッケージ リストを取得します。

戻り値

サービス コンフィギュレーションのパッケージ リスト

getPolicySettings

構文

```
public PolicySettings getPolicySettings()
```

説明

サービス コンフィギュレーションの設定を取得します。これらの設定は、このサービス コンフィギュレーションのドメイン内にある SCE プラットフォーム用の一般的なシステム設定です。

戻り値

サービス コンフィギュレーションの設定

getProtocolRedirectIndexNameArray

構文

```
public ProtocolRedirectIndexNameArray getProtocolRedirectIndexNameArray()
```

説明

プロトコルのリダイレクト インデックス名のリストを取得します。

戻り値

プロトコルのリダイレクト インデックス名のリスト

getProtocolRedirectString

構文

```
public String getProtocolRedirectString(String protocolName,
int redirectIndex)
throws ItemNotFoundException
```

説明

リダイレクト ストリング アレイ内の特定のインデックス上にあるプロトコルに使用するリダイレクト アドレスを取得します。

リダイレクションはプロトコルの仕様の一部であり、少数の事前定義されたプロトコル用としてのみ存在します。

パラメータ

- ***protocolName*** — クエリーされたプロトコル
- ***redirectIndex*** — リダイレクト ストリング アレイのインデックス

戻り値

リダイレクト アドレス

例外

この方式には、次の例外があります。

- ItemNotFoundException — このサービス コンフィギュレーションの ProtocolArray 内に該当する事前定義プロトコルが存在しない場合、またはリダイレクト インデックスが範囲外の場合

getRealTimeFrameName

構文

```
public String getRealTimeFrameName(String name)
throws ItemNotFoundException
```

説明

特定の TimeFrame 用に事前定義された API 名を取得します。

パラメータ

- ***name*** — TimeFrame 用のこのサービス コンフィギュレーションのエイリアス

戻り値

TimeFrame の API 名

例外

この方式には、次の例外があります。

- ItemNotFoundException — このサービス コンフィギュレーションに該当するエイリアスが存在しない場合

getServiceList

構文

```
public ServiceArray getServiceList()
```

説明

サービス コンフィギュレーションのサービス リストを取得します。

戻り値

サービス コンフィギュレーションのサービス リスト

getSubNotifications

構文

```
public SubNotificationArray getSubNotifications()
```

説明

サービス コンフィギュレーションに設定された加入者通知を取得します。

戻り値

加入者通知

getTimeFrameNames

構文

```
public String[] getTimeFrameNames()
```

説明

このサービス コンフィギュレーションによって異なる TimeFrame に割り当てられたタイムフレーム名を取得します。

これらのエイリアスでは、タイムフレームに意味のある名前を付けることができます。

返されたストリング アレイは、インデックス X（TimeFrame のインデックス X のエイリアス）に保存されます。

戻り値

このサービス コンフィギュレーションのタイムフレーム名のストリング アレイ

getProtocolRedirectString

構文

```
public String getProtocolRedirectString(String protocolName)
throws ItemNotFoundException
```

説明

プロトコル用のデフォルトのリダイレクト アドレスを取得します。

リダイレクションはプロトコルの仕様の一部であり、少数の事前定義されたプロトコル用としてのみ存在します。

パラメータ

- ***protocolName*** — クエリーされたプロトコル

戻り値

リダイレクト アドレス

例外

この方式には、次の例外があります。

- ItemNotFoundException — このサービス コンフィギュレーションの ProtocolArray 内に該当する事前定義プロトコルが存在しない場合

getZoneList

構文

```
public ZoneList getZoneList()
```

説明

サービス コンフィギュレーションの IP、IP 範囲、およびホスト リスト アレイを取得します。これらのリストはこのサービス コンフィギュレーションのサービスによって参照されます。

戻り値

サービス コンフィギュレーションのリストの配列

isProtocolRedirectable

構文

```
public boolean isProtocolRedirectable(String protocolName)
throws ItemNotFoundException
```

説明

指定されたプロトコルがリダイレクションをサポートしているかチェックします。

リダイレクションはプロトコルの仕様の一部であり、少数の事前定義されたプロトコル用としてのみ存在します。

パラメータ

- ***protocolName*** — クエリーされたプロトコルの名前

戻り値

プロトコルがリダイレクションをサポートしている場合は true、それ以外は false

例外

この方式には、次の例外があります。

- ItemNotFoundException — このサービス コンフィギュレーションの ProtocolArray 内に該当する事前定義プロトコルが存在しない場合

setProtocolRedirectString

構文

```
public void setProtocolRedirectString(String protocolName,
int redirectIndex,
String value)
throws ItemNotFoundException,
MalformedURLException
```

説明

一定のフローをリダイレクトするアドレスを設定します。要求されたプロトコルを使用する一部のサービス向けの *ACCESS_BLOCK_AND_REDIRECT* アクセス モードがルールに含まれる場合、リダイレクションが発生します。リダイレクト ストリング アレイ内に一定の String が発生すると設定が行われます。使用するリダイレクト String はリダイレクト ストリング アレイから決定されます。

パラメータ

- ***protocolName*** — アクティビティがリダイレクトされるプロトコル
- ***redirectIndex*** — プロトコルのリダイレクト ストリング アレイ内にあるリダイレクト String のインデックス
- ***value*** — プロトコルのリダイレクト アドレス

例外

この方式には、次の例外があります。

- ItemNotFoundException — このサービス コンフィギュレーションの ProtocolArray 内に該当する事前定義プロトコルが存在しない場合、またはリダイレクト インデックスが範囲外の場合
- MalformedURLException — String が不正な URL 名である場合

setProtocolRedirectString

- 構文（p.3-32）
- 説明（p.3-32）
- パラメータ（p.3-32）
- 例外（p.3-32）

構文

```
public void setProtocolRedirectString(String protocolName,
String value)
throws ItemNotFoundException,
MalformedURLException
```

説明

一定のフローをリダイレクトするデフォルト アドレスを設定します。要求されたプロトコルを使用する一定のサービスに対する *ACCESS_BLOCK_AND_REDIRECT* アクセス モードがルールに含まれている場合、リダイレクションが発生します。

パラメータ

- ***protocolName*** — アクティビティがリダイレクトされる要求プロトコル
- ***value*** — プロトコルのリダイレクト アドレス

例外

この方式には、次の例外があります。

- ItemNotFoundException — このサービス コンフィギュレーションの ProtocolArray 内に該当する事前定義プロトコルが存在しない場合
- MalformedURLException — String が不正な URL 名である場合

setTimeFrameName

構文

```
public void setTimeFrameName(int index,
String newName)
throws ItemNotFoundException,
DuplicateItemException
```

説明

特定のインデックスを持つ TimeFrame 用のエイリアスを設定します。

エイリアスでは、タイムフレームに意味のある名前を付けることができます。

パラメータ

- ***index*** — エイリアスが与えられる TimeFrame のインデックス
- ***newName*** — エイリアス

例外

この方式には、次の例外があります。

- ItemNotFoundException — 該当する TimeFrame が存在しない場合
- DuplicateItemException

setTimeFrameName

構文

```
public void setTimeFrameName(TimeFrame frame,
String newName)
throws ItemNotFoundException,
DuplicateItemException
```

説明

指定された TimeFrame 用のエイリアスを設定します。エイリアスでは、タイムフレームに意味のある名前を付けることができます。

パラメータ

- frame — エイリアスが与えられる TimeFrame
- newName — エイリアス

例外

この方式には、次の例外があります。

- ItemNotFoundException — 該当する TimeFrame が存在しない場合

- DuplicateItemException

setTimeFrameNames

構文

```
public void setTimeFrameNames(String[] newNames)
```

説明

すべてのタイムフレームの名前を設定します。

パラメータ

- ***newNames*** — 設定する名前

Service Configuration API のプログラミングに関するガイドライン

SCE プラットフォームへの接続

いずれかの login() 方式を使用して作成した接続は、logout() によって適切に終了している必要があります（「Class SCABB の概要」[p.3-3] を参照）。

Service Configuration API のコード例

ここでは、Service Configuration API の使用に関する複数のコード例を示します。

サービス コンフィギュレーションの適用

次の例では、新しいサービス コンフィギュレーションを、コマンドラインで指定された SCE プラットフォームに適用します。

```
package examples;
import java.io.File;
import com.cisco.scabb.servconf.mgmt.ConnectionApi;
import com.cisco.scabb.servconf.mgmt.ImportExportApi;
import com.cisco.scabb.servconf.mgmt.SCABB;
import com.cisco.scabb.servconf.mgmt.ServiceConfig;
import com.cisco.scabb.servconf.mgmt.ServiceConfigApi;
import com.pcube.apps.engage.Connection;
import com.pcube.apps.engage.ConnectionFailedException;
/**
* applies the service configuration in the PQB file to the SCE
* specified in the command line. message is printed to standard error
* in case of failure.
* <p>
* usage: java examples.SimpleApplyPqb <sce-address><password>
* <pqb-filename>
*/
public class SimpleApplyPqb {
public static void main(String[] args) {
if (args.length != 3) {
System.err.println("usage: java examples.SimpleApplyPqb "
+ "<sce-address><password><pqb-filename>");
System.exit(1);
}
String sceAddress = args[0];
String password = args[1];
String pqbFilename = args[2];
ServiceConfig serviceConfig = openPqbFile(pqbFilename);
if (serviceConfig == null) {
return;
}
applyPqb(sceAddress, password, serviceConfig);
}
/**
* apply the service configuration in the specified PQB file to the
* specified SCE. message is printed to standard error in case of
* failure.
*
* @param sceAddress
* @param password
* @param serviceConfig
*/
private static void applyPqb(String sceAddress, String password,
ServiceConfig serviceConfig) {
ConnectionApi connection = null;
try {
System.out.println("connecting to SCE at " + sceAddress);
connection = SCABB.login(sceAddress, "admin", password,
Connection.SE_DEVICE);
System.out.println("connected to SCE");
System.out.println("applying service configuration");
ServiceConfigApi.applyServiceConfiguration(connection,
```

```
serviceConfig);
System.out.println("service configuration applied");
} catch (ConnectionFailedException e) {
System.err.println("connection to SCE failed: "
+ e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.err.println("apply operation failed: "
+ e.getMessage());
e.printStackTrace();
} finally {
if (connection != null) {
System.out.println("disconnecting from SCE");
SCABB.logout(connection);
System.out.println("disconnected");
}
}
}
/**
* return the service configuration in the specified PQB file, or
* null if reading the file has failed. message is printed to
* standard error in case of failure.
*
* @param pqbFilename
* @return
*/
private static ServiceConfig openPqbFile(String pqbFilename) {
ServiceConfig serviceConfig = null;
try {
System.out.println("opening PQB file " + pqbFilename);
serviceConfig = ImportExportApi
.importServiceConfiguration(new File(pqbFilename));
System.out.println("PQB file opened");
} catch (Exception e) {
System.err.println("opening PQB file failed: "
+ e.getMessage());
e.printStackTrace();
}
return serviceConfig;
}
}
```

ゾーンの自動アップデート

次の例では、ゾーン IP アドレスを使用して SCE をアップデートします。ゾーン IP アドレスは CSV ファイルで指定されます。

```
package examples;
import java.io.File;
import com.cisco.scabb.servconf.mgmt.ConnectionApi;
import com.cisco.scabb.servconf.mgmt.ImportExportApi;
import com.cisco.scabb.servconf.mgmt.SCABB;
import com.cisco.scabb.servconf.mgmt.ServiceConfig;
import com.cisco.scabb.servconf.mgmt.ServiceConfigApi;
import com.cisco.scasbb.backend.classification.Zone;
import com.pcube.apps.engage.Connection;
import com.pcube.apps.engage.ConnectionFailedException;
import com.pcube.apps.engage.common.ImportExportException;
/**
* updates an SCE with zone IP addresses. the zone IP address are
* specified in a CSV file. the SCE address and CSV filename are taken
* from the cmd-line argument.
* <p>
* usage: java examples.UpdateZoneFromCsv <sce-address><password>
* <zone-csv-file><zone-name>
*
*/
```

```
public class UpdateZoneFromCsv {
public static void main(String[] args) {
if (args.length != 4) {
System.err.println("usage: java examples.UpdateZoneFromCsv"
+ " <sce-address><password>"
+ " <zone-csv-file><zone-name>");
System.exit(1);
}
String sceAddress = args[0];
String password = args[1];
String csvFilename = args[2];
String zoneName = args[3];
ServiceConfig serviceConfig = retrievePqb(sceAddress, password);
if (serviceConfig == null) {
return;
}
ServiceConfig updatedServiceConfig = importZoneFromCsv(
serviceConfig, csvFilename, zoneName);
if (updatedServiceConfig == null) {
return;
}
applyPqb(sceAddress, password, updatedServiceConfig);
}
/**
* apply the service configuration in the specified PQB file to the
* specified SCE. message is printed to standard error in case of
* failure.
*
* @param sceAddress
* @param password
* @param serviceConfig
*/
private static void applyPqb(String sceAddress, String password,
ServiceConfig serviceConfig) {
ConnectionApi connection = null;
try {
System.out.println("connecting to SCE at " + sceAddress);
connection = SCABB.login(sceAddress, "admin", password,
Connection.SE_DEVICE);
System.out.println("connected to SCE");
System.out.println("applying service configuration");
ServiceConfigApi.applyServiceConfiguration(connection,
serviceConfig);
System.out.println("service configuration applied");

} catch (ConnectionFailedException e) {
System.err.println("connection to SCE failed: "
+ e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.err.println("apply operation failed: "
+ e.getMessage());
e.printStackTrace();
} finally {
if (connection != null) {
System.out.println("disconnecting from SCE");
SCABB.logout(connection);
System.out.println("disconnected");
}
}
}
private static ServiceConfig importZoneFromCsv(
ServiceConfig serviceConfig, String csvFilename,
String zoneName) {
// clear zone items
Zone zone = (Zone) serviceConfig.getClassificationCfg()
.getZoneList().findByName(zoneName);
if (zone == null) {
System.err.println("WARNING: zone not found: " + zoneName);
```

```
} else {
zone.getZoneItems().clear();
}
// import new zone items
try {
ImportExportApi.importZones(serviceConfig, new File(
csvFilename));
} catch (ImportExportException e) {
System.err.println("importing zones failed: "
+ e.getMessage());
e.printStackTrace();
return null;
}
return serviceConfig;
}
private static ServiceConfig retrievePqb(String sceAddress,
String password) {
ServiceConfig retrievedServiceConfig = null;
ConnectionApi connection = null;
try {
System.out.println("connecting to SCE at " + sceAddress);
connection = SCABB.login(sceAddress, "admin", password,
Connection.SE_DEVICE);
System.out.println("connected to SCE");
System.out.println("retrieving service configuration");
retrievedServiceConfig = ServiceConfigApi
.retrieveServiceConfiguration(connection);
System.out.println("service configuration retrieved");
} catch (ConnectionFailedException e) {
System.err.println("connection to SCE failed: "
+ e.getMessage());
e.printStackTrace();
} catch (Exception e) {
System.err.println("retrieve operation failed: "
+ e.getMessage());
e.printStackTrace();
} finally {
if (connection != null) {
System.out.println("disconnecting from SCE");
SCABB.logout(connection);

System.out.println("disconnected");
}
}
return retrievedServiceConfig;
}
}
```

サービス名およびパッケージ名のリスト

次の例では、サービス コンフィギュレーション内のサービス名およびパッケージ名を出力します。

```
package examples;
import java.io.File;
import java.io.IOException;
import java.util.Iterator;
import com.cisco.scabb.servconf.mgmt.ImportExportApi;
import com.cisco.scabb.servconf.mgmt.ServiceConfig;
import com.cisco.scabb.servconf.mgmt.ServiceConfigApi;
import com.pcube.apps.engage.common.ImportExportException;
import com.pcube.apps.engage.policy.Package;
import com.pcube.apps.engage.policy.Service;
public class IterateServiceConfig {
public static void main(String[] args)
throws ImportExportException, IOException {
// take the PQB filename from the cmd-line, or use the default
// service configuration instead
ServiceConfig serviceConfig = null;
if (args.length >0) {
serviceConfig = ImportExportApi
.importServiceConfiguration(new File(args[0]));
} else {
serviceConfig = ServiceConfigApi.importDefaultServConf();
}
System.out.println("-------- package names --------");
Iterator pkgIter = serviceConfig.getPackageList().iterator();
while (pkgIter.hasNext()) {
Package pkg = (Package) pkgIter.next();
System.out.println(pkg.getNumericId() + ": "
+ pkg.getName());
}
System.out.println("-------- service names --------");
Iterator svcIter = serviceConfig.getServiceList().iterator();
while (svcIter.hasNext()) {
Service svc = (Service) svcIter.next();
System.out.println(svc.getNumericId() + ": "
+ svc.getName());
}
}
}
```