CHAPTER 3

Programming with the SCA BB Service Configuration API

Revised: July 23, 2009, OL-7207-05

Introduction

This chapter is a reference for the main classes and methods of the Cisco SCA BB Service Configuration API. It also contains programming guidelines and code examples.

Service Configuration API Packages

The Service Configuration API includes the following packages.

- Service Configuration Management API
  - *com.cisco.scabb.servconf.mgmt*
  - *com.pcube.apps.engage*
- Service Configuration Editing API
  - *com.cisco.scasbb.backend.classification*—Provides representation of the various model objects
  - *com.pcube.apps.engage.common*—Provides the classes that are used by the Policy and Subscriber classes
  - *com.pcube.apps.engage.policy*—Provides the classes that define service configurations

Only *com.cisco.scabb.servconf.mgmt* is documented in this guide. For details of the other packages, refer to the Javadoc that is part of the Service Configuration API distribution (see Distribution Content, page 2-2).

Package com.cisco.scabb.servconf.mgmt

Package *com.cisco.scabb.servconf.mgmt* contains the following classes:

- SCABB—Provides methods for connecting to the SCE platform; these methods are used for apply and retrieve operations.
- ConnectionApi—Provides the connection handle to the SCE platform.
- ImportExportApi—Provides methods for saving service configurations to and reading service configurations from PQB files, and for importing and exporting parts of a service configuration to and from CSV files.
- ServiceConfigApi—Provides methods for creating a new service configuration, and for applying service configurations to and retrieving service configurations from SCE platforms.
- ServiceConfig—Instances of this class are containers of configuration parameters that, when applied to the SCE platform, determine how the service control application performs classification, accounting and reporting, and control over network traffic.

Class SCABB

Class SCABB supplies login and logout methods to the SCE. The handle returned by the login serves all of the Service Configuration API and Subscriber API methods that directly affect the SCE.

Class SCABB Methods

Class SCABB methods are explained in the following sections.

login

Syntax

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

Description

Connects to the SCE platform; the method returns a handle that is used by all the API methods that affect the device.

Every login operation must be closed with a *logout(ConnectionApi)* operation.

Parameters

- **hostName—**The SCE host address.
- **userName—**The user name.
- **password—**The password.
- **deviceType—**The device type. Use device type *Connection.SE_DEVICE* to connect to an SCE platform.

Return Value

The connection, which is a handle passed to all the API methods.

Exceptions

The following exception may be thrown by this method:

- ConnectionFailedException—Thrown if the login fails; you can retrieve the reason for the failure from the exception

login

Syntax

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

Description

Connects to a device by using an existing *SessionObject* ; the method returns a handle that is used by all the API methods which affect the device.

Every login operation must be closed with a *logout(ConnectionApi)* operation.

Parameters

- **sessionObject—**An existing SessionObject of the device

Return Value

The connection, which is a handle passed to all the API methods.

logout

Syntax

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

Description

Disconnects from the SCE; the connection cannot be used after this method is called.

Parameters

- **connection—**The connection that keeps a handle to the connection to the SCE

addNotificationListener

Syntax

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

Description

Subscribes the specified SCABB notification listener to receive notifications regarding all operations performed by the SCABB API.

The listener must implement the NotificationListener interface.

Parameters

- **listener—**The NotificationListener

removeNotificationListener

Syntax

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

Description

Unsubscribes the SCABB listener from receiving notifications regarding SCABB API operations.

The removed listener will not receive any more notifications.

Parameters

- **listener—**The NotificationListener to be removed

Exceptions

The following exception may be thrown by this method:

- IllegalArgumentException—If the listener was never subscribed to listen to SCABB notifications

getDefaultProtocolFamilies

Syntax

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

Description

Gets the default protocol families for this class

Return Value

An InputStream of the protocol families

See Also

Class.getResourceAsStream(java.lang.String)

Class ConnectionApi

Class ConnectionApi is a connection handle to the SCE that is used by all of the Service Configuration API and Subscriber API methods.

Class ConnectionApi Methods

Class ConnectionApi contains one method, which is explained in the following section.

isConnected

Syntax

```
public boolean isConnected()
```

Description

Checks if the connection is valid.

Return Value

`true` if connected to the device, `false` otherwise

Class ImportExportApi

Class ImportExportApi is an API for import and export operations. Service configuration elements are imported and exported in CSV formats. Service configuration are imported and exported in XML format.

- Class ImportExportApi Constructor, page 3-6
- Class ImportExportApi Methods, page 3-6

Class ImportExportApi Constructor

The Class ImportExportApi constructor is explained in the following section.

ImportExportApi

Syntax

```
public ImportExportApi()
```

Description

The ImportExportApi constructor.

Class ImportExportApi Methods

Class ImportExportApi methods are explained in the following sections.

exportServiceConfiguration

Syntax

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

Description

Exports a service configuration to a file.

Parameters

- **servConf—**The service configuration to export
- **f—**The file to which to export the service configuration

Exceptions

The following exceptions may be thrown by this method:

- FileNotFoundException
- ImportExportException—If an error occurs during export

importServiceConfiguration

Syntax

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

Description

Imports a service configuration from the specified file.

Parameters

- **f—**The file containing the service configuration to import

Return Value

The imported service configuration

Exceptions

The following exceptions may be thrown by this method:

- ImportExportException—If an error occurs during import
- IOException

importFlavors

Syntax

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

Description

Imports flavors of a specified type from a specified CSV file

Parameters

- **servConf—**The service configuration into which to import the flavors
- **flavorType—**The type of the imported flavors
- **file—**The CSV file from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

importFlavors

Syntax

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

Description

Imports flavors of a specified type from a given input-stream

Parameters

- **servConf—**The service configuration into which to import the flavors
- **flavorType—**The type of the imported flavors
- **inStream—**The input stream from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

importZones

Syntax

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

Description

Imports zones from a specified CSV file

Parameters

- **servConf—**The service configuration into which to import the zones
- **file—**The CSV file from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

importZones

Syntax

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

Description

Imports zones from a given input stream

Parameters

- **servConf—**The service configuration into which to import the zones
- **inStream—**The input stream from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

importProtocols

Syntax

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

Description

Imports protocols from a specified CSV file

Parameters

- **servConf—**The service configuration into which to import the protocols
- **file—**The CSV file from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

importProtocols

Syntax

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

Description

Imports protocols from a given input stream

Parameters

- **servConf—**The service configuration into which to import the protocols
- **inStream—**The input stream from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

importServices

Syntax

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

Description

Imports services from a specified CSV file

Parameters

- **servConf—**The service configuration into which to import the services
- **file—**The CSV file from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

importServices

Syntax

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

Description

Imports services from a given input stream

Parameters

- **servConf—**The service configuration into which to import the services
- **inStream—**The input stream from which to import

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during import

exportProtocols

Syntax

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

Description

Exports protocols to a specified file in a CSV format

Parameters

- **protocols—**The list of protocols to export
- **file—**The file to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

exportProtocols

Syntax

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

Description

Exports protocols to a given output stream in a CSV format

Parameters

- **protocols—**The list of protocols to export
- **outStream—**The output stream to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

exportZones

Syntax

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

Description

Exports zones to a specified file in a CSV format

Parameters

- **zones—**The list of zones to export
- **file—**The file to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

exportZones

Syntax

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

Description

Exports zones to a given output stream in a CSV format

Parameters

- **zones—**The list of zones to export
- **outStream—**The output stream to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

exportFlavors

Syntax

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

Description

Exports flavors of a specified type to a specified file in a CSV format

Parameters

- **flavors—**The list of flavors to export
- **flavorType—**The type of the exported flavors
- **file—**The file to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

exportFlavors

Syntax

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

Description

Exports flavors of a specified type to a given output stream in a CSV format

Parameters

- **flavors—**The list of flavors to export
- **flavorType—**The type of the exported flavors
- **outStream—**The output stream to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

exportServices

Syntax

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

Description

Exports services from a service configuration to a file in CSV format

Parameters

- **servConf—**The service configuration from which to export the services
- **file—**The file to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

exportServices

Syntax

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

Description

Exports services from a service configuration to an output stream in CSV format

Parameters

- servConf—The service configuration from which to export the services
- outStream—The output stream to which to export

Exceptions

The following exception may be thrown by this method:

- ImportExportException—If an error occurs during export

loadListArray

Syntax

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

Description

Deprecated—Only used for 2.57 host list and IP list:

- It converts 2.57 CSV host-list files into 3.0 CSV HTTP URL flavor files
- It converts 2.57 CSV IP-list files into 3.0 CSV zone files

For loading 3.0 CSV files use the methods importFlavors(ServiceConfig, FlavorType, File) and importZones(ServiceConfig, File)

Parameters

- **servConf—**The service configuration into which to import the services
- **inStream—**The input stream to import to

Exceptions

The following exception may be thrown by this method:

- ImportExportException

Class ServiceConfigApi

Class ServiceConfigApi exposes Service Configuration API methods. All of these methods get or set data located in the SCE, and therefore require a connection to the SCE platform.

Class ServiceConfigApi Methods

Class ServiceConfigApi methods are explained in the following sections.

applyServiceConfiguration

Syntax

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

Description

Applies a service configuration to the specified SCE.

Parameters

- **connectionApi—**The ConnectionApi, which keeps a handle to the connection to the SCE
- **servConf—**The service configuration to apply

Return Value

The operation timestamp

Exceptions

The following exceptions may be thrown by this method:

- ElementManagementException
- ApplyException

applyServiceConfiguration

Syntax

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

Description

Applies a service configuration to the specified SCE.

Parameters

- connectionApi—The ConnectionApi, which keeps a handle to the connection to the SCE
- servConf—The service configuration to apply
- applySettings—Properties

Return Value

The operation timestamp

Exceptions

The following exceptions may be thrown by this method:

- ElementManagementException
- ApplyException

applyServiceConfiguration

Syntax

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

Description

Applies a service configuration to the specified SCE.

Parameters

- **connectionApi—**The ConnectionApi, which keeps a handle to the connection to the SCE
- **servConf—**The service configuration to apply
- **updateCm—**Whether the Collection Manager for the specified SCE will be updated with the specified service configuration values
- **cmIpRemap—**A map from the CM IP address as configured in the SCE, to the actual CM addresses
- **cmUpdateMethod—**The method to use to connect to the CM
- **rpcPort—**Port number for the CM RPC connection
- **forceTemplateVirtualLinksInVlMode—**

Return Value

The operation timestamp

Exceptions

The following exceptions may be thrown by this method:

- ElementManagementException
- ApplyException

See Also

PolicyAPI.DC_UPDATE_METHOD_RPC , *PolicyAPI.DC_DEFAULT_RPC_PORT*

retrieveServiceConfiguration

Syntax

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

Description

Retrieves the service configuration loaded in the specified SCE.

Parameters

- **connectionApi—**The ConnectionApi that keeps a handle to the connection to the SCE

Return Value

The service configuration in the SCE platform

Exceptions

The following exceptions may be thrown by this method:

- IOException
- ElementManagementException
- ApplyException

updateValuesIni

Syntax

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

Description

Updates the CM with data from the SCE platform's service configuration.

Parameters

- **cmAddress—**The address of the CM to update
- **sceAddress—**The address of the SCE platform that enforces the given service configuration
- **servConf—**The service configuration

Exceptions

The following exception may be thrown by this method:

- ApplyException

updateValuesIni

Syntax

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

Description

Updates the CM with data from the SCE platform's service configuration.

Parameters

- **cmAddress—**The address of the CM to update
- **sceAddress—**The address of the SCE platform that enforces the given service configuration
- **servConf—**The service configuration
- **cmUpdateMethod**
- **rpcPort**

Exceptions

The following exception may be thrown by this method:

- ApplyException

validateServiceConfiguration

Syntax

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

Description

Validates a service configuration

Parameters

- servConf—The service configuration to validate

Return Value

A vector with warning messages regarding rules that might have undesirable results

See Also

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

importServConf

Syntax

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

Description

Loads the service configuration from a PQB file.

Parameters

- **pqbfile—**The PQB file to load

Return Value

The resulting service configuration

Exceptions

The following exceptions may be thrown by this method:

- ImportExportException
- IOException

exportServConf

Syntax

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

Description

Saves a service configuration to a PQB file

Parameters

- **servConf—**The service configuration to save
- **pqbfile—**The PQB file into which to save the service configuration

Exceptions

The following exceptions may be thrown by this method:

- FileNotFoundException
- ImportExportException

importDefaultServConf

Syntax

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

Description

Loads the default service configuration

Return Value

The default service configuration

Exceptions

The following exception may be thrown by this method:

- ImportExportException

Class ServiceConfig

Class ServiceConfig is the whole set of lists, protocols, services, and packages that an ISP has defined. The service configuration is applied to the ServiceConfig Domain's SCE platforms and sets their application parameters to regulate subscribers' flows.

Class ServiceConfig Methods

Class ServiceConfig methods are explained in the following sections.

- setProtocolRedirectString, page 3-29
- setTimeFrameName, page 3-29
- setTimeFrameName, page 3-30
- setTimeFrameNames, page 3-30

getCalendarList

Syntax

```
public CalendarArray getCalendarList()
```

Description

Gets the calendar list

Return Value

The list of calendars in the service configuration

getClassificationCfg

Syntax

```
public ClassificationConfiguration getClassificationCfg()
```

Description

Gets the classification configuration

Return Value

The classification-related configuration in the domain

getDynamicSignatureScript

Syntax

```
public DynamicSignaturesScript getDynamicSignatureScript()
```

Description

Gets the dynamic-signature configuration

Return Value

The dynamic-signature script

getPackageList

Syntax

```
public PackageArray getPackageList()
```

Description

Gets this service configuration's package list.

Return Value

This service configuration's package list

getPolicySettings

Syntax

```
public PolicySettings getPolicySettings()
```

Description

Gets this service configuration's settings. These settings are general system settings for the SCE platforms in this service configuration's domain.

Return Value

This service configuration's settings

getProtocolList

Syntax

```
public ProtocolList getProtocolList()
```

Description

Gets this service configuration's protocol list. The protocols in this list are referred to by the service configuration's services.

Return Value

This service configuration's protocols list

getProtocolRedirectIndexNameArray

Syntax

```
public ProtocolRedirectIndexNameArray getProtocolRedirectIndexNameArray()
```

Description

Gets the list of the protocols' redirect index names.

Return Value

The list of the protocols' redirect index names

getProtocolRedirectString

Syntax

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

Description

Gets the redirect address for the protocol in a certain index in its redirect String array.

Redirection is part of the protocol's specification, and exists for only a few predefined protocols.

Parameters

- **protocolName—**The queried protocol
- **redirectIndex—**The index in the redirect String array

Return Value

The redirect address

Exceptions

The following exception may be thrown by this method:

- ItemNotFoundException—If there is no such predefined protocol in this service configuration's ProtocolArray, or if the redirect index is out of bound

getRealTimeFrameName

Syntax

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

Description

Gets the predefined API name for a certain TimeFrame.

Parameters

- **name—**This service configuration's alias for the TimeFrame

Return Value

The TimeFrame's API name

Exceptions

The following exception may be thrown by this method:

- ItemNotFoundException—If there is no such alias in this service configuration

getServiceList

Syntax

```
public ServiceArray getServiceList()
```

Description

Gets this service configuration's service list.

Return Value

This service configuration's service list

getSubNotifications

Syntax

```
public SubNotificationArray getSubNotifications()
```

Description

Gets the subscriber notifications as configured in the service configuration

Return Value

The subscriber notifications

getTimeFrameNames

Syntax

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

Description

Gets the time-frame names this service configuration has assigned to the different TimeFrames.

These aliases allow time frames to have meaningful names.

The returned String array keeps in index X the alias of the TimeFrame index X.

Return Value

The String array of this service configuration's time-frame names

getProtocolRedirectString

Syntax

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

Description

Gets the default redirect address for the protocol.

Redirection is part of the protocol's specification, and exists for only a few predefined protocols.

Parameters

- **protocolName—**The queried protocol

Return Value

The redirect address

Exceptions

The following exception may be thrown by this method:

- ItemNotFoundException—If there is no such predefined protocol in this service configuration's ProtocolArray

getZoneList

Syntax

```
public ZoneList getZoneList()
```

Description

Gets this service configuration's IP, IP ranges, and host lists array. These lists are referred to by this service configuration's services.

Return Value

This service configuration's array of lists

isProtocolRedirectable

Syntax

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

Description

Checks if a specified protocol supports redirecting.

Redirection is part of the protocol's specification, and exists for only a few predefined protocols.

Parameters

- **protocolName—**The queried protocol's name

Return Value

`true` if the protocol support redirection, `false` otherwise

Exceptions

The following exception may be thrown by this method:

- ItemNotFoundException—If there is no such predefined protocol in this service configuration's ProtocolArray

setProtocolRedirectString

Syntax

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

Description

Sets the address to which to redirect a certain flow. The redirection occurs when a rule has an *ACCESS_BLOCK_AND_REDIRECT* access mode for a certain service that uses the desired protocol. The setting is done to a certain String in the redirect String array. The rule chooses which redirect String to use from the redirect String array.

Parameters

- **protocolName—**The protocol whose activity will be redirected
- **redirectIndex—**The redirect String index in the protocol's redirect String array
- **value—**The protocol's redirect address

Exceptions

The following exceptions may be thrown by this method:

- ItemNotFoundException—If there is no such predefined protocol in this service configuration's ProtocolArray, or if the redirect index is out of bound
- MalformedURLException—If the String is an illegal URL name

setProtocolRedirectString

Syntax

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

Description

Sets the default address to which to redirect a certain flow. The redirection occurs when a rule has an *ACCESS_BLOCK_AND_REDIRECT* access mode for a certain service that uses the desired protocol.

Parameters

- **protocolName—**The desired protocol whose activity will be redirected
- **value—**The protocol's redirect address

Exceptions

The following exceptions may be thrown by this method:

- ItemNotFoundException—If there is no such predefined protocol in this service configuration's ProtocolArray
- MalformedURLException—If the String is an illegal URL name

setTimeFrameName

Syntax

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

Description

Sets an alias for the TimeFrame that has a given index.

The alias allows the time frame to have a meaningful name.

Parameters

- **index—**The index of the TimeFrame to which the alias is given
- **newName—**The alias

Exceptions

The following exceptions may be thrown by this method:

- ItemNotFoundException—If there is no such TimeFrame
- DuplicateItemException

setTimeFrameName

Syntax

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

Description

Sets an alias for the specified TimeFrame.

The alias allows the time frame to have a meaningful name.

Parameters

- frame—The TimeFrame to which the alias is given
- newName—The alias

Exceptions

The following exceptions may be thrown by this method:

- ItemNotFoundException—If there is no such TimeFrame
- DuplicateItemException

setTimeFrameNames

Syntax

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

Description

Sets the names of all time frames.

Parameters

- **newNames—**The names to set

Service Configuration API Programming Guidelines

Connections to the SCE Platform

Make sure that connections you create using any of the login() methods are properly closed using logout() (see logout, page 3-4).

Service Configuration API Code Examples

This section gives several code examples of Service Configuration API usage.

Applying a Service Configuration

The following example applies a new service configuration to the SCE platform that is specified in the command line.

```
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.
 * &lt;p&gt;
 * usage: java examples.SimpleApplyPqb &lt;sce-address&gt; &lt;password&gt;
 * &lt;pqb-filename&gt;
 */
public class SimpleApplyPqb {

   public static void main(String[] args) {

      if (args.length != 3) {
         System.err.println("usage: java examples.SimpleApplyPqb "
                            + "&lt;sce-address&gt; &lt;password&gt; &lt;pqb-filename&gt;");
         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;
      }
   }
```

Updating Zones Automatically

The following example updates an SCE with zone IP addresses. The zone IP addresses are specified in a CSV file.

```
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.
 * &lt;p&gt;
 * usage: java examples.UpdateZoneFromCsv &lt;sce-address&gt; &lt;password&gt;
 * &lt;zone-csv-file&gt; &lt;zone-name&gt;
 *
 */
public class UpdateZoneFromCsv {

   public static void main(String[] args) {

      if (args.length != 4) {
         System.err.println("usage: java examples.UpdateZoneFromCsv"
                           + " &lt;sce-address&gt; &lt;password&gt;"
                           + " &lt;zone-csv-file&gt; &lt;zone-name&gt;");
         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;
   }

}
```

Listing Names of Services and Packages

The following example prints the names of the services and packages that are in a service configuration.

```
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 &gt; 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());

      }
   }
}
```