‹ Back to Knox SDK API References page

SAMSUNG Knox SDK API reference

Map API level to SDK version ☐ Filter by API level: 1

public class

GenericVpnPolicy

extends Object

java.lang.Object
↳com.samsung.android.knox.net.vpn.GenericVpnPolicy

Class Overview

The class provides APIs to configure SSL/IPSEC VPN profiles on the device.

Description:

The below steps need to be followed before calling any API's in `GenericVpnPolicy` class.

1. The administrator has to get the instance of `GenericVpnPolicy` in `EnterpriseKnoxManager` class by passing the VPN vendor's package name as parameter.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
```

Once the above code is executed, the framework will try to bind to the VPN vendor's application and once the bind is successful, a broadcast message will be sent to the administrator.

2. Now, the administrator has to listen for the intent action ACTION_BIND_RESULT.

3. The intent received will have the following info:
a) EXTRA_BIND_VENDOR will have the VPN vendor's package name as the value.
b) EXTRA_BIND_CID will have the container ID in which the VPN vendor's application is installed.
c) EXTRA_BIND_STATUS will return true or false, specifying whether the bind to the VPN vendor's application was successful or not.

```
public class VPNBindReceiver extends BroadcastReceiver {

    public static final String ACTION_BIND_RESULT = "com.samsung.android.knox.intent.action.VPN_BIND_RESULT";
    public static final String EXTRA_BIND_VENDOR = "com.samsung.android.knox.intent.extra.VPN_BIND_VENDOR";
    public static final String EXTRA_BIND_CID = "com.samsung.android.knox.intent.extra.VPN_BIND_CID";
    public static final String EXTRA_BIND_STATUS = "com.samsung.android.knox.intent.extra.VPN_BIND_STATUS";

    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equalsIgnoreCase(ACTION_BIND_RESULT)) {
            String vendorName = intent.getExtra(EXTRA_BIND_VENDOR);
            int containerId = intent.getExtra(EXTRA_BIND_CID);
            boolean status = intent.getExtra(EXTRA_BIND_STATUS);
        }
    }
}
```

4. Now, once the received value of vpn_bind_status is true, then the API's present in GenericVpnPolicy class needs be called with reference to the GenericVpnPolicy object (gm) which was instantiated in step 1. For example:

```
String vpnConnection = gm.getVpnProfile("profileName");
int success = gm.createVpnProfile("profileInfo");
```

Note:
i. The GenericVpnPolicy object (gm) has to be instantiated only once and all the API's need to be called with reference to the same object.
ii. If the GenericVpnPolicy object is instantiated more than once (for example, for every API), then an intent will be sent from the framework for every call, which can be avoided.
iii. All API's in the GenericVpnPolicy class should be called only after receiving the intent and the vpn_bind_status extra value is true, otherwise the call to all API's will fail.

5. The API flow for creating and starting a VPN connection will be createVpnProfile(String) -> setVpnModeOfOperation(String, int) (if applicable) -> addPackagesToVpn(String[], String) -> activateVpnProfile(String, boolean) API.

6. The API flow for removing a VPN profile will be activateVpnProfile(String, boolean) (De-activate it) -> removeVpnProfile(String) API.

7. The API flow for creating and starting a VPN connection which has all the packages added to VPN is -> createVpnProfile(String) -> setVpnModeOfOperation(String, int) (if applicable) -> addAllPackagesToVpn(String) -> activateVpnProfile(String, boolean) API.

8. The API flow for removing a VPN profile which has all the packages added to VPN follows the same approach as described in item 6.

Since
API level 9
KNOX 1.1.0

Summary

Constants

String	ACTION_BIND_RESULT	Intent action used to notify the administrator about bind result with the VPN client.
String	EXTRA_BIND_CID	Used as an intent extra field with `ACTION_BIND_RESULT`.
String	EXTRA_BIND_STATUS	Used as an intent extra field with `ACTION_BIND_RESULT`.
String	EXTRA_BIND_VENDOR	Used as an intent extra field with `ACTION_BIND_RESULT`.
String	KEY_TETHER_CA_CERTIFICATE	Key in Bundle which is used to pass the CA cert for usb tethering authentication in byte array format used in `allowUsbTetheringOverVpn(String, boolean, Bundle)`
String	KEY_TETHER_USER_CERTIFICATE	Key in Bundle which is used to pass the user cert for usb tethering authentication in byte array format used in `allowUsbTetheringOverVpn(String, boolean, Bundle)`
String	KEY_TETHER_USER_CERT_PASSWORD	Key in Bundle which is used to pass the user cert password for usb tethering authentication used in `allowUsbTetheringOverVpn(String, boolean, Bundle)`

Public Methods

int	activateVpnProfile(String profileName, boolean enable) API to activate or de-activate a VPN connection.
int	addAllContainerPackagesToVpn(int containerId, String profileName) The API is used to add all the packages present inside the container to VPN.
int	addAllPackagesToVpn(String profileName) The API is used to add all the packages present under the user to VPN.
int	addContainerPackagesToVpn(int containerId, String[] packageList, String profileName) The API is used to add the list of containerized packages to VPN.
int	addPackagesToVpn(String[] packageList, String profileName) The API is used to add the list of packages to a VPN.
int	allowUsbTetheringOverVpn(String profileName, boolean allow, Bundle authInfo) The API is used to allow/disallow usb tethering which allows/disallows the tethered traffic from usb accessory like laptop to go through VPN.
int	createVpnProfile(String profileInfo) API used to create a new VPN connection.
String[]	getAllContainerPackagesInVpnProfile(int containerId, String profileName) The API will retrieve the list of containerized application belonging to the profile.
String[]	getAllPackagesInVpnProfile(String profileName) The API will return the list of packages which was added to the VPN for the given profile.
List<String>	getAllVpnProfiles() API to get the list of all Knox VPN connections added by the administrator.
CertificateInfo	getCACertificate(String profileName) The API returns the CA Certificate for the specified profile.
String	getErrorString(String profileName) The API is used to get the error state of the VPN profile.
int	getState(String profileName) The API is used to get the current state of the VPN profile.
CertificateInfo	getUserCertificate(String profileName) The API returns the User certificate for the specified profile.
int	getVpnModeOfOperation(String profileName) API to get the current mode of operation for the given profile.
String	getVpnProfile(String profileName) API to retrieve the VPN connection details belonging to a particular profile.
int	isUsbTetheringOverVpnEnabled(String profileName) The API is used query if usb tethering is enabled or not for the profile.
int	removeAllContainerPackagesFromVpn(int containerId, String profileName) The API will remove all the containerized applications belonging to the profile from VPN.

int	removeAllPackagesFromVpn(String profileName) The API is used to remove all the packages present under the user from VPN.
int	removeContainerPackagesFromVpn(int containerId, String[] packageList, String profileName) The API will remove the list of packages from the profile inside the container.
int	removePackagesFromVpn(String[] packageList, String profileName) The API will remove the list of packages added from the VPN.
int	removeVpnProfile(String profileName) API to remove an enterprise VPN profile.
boolean	setAutoRetryOnConnectionError(String profileName, boolean enable) API to set whether to enable auto-reconnect feature or not for the given profile.
boolean	setCACertificate(String profileName, byte[] certificateBlob) The API allows the administrator to configure the CA certificate for a VPN profile.
boolean	setServerCertValidationUserAcceptanceCriteria(String profileName, boolean enableValidation, List<Integer> condition, int frequency) API to enable list of SRG requirements for a given profile.
boolean	setUserCertificate(String profileName, byte[] pkcs12Blob, String password) The API allows administrator to configure the User certificate for a VPN profile.
int	setVpnModeOfOperation(String profileName, int vpnMode) API to set VPN mode of operation in either FIPS or non-FIPS mode.

Inherited Methods

- From class java.lang.Object

Constants

public static final String ACTION_BIND_RESULT

Since: API level 9

Intent action used to notify the administrator about bind result with the VPN client. It will have EXTRA_BIND_VENDOR, EXTRA_BIND_CID, and EXTRA_BIND_STATUS as extra information. Receiver must hold "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" to receive this broadcast.

Since

API level 9
KNOX 1.1.0

Constant Value: "com.samsung.android.knox.intent.action.VPN_BIND_RESULT"

public static final String EXTRA_BIND_CID

Since: API level 9

Used as an intent extra field with ACTION_BIND_RESULT. Contains the container ID in which the VPN vendor's application is installed.

Since

API level 9
KNOX 1.1.0

Constant Value: "com.samsung.android.knox.intent.extra.VPN_BIND_CID"

public static final String EXTRA_BIND_STATUS

Since: API level 9

Used as an intent extra field with ACTION_BIND_RESULT. Contains `true` if the bind to the VPN vendor's application was successful and `false` otherwise.

Since

API level 9
KNOX 1.1.0

Constant Value: "com.samsung.android.knox.intent.extra.VPN_BIND_STATUS"

public static final String EXTRA_BIND_VENDOR

Since: API level 9

Used as an intent extra field with `ACTION_BIND_RESULT`. Contains the VPN vendor's package name.

Since

API level 9
KNOX 1.1.0

Constant Value: "com.samsung.android.knox.intent.extra.VPN_BIND_VENDOR"

public static final String KEY_TETHER_CA_CERTIFICATE

Since: API level 32

Key in Bundle which is used to pass the CA cert for usb tethering authentication in byte array format used in `allowUsbTetheringOverVpn(String, boolean, Bundle)`

Since

API level 32
KNOX 3.6

Constant Value: "key-tether-ca-certificate"

public static final String KEY_TETHER_USER_CERTIFICATE

Since: API level 32

Key in Bundle which is used to pass the user cert for usb tethering authentication in byte array format used in `allowUsbTetheringOverVpn(String, boolean, Bundle)`

Since

API level 32
KNOX 3.6

Constant Value: "key-tether-user-certificate"

public static final String KEY_TETHER_USER_CERT_PASSWORD

Since: API level 32

Key in Bundle which is used to pass the user cert password for usb tethering authentication used in `allowUsbTetheringOverVpn(String, boolean, Bundle)`

Since

API level 32
KNOX 3.6

Constant Value: "key-tether-user-cert-password"

Public Methods

public int activateVpnProfile (String profileName, boolean enable)

Since: API level 9

API to activate or de-activate a VPN connection.

Additional info:
The API will be used only when `vpn_route_type` is set to 1 (per-app-vpn) while creating a VPN connection.

Once the profile is activated, then the VPN connection will be started. The following two scenarios should be considered:

1. If the API is called before `addPackagesToVpn(String[], String)`, VPN will not be started unless the packages are added.
2. If the API is called after `addPackagesToVpn(String[], String)`, then the VPN will be started and the added package will go through VPN.

Once the profile is deactivated, then the VPN connection for the profile will be stopped.
By default, the profile will be in deactivated state.

Parameters

profileName	Name of the profile to be activated or deactivated.
enable	Specifies whether the profile is to be activated (`true`) or deactivated (`false`).

Returns

0: If the requested operation is successful.
1: If the requested operation is not successful.
-1: Error while activating or deactivating the profile.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    int profileStatus = gm.activateVpnProfile("profileName", true);
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

---

public int **addAllContainerPackagesToVpn** (int containerId, String profileName) Since: API level 9

The API is used to add all the packages present inside the container to VPN.

Additional info:
VPN profile should be created by calling `createVpnProfile(String)` before calling this API.
This API will also cover future installed applications in same user space.

Parameters

containerId	Container Id.
profileName	Name of the profile.

Returns

0: If all the packages are added successfully.
1: If some packages are not added successfully: in this scenario, the administrator has to call `getAllContainerPackagesInVpnProfile(int, String)` API to get the list of packages which were added successfully.
2: None of the packages were added successfully.
-1: Error while adding packages.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions. Only administrator who created container can enforce policy, if the calling administrator is not an owner of container the API throws `SecurityException`.
IllegalArgumentException	If container does not exist or container creation/removal is in progress.

Usage

Calling this API will add the entire set of containerized applications within the specified container to an auto start VPN list, that will enable the system to start VPN automatically for that container.

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.addAllContainerPackagesToVpn(containerID, "profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
} catch (IllegalArgumentException e) {
    Log.e(TAG, "IllegalArgumentException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

public int **addAllPackagesToVpn** (String profileName) Since: API level 11

The API is used to add all the packages present under the user to VPN.

Additional info:
VPN profile should be created by calling `createVpnProfile(String)` before calling this API.
This API will also cover future installed applications in same user space.

Parameters

profileName	Name of the profile.

Returns

0: If all the packages are added successfully.
1: If some packages are not added successfully: in this scenario, the administrator has to call `getAllPackagesInVpnProfile(String)` API to get the list of packages which were added successfully.
2: None of the packages were added successfully.
-1: Error while adding packages.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.addAllPackagesToVpn("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 11
KNOX 2.0

Multiuser Environment

User Scope

public int **addContainerPackagesToVpn** (int containerId, String[] packageList, String profileName) Since: API level 9

The API is used to add the list of containerized packages to VPN. The behavior when applications with shared UID inside container are added needs to be noted. When an application that shares UID with another is added, all applications that have a common UID will have their traffic routed through the tunnel of a single activated VPN profile. An error will be returned when applications sharing a UID is added to different profiles.

Additional info:
The API should be used after creating a VPN connection.

Parameters

containerId	Container Id.
packageList	List of containerized packages to be added to VPN list.
profileName	Name of the profile.

Returns

0: If all the packages are added successfully.
1: If some packages are not added successfully: in this scenario, the administrator has to call `getAllContainerPackagesInVpnProfile(int, String)` API to get the list of packages which were added successfully.
2: None of the packages were added successfully.
-1: Error while adding packages.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions. Only administrator who created container can enforce policy, if the calling administrator is not an owner of container the API throws `SecurityException`.
IllegalArgumentException	If container does not exist or container creation/removal is in progress.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.addContainerPackageToVpn(containerID, packageList, "profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
} catch (IllegalArgumentException e) {
    Log.e(TAG, "IllegalArgumentException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

public int addPackagesToVpn (String[] packageList, String profileName) Since: API level 9

The API is used to add the list of packages to a VPN. The behavior when applications with shared UID are added needs to be noted. When an application that shares UID with another is added, all applications that have a common UID will have their traffic routed through the tunnel of a single activated VPN profile. An error will be returned when applications sharing a UID is added to different profiles.

Additional info:
This API should be called after creating a VPN connection.

Parameters

packageList	List of packages to be added to the VPN.
profileName	Name of the profile.

Returns

0: If all the packages are added successfully.
1: If some packages are not added successfully: in this scenario, the administrator has to call `getAllPackagesInVpnProfile(String)` API to get the list of packages which were added successfully.
2: None of the packages were added successfully.
-1: Error while adding packages.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.addPackagesToVpn(packageList, "profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public int allowUsbTetheringOverVpn (String profileName, boolean allow, Bundle authInfo) Since: API level 32

The API is used to allow/disallow usb tethering which allows/disallows the tethered traffic from usb accessory like laptop to go through VPN.

Additional Info:
VPN profile should be created by calling `createVpnProfile(String)` before calling this API.
After EMM calls the below API, It requires the end-user to manually turn-on the USB Tethering feature from the Settings Menu; Once turned on, the tethered traffic will go through VPN;

If the VPN goes down for an activated vpn profile, the tethered network traffic originating from the usb accessory like laptop will be blocked;
The tethered network traffic between tethered devices like laptop/Desktop and mobile device will be blocked until the mutual authentication is successful;

Parameters

profileName	Name of the profile.
allow	allow (true) or disallow (false) usb tethering.
authInfo	Bundle containing authentication info to authenticate the tethered device;

Returns

0: No Error
ERROR_NULL_PARAMETER
ERROR_PROFILE_NAME_NOT_EXIST_DEVICE
ERROR_PROFILE_NAME_EXISTS_DIFFERENT_ADMIN
ERROR_INVALID_USB_TETHERING_CONFIGURATION

Throws

SecurityException	If caller does not have required permissions.

Usage

```
   Allow Usb Tethering:


EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    Bundle bundle = new Bundle();
    bundle.putByteArray(KEY_TETHER_CA_CERTIFICATE,cACertArray);
    bundle.putByteArray(KEY_TETHER_USER_CERTIFICATE,userCertArray);
    bundle.putString(KEY_TETHER_USER_CERT_PASSWORD,"psswd");


    boolean success = gm.allowUsbTetheringOverVpn("profileName",true,bundle);
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}

   Disallow Usb Tethering:


EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.allowUsbTetheringOverVpn("profileName",false,null);
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

KNOX 3.6

Multiuser Environment

User Scope

---

public int **createVpnProfile** (String profileInfo) Since: API level 9

API used to create a new VPN connection.

Additional info:
This is the first step for creating a VPN connection.

For JSON files creation please follow the instructions in Developer Guide.

Parameters

profileInfo	JSON object in `String` format which contains the profile information.

Returns

0: If the VPN profile is added successfully.
1: If the VPN profile is not added successfully.
-1: Error occurred while creating a VPN profile.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    int success = gm.createVpnProfile("profileInfo");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public String[] getAllContainerPackagesInVpnProfile (int containerId, String profileName)

Since: API level 9

The API will retrieve the list of containerized application belonging to the profile. This API will also return the list of packages installed inside container that share UID with the applications that are added to the VPN profile.

Parameters

containerId	Container Id.
profileName	Name of the profile.

Returns

The list of packages names as `String[]` or `null` if no packages found.

Throws

SecurityException	If caller does not have required permissions. Only administrator who created container can enforce policy, if the calling administrator is not an owner of container the API throws `SecurityException`.
IllegalArgumentException	If container does not exist or container creation/removal is in progress.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    String[] packages = gm.getAllContainerPackagesInVpnProfile(containerID, "profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
} catch (IllegalArgumentException e) {
    Log.e(TAG, "IllegalArgumentException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

public String[] getAllPackagesInVpnProfile (String profileName)

Since: API level 9

The API will return the list of packages which was added to the VPN for the given profile. This API will also return the list of installed packages that share UID with the applications that are added to the VPN profile.

Additional info:
The API should be used after creating a VPN connection.

Parameters

profileName	Name of the profile whose packages we want to get.

Returns

The list of packages which was added to VPN for the given profile or `null` if the profile does not exist.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    String[] packages = gm.getAllPackagesInVpnProfile("testprofile");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public List<String> getAllVpnProfiles ()

Since: API level 9

API to get the list of all Knox VPN connections added by the administrator.

NOTE: Starting from API Level 30, this method can be called for `com.android.settings` vendor to retrieve all VPN profiles created by the end-user in Settings.

Returns

The list of VPN connections added by the administrator. It returns a list of JSON objects in `String` format.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    List<String> profileLists = gm.getAllVpnProfiles();
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public CertificateInfo getCACertificate (String profileName)

Since: API level 9

The API returns the CA Certificate for the specified profile.

Parameters

profileName Name of the profile.

Returns

`CertificateInfo` object for the CA certificate. Return value is `null`, if no VPN profile with the given `profileName` is found, or if a CA certificate is not found for this profile.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    CertificateInfo listCerts = gm.getCACertificate("profileName");
} catch(SecurityException e) {
    Log.e(TAG,"SecurityException" + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public String getErrorString (String profileName)

Since: API level 9

The API is used to get the error state of the VPN profile.

Parameters

profileName Name of the profile.

Returns

Current error code associated with the given profile or `null` if the profile does not exist. If the current code is `ERROR_VPN_RECREATE_PROFILE_FAIL` (added in `VpnErrorValues`) immediately the method returns a Json with the profile data:

```
{
    "errorType":307,
    "profileName":"nameOfProfile",
    "vendorName":"vendorPackage",
    "userID":"10",
    "packageList":[
        "package1",
        "package2"
     ]
}
```

Where: userID = Id of the android user; packageList = List of applications that shall communicate via VPN profile connection; vendorPackage = VPN client Identification.

Note: From Knox 3.8 on, this API will return a Json following the schema above containing the error (`errorType`) ERROR_VPN_RECREATE_PROFILE_FAIL added in `VpnErrorValues` class. The getErrorString API will be able to return this Json only at the first time that the EMM calls it. In other words, it means that all the information related to the error will be deleted immediately after the first API call. For the case that the getErrorString API is not called after the VPN profile recreation failure and the EMM application tries to create a new profile with the same name, the framework will delete all the information persisted on the database about the previously recreation failure.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    String stateResponse = gm.getErrorString("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException" + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public int getState (String profileName) Since: API level 9

The API is used to get the current state of the VPN profile.

Additional info:
The current status of the VPN profile can be broadly classified into either Activate or in De-activate state.

- If the profile is in Activated state then the return value for the profile can be IDLE (1), CONNECTING (2), DISCONNECTING (3), CONNECTED (4) or FAILED (5).
- If the profile is in De-Activated state then the return value for the profile will only be DEACTIVATED (0).

Parameters

profileName Name of the profile.

Returns

DEACTIVATED = 0;
IDLE = 1;
CONNECTING = 2;
DISCONNECTING = 3;
CONNECTED = 4;
FAILED = 5;
Error in retrieving the info = -1;

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    int stateResponse = gm.getState("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException" + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public CertificateInfo getUserCertificate (String profileName) Since: API level 9

The API returns the User certificate for the specified profile.

Parameters

profileName Name of the profile.

Returns

`CertificateInfo` object for the User certificate. Return value is `null`, if no VPN profile with the given `profileName` is found, or if an User certificate is not found for this profile.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    CertificateInfo listCerts = gm.getUserCertificate("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException" + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public int getVpnModeOfOperation (String profileName)

Since: API level 9

API to get the current mode of operation for the given profile.

Parameters

profileName Name of the profile.

Returns

0: The VPN profile is not operating in FIPS mode.
1: The VPN profile is operating in FIPS mode.
-1: Unexpected error.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    int success = gm.getVpnModeOfOperation(profileName);
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

See Also

`setVpnModeOfOperation(String, int)`

public String getVpnProfile (String profileName)

Since: API level 9

API to retrieve the VPN connection details belonging to a particular profile.

Parameters

profileName Name of the connection to be retrieved.

Returns

JSON object in `String` format which contains the connection info.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    String vpnConnection = gm.getVpnProfile("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public int isUsbTetheringOverVpnEnabled (String profileName)

Since: API level 32

The API is used query if usb tethering is enabled or not for the profile.

Parameters

profileName Name of the profile.

Returns

0: usb tethering is disabled for the profile
1: usb tethering is enabled for the profile
ERROR_NULL_PARAMETER
ERROR_PROFILE_NAME_NOT_EXIST_DEVICE
ERROR_PROFILE_NAME_EXISTS_DIFFERENT_ADMIN
ERROR_INVALID_USB_TETHERING_CONFIGURATION

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.isUsbTetheringOverVpnEnabled("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

KNOX 3.6

Multiuser Environment

User Scope

public int removeAllContainerPackagesFromVpn (int containerId, String profileName)

Since: API level 9

The API will remove all the containerized applications belonging to the profile from VPN.

Additional info:
The below API can be used only when the administrator has added the applications to VPN profile using addAllContainerPackagesToVpn(int, String).

Parameters

containerId Container Id.
profileName All the container VPN Packages added to the given profile will be removed.

Returns

0: If all the packages are removed successfully
1: If some packages are not removed successfully: in this scenario, the administrator has to call getAllContainerPackagesInVpnProfile(int, String) API to get the list of packages which were removed successfully.
2: None of the packages were removed successfully.
-1: Error while removing packages.

Note: From Knox 2.4 on, for other error codes refer in VpnErrorValues.

Throws

SecurityException	If caller does not have required permissions. Only administrator who created container can enforce policy, if the calling administrator is not an owner of container the API throws SecurityException.
IllegalArgumentException	If container does not exist or container creation/removal is in progress.

Usage

Calling this API will remove the entire set of containerized applications within the specified container from an auto start VPN list.

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.removeAllContainerPackagesFromVpn(containerID, "profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
} catch (IllegalArgumentException e) {
    Log.e(TAG, "IllegalArgumentException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

public int removeAllPackagesFromVpn (String profileName) Since: API level 11

The API is used to remove all the packages present under the user from VPN.

Additional info:
VPN profile should be created by calling `createVpnProfile(String)` before calling this API.
Once all the packages are removed, the VPN connection will be stopped for the given profile.
The below API can be used only when the administrator has added the applications to VPN profile using `addAllPackagesToVpn(String)`.

Parameters

profileName Name of the profile.

Returns

0: If all the packages are removed successfully
1: If some packages are not removed successfully: in this scenario, the administrator has to call `getAllPackagesInVpnProfile(String)` API to get the list of packages which were removed successfully.
2: None of the packages were removed successfully.
-1: Error while removing packages.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.removeAllPackagesFromVpn("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 11
KNOX 2.0

Multiuser Environment

User Scope

public int removeContainerPackagesFromVpn (int containerId, String[] packageList, String profileName) Since: API level 9

The API will remove the list of packages from the profile inside the container. The behavior when applications inside the container with shared UID are removed needs to be noted. When an application that shares UID with another is removed, all applications that have a common UID will no longer have their network traffic tunneled.

Additional info:
The below API can be used only when the administrator has added the applications to VPN profile using `addContainerPackagesToVpn(int, String[], String)`.
or `addAllContainerPackagesToVpn(int, String)`.

Parameters

containerId	Container Id.
packageList	List of the Container packages to be removed from VPN list for the given profile.
profileName	Name of the profile.

Returns

0: If all the packages are removed successfully
1: If some packages are not removed successfully: in this scenario, the administrator has to call `getAllContainerPackagesInVpnProfile(int, String)` API to get the list of packages which were removed successfully.
2: None of the packages were removed successfully.
-1: Error while removing packages.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions. Only administrator who created container can enforce policy, if the calling administrator is not an owner of container the API throws `SecurityException`.
IllegalArgumentException	If container does not exist or container creation/removal is in progress.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.removeContainerPackageFromVpn(containerID, packageList);
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
} catch (IllegalArgumentException e) {
    Log.e(TAG, "IllegalArgumentException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

public int **removePackagesFromVpn** (String[] packageList, String profileName) Since: API level 9

The API will remove the list of packages added from the VPN. The behavior when applications with shared UID are removed needs to be noted. When an application that shares UID with another is removed, all applications that have a common UID will no longer have their network traffic tunneled.

Additional info:
The VPN connection for the profile will be stopped if all the packages belonging to the profile is removed.
The below API can be used only when the administrator has added the applications to VPN profile using `addPackagesToVpn(String[], String)` or `addAllPackagesToVpn(String)`.

Parameters

packageList	List of the packages to be removed from VPN list for the given profile.
profileName	Name of the profile.

Returns

0: If all the packages are removed successfully.
1: If some packages are not removed successfully: in this scenario, the administrator has to call `getAllPackagesInVpnProfile(String)` API to get the list of packages which were removed successfully.
2: None of the packages were removed successfully.
-1: Error while removing packages.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.removePackageFromVpn("com.android.browser");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

---

public int **removeVpnProfile** (String profileName) Since: API level 9

API to remove an enterprise VPN profile.

Additional info:
If the profile is in connected state, the profile needs to be deactivated by calling the `activateVpnProfile(String, boolean)` API and then, once the profile is in deactivated state, the profile can be removed.

NOTE: Starting from API level 30, this method can be called for `com.android.settings` vendor to delete a VPN profile created by the end-user in Settings. As an Android Settings cannot be activated/deactivated, the state is not considered when this method is called.

Parameters

profileName Name of the connection to be removed.

Returns

0: If the VPN profile is removed successfully.
1: If the VPN profile is not removed successfully.
-1: Error occurred while removing the VPN profile.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    int removeSuccess = gm.removeVpnProfile("profileName");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

---

public boolean **setAutoRetryOnConnectionError** (String profileName, boolean enable) Since: API level 9

API to set whether to enable auto-reconnect feature or not for the given profile.

Parameters

profileName The profile name for which the auto-reconnect feature has to be set or not.
enable `True` to enable auto-reconnect feature, `false` otherwise.

Returns

`True` if the requested mode has been successfully set, `false` otherwise.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.setAutoRetryOnConnectionError("profileName", true);
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public boolean setCACertificate (String profileName, byte[] certificateBlob)

Since: API level 9

The API allows the administrator to configure the CA certificate for a VPN profile.

Additional info:
The API should be used after creating a VPN connection and before calling `addPackagesToVpn(String[], String)` API.

Parameters

profileName	Name of the profile.
certificateBlob	Byte array of the CA certificate in DER/PEM format.

Returns

If the CA certificate is configured with given profile successfully, it returns `true`. If API fails to read certificate or fails to store the CA certificate for the specified profile, it returns `false`.

Throws

SecurityException	If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
String filePath = "/data/system/client1.der";
byte[] certificateBlob = getByteArray(filePath); // internal function to retrieve byte array from file.
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.setCACertificate("profileName", certificateBlob);
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException" + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public boolean setServerCertValidationUserAcceptanceCriteria (String profileName, boolean enableValidation, List<Integer> condition, int frequency)

Since: API level 9

API to enable list of SRG requirements for a given profile. More information is given in SRG requirement document.

Parameters

profileName	The profile name for which the SRG requirements has to be set or not.
enableValidation	Whether to enable or disabled SRG requirements for the given profile.
condition	Subject-Mismatch (0), Key-Usage-Violation (1), Revocation-Verification (2).
frequency	The frequency by which to notify the user: ignore-Always (0), ignore-Current-Session (1), ignore-Once (2).

Returns

`True` if the requested mode has been successfully set, `false` otherwise.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    List ruleList = new ArrayList();
    ruleList.add(1);
    boolean success = gm.setServerCertValidationUserAcceptanceCriteria("profileName", true, ruleList, 2);
} catch (SecurityException e) {
   Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public boolean setUserCertificate (String profileName, byte[] pkcs12Blob, String password)

Since: API level 9

The API allows administrator to configure the User certificate for a VPN profile.

Additional info:
The User certificate must be provided in PKCS12 format and, in order to decipher its content, the password is also necessary. This is a synchronous API.
The API should be used after creating a VPN connection and before calling `addPackagesToVpn(String[], String)` API.

Parameters

profileName	Name of the profile.
pkcs12Blob	Byte array of User certificate in PKCS12 format.
password	Password to decipher the content of PKCS12 blob.

Returns

`True` if User certificate is configured with given profile successfully. If the API fails to read PKCS12 blob OR fails to store the User certificate for the specified profile, it returns `false`.

Throws

SecurityException If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
String filePath = "/data/system/client1.p12";
byte[] pkcs12Blob = getByteArray(filePath); // internal function to retrieve byte array from file.
try {
    GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
    boolean success = gm.setUserCertificate("profileName", pkcs12Blob, "password");
} catch (SecurityException e) {
    Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

public int setVpnModeOfOperation (String profileName, int vpnMode)

Since: API level 9

API to set VPN mode of operation in either FIPS or non-FIPS mode.

Additional info:

The API is based on VPN vendor's support.
The Support for FIPS Mode falls under 2 categories.

1. If the vendor supports profile based FIPS Mode, then the FIPS/Non-FIPS Mode will be set based on the `profileName` parameter.
   - In this scenario, the API needs to be called after `createVpnProfile(String)` API.
2. If the vendor does not support FIPS Mode, then the FIPS/Non-FIPS mode will be applied across all profiles.
   - In this scenario, the API needs to be called after creating the first profile.

Parameters

profileName	Profile name for which the VPN mode needs to be set.
vpnMode	0: Non-FIPS Mode; 1: FIPS Mode;

Returns

0: If the requested mode was set successfully.
1: If the requested mode was not set successfully.
-1: Error occurred while setting the VPN mode of operation.

Note: From Knox 2.4 on, for other error codes refer in `VpnErrorValues`.

Throws

SecurityException	If caller does not have required permissions.

Usage

```
EnterpriseKnoxManager ekm = EnterpriseKnoxManager.getInstance(context);
try {
   GenericVpnPolicy gm = ekm.getGenericVpnPolicy("com.vpn.vendor.packageName", userId);
   int success = gm.setVpnModeOfOperation("name", 0);
} catch (SecurityException e) {
   Log.e(TAG, "SecurityException: " + e);
}
```

Permission

The use of this API requires the caller to have the "com.samsung.android.knox.permission.KNOX_VPN_GENERIC" permission with a protection level of signature.

Since

API level 9
KNOX 1.1.0

Multiuser Environment

User Scope

See Also

`getVpnModeOfOperation(String)`