Pulse Secure Android Client Administration Guide

Product Release	9.3.0
Published	November 2020
Document Version	1.0

Pulse Secure, LLC
2700 Zanker Road,
Suite 200 San Jose
CA 95134

www.pulsesecure.net

END USER LICENSE AGREEMENT

Revision History

The following table lists the revision history for this document

Document Version	Date	Description
1.0	October 2020	Initial Publication for Release 9.3.0

Contents

Pulse Mobile Client for Android

Pulse Mobile Client for Android Overview

Pulse Mobile Client can create an authenticated SSL session between a device running Google Android and Pulse Connect Secure. Pulse Mobile Client enables secure connectivity to Web-based applications and data based on identity, realm, and role. Pulse Mobile Client is available for download from the Android Market.

The *Pulse Secure Mobile Client Supported Platforms Guide*, available from the Pulse Secure website (www.pulsesecure.net), lists the mobile device OS versions supported by Pulse Mobile Client and the security features supported on each mobile device OS.

Pulse Secure has created an App VPN SDK that enables app developers to integrate Pulse Secure VPN connectivity within individual apps. Using an App VPN tunnel means that a mobile device user does not need to activate a Pulse Mobile Client connection and direct all traffic from the mobile device to the Pulse Secure gateway. Only traffic from the app goes through the tunnel. Configuration on the Pulse Secure server for App VPN is no different than the configuration for Pulse Mobile Client connections. For more information about Pulse Secure VPN SDK, see your Pulse Secure representative.

Note: The Google Android OS has limitations in its support for certificate-based authentication. If necessary, you can separate the private key from the certificate by using openssl commands before you install the certificate and the key on the Android device. The Pulse Secure Knowledge Center (https://kb.pulsesecure.net) includes an article, KB19692, that describes in detail how to create a certificate and key that enables successful certificate authentication for Pulse Mobile Client on Android.

Note: To ensure that users see consistent bookmarks in the Pulse Mobile Client UI no matter which server they are connected to, you should configure and enable user record synchronization, a feature of the Pulse Connect Secure platform.

Configuring a Role and Realm for Pulse Mobile Client for Android

To enable access from an Android device to Pulse Connect Secure the administrator must configure specific realm and role settings on the Pulse Secure server.

To configure Pulse Connect Secure for Android device access:

1. Log in to the Pulse Secure server admin console.
2. Select **User Roles > New User Role**.
3. On the New Role page, specify a name for the role and, optionally, a description. Make note of the name because later in this procedure, you create a realm and map realm users to this role.
4. In the Access Features section, select **Web and VPN Tunneling**.
5. Click **Save Changes** to create the role and to display the role configuration tabs.
6. Select **Web > Bookmarks** and then click **New Bookmark**.

   You must create bookmarks to enable the buttons that appear in the Pulse Mobile Client user interface. Typically, you create a bookmark for your company intranet and for Web e-mail. You must create an e-mail bookmark to enable the e-mail button within the Pulse Mobile Client interface on the Android device, and that e-mail bookmark must be named "Mobile Webmail".

Figure 1 Creating the E-mail Bookmark for Pulse Mobile Client

Note: Alternatively, you can use Web resource policies to define the bookmarks.

7. On the "VPN Tunneling" tab, set the "Split Tunneling" options by selecting the following options:

   Split Tunneling:

   - **Enable**: Split tunneling resource policies specify the traffic that passes through the VPN tunnel.
   - **Disable**: All network traffic goes through the VPN tunnel.

   Route Precedence:

   - **Tunnel Routes**: The route table associated with the Pulse Mobile Client virtual adapter take precedence. Pulse Mobile Client overwrites the physical interface routes if there is conflict between the Pulse Mobile Client virtual adapter and the physical adapters. Pulse Mobile Client restores the original routes when the connection is ended.
   - Tunnel Routes with local subnet access (Pulse Client on Windows and macOS only)
   - **Endpoint Routes**: The route table associated with the endpoint's physical adapter take precedence.

8. To change default session time-outs, select **General > Session Options**.
9. In the Session lifetime section, specify **Max. Session Length** in minutes. The remaining session time appears on the Pulse Mobile Client interface in the format days hours:minutes:seconds. The other session settings are not applied to mobile clients.
10. Select **Users > User Realms > New User Realm**.
11. Specify a name and, optionally, a description and then click **Save Changes** to create the realm and to display the realm option tabs.
12. On the "Role Mapping" tab for the realm, create a new rule that maps all users to the Android role you created earlier in this procedure.

Pulse Mobile Client and User Agent Strings

A user agent string is how a Web server identifies the type of client that is requesting service. A server can use that information to provide content that is tailored for the client. For example, a Web server can serve content for a particular browser. Portions of the Pulse Mobile Client user interface, such as the login screen and intranet bookmarks, are Web pages served from the Pulse Secure server and displayed by the embedded browser of Pulse Mobile Client. The appearance of these Web pages can be affected by how the Pulse Secure server is configured to map user agent strings to specific client types.

Note: You can configure user agent strings at the role or the realm level to define policies for a user based on client type.

To view and edit the user agent string and client type pairings on your Pulse Secure server:

1. Click **System > Configuration > Client Types**.

   To edit an existing item, click an item in the table to select it. The string pattern is available for editing in a text box and you can select the client type from a list box.

   To add a new user agent string and client type pairing, type a string pattern in the edit box at the top of the list, choose a client type from the list box, and then click **Add**. You can use the * and ? wildcard characters in your string. Note that user agent strings are not case-sensitive.

2. To reorder the list, click an item to select it and then use the up and down arrow buttons to move the item up or down in the list. When a browser requests access, the user agent string submitted by the browser is compared against the list starting at the top of the list and continuing down the list until the first match is reached.

   The default pairing (User Agent String = "*", Client Type = "Standard HTML") is listed as the last entry in the table to ensure that it is used only when no other pairing applies. You cannot edit, delete, or reorder the default pairing.

3. When you finish making changes to the table, click **Save Changes**.

Allowing Pulse Mobile Client for Android Users to Save Webmail Password

A Web bookmark on the role for Android users allows users to access e-mail through a Web link. You can allow users of the Pulse Mobile Client for Android app to save their e-mail password when they login to the e-mail system. After you have created a Mobile Webmail bookmark for the role used by Android users, enable password the option for user to save their e-mail password by doing the following.

1. Open the role you created for Android users.
2. Click **General > Session Options**.
3. In the section “Persistent Password Caching”, select **Enabled**.
4. Click **Save Changes**.

Host Checker for Pulse Mobile Client for Android

Host Checker is a component of Pulse Mobile Client that reports the integrity of Android endpoints that are attempting to connect to Pulse Connect Secure. Host Checker runs as a Trusted Network Connect (TNC) client on the endpoint. The client evaluates the endpoint according to predefined criteria and reports to the Trusted Network Connect server, which is a part of Pulse Connect Secure. If the endpoint is not in compliance with the Host Checker policies, then the user might not get access to the network or might get limited access to the network depending upon the enforcement policies configured by the administrator.

For Pulse Mobile Client on Android, Host Checker can evaluate client compliance based on the following predefined criteria:

- **OS Checks**: You can specify the iOS version or minimal version that must be installed on the device.
- **Root Detection**: Rooting is a process that allows Android users to gain root access to the operating system and bypass usage and access limitations imposed by device manufacturers and carriers. With a rooted device, a user can install applications that are not available and have not been certified by the device manufacturer or by the app store process. Rooted devices expose the device to a greater risk of running malicious applications. Host Checker can detect rooted devices and then allows or deny network access based on the Host Checker enforcement policy.

Configuring Host Checker for Pulse Mobile Client for Android

Host Checker policies can be part of a larger Host Checker configuration that applies to many different types of clients or to Android devices only.

To create a Host Checker policy for Android devices:

1. From the admin console, select Authentication > Endpoint Security > Host Checker.
2. In the Policies section, click New to open a New Host Checker Policy page.
3. Specify a name for the new policy and then click Continue to open the Host Checker Policy page.

   The name appears in lists when you implement the policy so be sure to use a descriptive name, such as Android HC Policy.
4. Click the Mobile tab, and then click the Android tab.
5. In the Rule Settings section, click Select Rule Type and select one of the following options and then click Add:

   - **OS Checks**: To specify the iOS version that must be installed on the device:

     1. Specify a descriptive name for this rule. For example, Must-Be-Android-2-2-or-higher. Rule names cannot include spaces.
     2. Specify the criteria. For example, to enforce Android 2.2 or higher, create two conditions: Equal to 2.2 and Above 2.2.

        Host Checker supports Android versions 1.6 through 3.1.
     3. Click Save Changes.

- **Rooting Detection**: Rooting is a process that allows Android users to gain root access to the operating system and bypass usage and access limitations imposed by manufacturers and service providers. With a rooted device, an Android user can install applications that have not been certified through the app store process. Rooted devices possess a greater risk of running malicious applications.

  1. Specify a descriptive name for this rule. For example, "No-Android-root".
  2. The **Don't allow rooted devices** check box is enabled by default.
  3. Click **Save Changes**.

6. After you have configured all of your rules, specify how you want to enforce them by choosing one of the following options:

   - All of the rules
   - Any of the rules
   - Custom

   For Custom requirements, you can specify a custom expression using Boolean operators AND and OR and group and nest conditions using parenthesis.

   Specify remediation options:

   - **Enable custom instructions**: If you enable this check box, a text box appears and allows you to type information that appears on the user's device if Host Checker discovers an issue. For example, if you enabled the MSS rule that terminates the VPN session of Host Checker discovers a virus, you can instruct the user to run a virus scan to clear the issue before trying to connect.
   - **Send reason strings**: Select this option to display a message to users (called a reason string) that is returned by Host Checker or integrity measurement verifier (IMV) and explains why the client machine does not meet the Host Checker policy requirements. For example, if the rooting detection policy fails, Pulse Mobile Client displays, "A rooted device is not allowed to access the network. Please contact your network administrator."

7. When you are finished, click **Save Changes**.

Implementing Host Checker Policies for Pulse Mobile Client for Android

After you create one or more Host Checker policies for Android devices, you must implement them. Pulse Connect Secure can use Host Checker policies at the realm or the role level.

Realm Authentication: You can configure a realm authentication policy to download and run Host Checker with a particular Host Checker policy. If the Android device does not meet the Host Checker requirements, then Pulse Connect Secure can deny access. You can provide remediation information in the Host Checker policy to describe the requirement and help users take steps to solve the issue.

To enable a Host Checker policy for a realm:

1. From the admin console, select **Users > User Realms > SelectRealm > Authentication Policy > Host Checker**. The Host Checker page displays all of the available Host Checker policies.
2. Select the check box next to each policy you want to include. Select one or both of the following check boxes next to the policy:
   - **Evaluate Policies**: Evaluates without enforcing the policy on the Android device and allows access.
   - **Require and Enforce**: Requires that the Android device be in compliance with the Host Checker policy. Pulse Connect Secure downloads Host Checker to the Android device after the user is authenticated and before the user is mapped to any roles in the system. Selecting this option automatically enables the **Evaluate Policies** option.
3. Optionally select **Allow access to realm if any ONE of the selected "Require and Enforce" policies is passed**. This check box is available if you selected more than one Host Checker policy. If you enable this check box, an Android device is allowed access if it passes any of the Require and Enforce policies. The Cache Cleaner policy does not apply to Android devices.
4. Click **Save Changes**.

Role: You can configure a role to download and run Host Checker with a particular Host Checker policy. If the Android device does not meet the Host Checker requirements, then Pulse Connect Secure can deny access or assign the user to a remediation role that has limited access. You can provide remediation information in the Host Checker policy to help users take steps to solve the issue.

To enable a Host Checker policy for a role:

1. From the admin console, select **Users > User Roles > SelectRole > General > Restrictions > Host Checker**. The Host Checker page displays all of the available Host Checker policies.
2. Select **Allow users whose workstations meet the requirements specified by these Host Checker policies**.
3. In the Available Policies list, select the policies that you want to apply to select them, and then click **Add** to move them to the Selected Policies list. To select a policy click it. To select more than one policy, use CTRL+click.
4. Optionally select **Allow access to realm if any ONE of the selected "Require and Enforce" policies is passed**. This check box is available if you selected more than one Host Checker policy. If you enable this check box, an Android device is allowed access if it passes any of the Require and Enforce policies. The Cache Cleaner policy does not apply to Android devices.
5. Click **Save Changes**.

Single Logout

Single logout is a mechanism for logging out a particular user from all the sessions created by the identity provider. This option allows the system to receive and send a single logout request for the peer SAML identity provider. Select the SLO service URL from the list to complete the Single Logout Service URL setting using metadata. The list is populated by the identity provider entities defined in metadata files added to the **System** > **Configuration** > **SAML** page. The system sends Single Logout requests to this URL. In addition, if you use the metadata option, the Single Logout Response URL setting is completed based on the selection for Single

Logout Service URL. If the identity provider has left this setting empty in its metadata file, the system sends the Single Logout response to the SLO service URL. If you complete these settings manually, ask the SAML identity provider administrator for guidance. The Support Single Logout service for the identity provider must present a valid certificate.

Split Tunneling

Split tunneling is configured as part of the role that is assigned to a user after authentication. When Pulse Client and Pulse Connect Secure establish a VPN tunnel, Pulse Connect Secure takes control of the routing environment on the endpoint to ensure that only permitted network traffic is allowed access through the VPN tunnel. Split tunneling settings enable you to further define the VPN tunnel environment by permitting some traffic from the endpoint to reach the local network or another connected subnet. When split tunneling is enabled, split tunneling resource policies enable you to define the specific IP network resources that are excluded from access or accessible through the VPN tunnel.

For more information on split tunneling refer to PCS administration guide

IPv6/IPv4 Split Tunneling

Pulse Client now allows accessing both IPv4, IPv6 corporate resources from IPv4 and IPv6 endpoints. It enables client to access both corporate network and local network at the same time. The network traffic designated are directed to tunnel interface for corporate network by configuring route policies, whereas other traffics are sent to direct interface.

Pulse Secure client Options

- Route Precedence—You can define which routing table takes precedence:
- Tunnel Routes—The route table associated with the Pulse virtual adapter take precedence. Pulse overwrites the physical interface routes if there is conflict between the Pulse virtual adapter and the physical adapters. Pulse restores the original routes when the connection is ended.
- Tunnel Routes with local subnet access —Network traffic addressed to the networks defined in the split tunnel resource policies goes through the VPN tunnel. Network traffic that is addressed to the directly connected (local) subnet goes to the local subnet. The default route is set to the local subnet, so all other network traffic is subject to the original endpoint routing table.
- Endpoint Routes—The route table associated with the endpoint's physical adapter take precedence.
- Route Monitor—Pulse can monitor the route tables and take appropriate action.
- Yes – VPN tunneling ends the connection only if the route change affects the VPN tunnel traffic. For example, if the route metric is changed higher, it should not disconnect VPN tunneling.

  No – Route tables can change on the client endpoint.

FQDN Split Tunneling

FQDN (Fully qualified domain name) based split tunneling lets the administrator configure split tunneling rules by directly specifying the domain names. This is helpful while configuring rules to ignore or tunnel cloud services. For FQDN resources wild card domains are be allowed.

Minimum Client Enforcement

This feature allows administrator to configure a minimum client version. If the client has version lower than the configured minimum version, then the PCS server rejects the client connection. If the client is older than the configured minimum client version, then PCS gateway rejects the connection.

User can upgrade it later through browser or SCCM server.

For example, the gateway can host Pulse Client 9.1R9, but the minimum version could be Pulse Client 9.1R1. Similarly, if Pulse Client 9.1R2 is connected to the gateway it would prompt for an upgrade, if the active version is greater than the Pulse Client version. For Pulse Client 9.1R1, the connectivity is rejected and displays an appropriate error message.

Pulse Mobile Client for Android Error Message Reference

The following error message summary for Pulse Mobile Client for Android describes possible issues and suggests resolution actions where possible.

Table 1 Pulse Mobile Client for Android Error Messages

Message	Possible Causes	Suggested Actions
The certificate for this server is invalid. Tap Accept to connect to this server anyway.	The server certificate for the page received from the Pulse Secure server is not valid.	Tap View Certificate to examine the certificate. Tap Accept and open the URL.Tap Decline. The connection attempt is ended. Contact the Pulse Secure administrator.
Session disconnected due to invalid certificate.	The Pulse Mobile Client downloads session information from the Pulse Secure server and the certificate received from the server does not match the stored session certificate. While accessing e-mail or intranet, the certificate received from Pulse Secure server does not match the previously stored session certificate.	Click the Close button on the Alert dialog to return to home screen. Try the operation again. If the error occurs again, contact the Pulse Secure administrator.
On the Pulse Mobile Client status screen, the VPN status appears as Not Supported.	The device does not support VPN.	Click the Close button on the Alert dialog to return to home screen. Try the operation again.
Failed to connect to the server.	Sign-in Process failed.	Check the network connection (for example, Wi-Fi, 3G, etc.), and then try the operation again.
Failed to process the HTML information from the server.	Sign-in Process failed.	Try the operation again. If the error occurs again, contact the Pulse Secure administrator.

Message	Possible Causes	Suggested Actions
Your session was terminated. Please connect again.	An invalid URL was used while accessing intranet and e-mail pages.	Reconnect to the Pulse Secure server. If the error occurs again, contact the Pulse Secure administrator.
Compliance Check couldn't be completed.	Host Checker compliance check could not be completed during sign-in process.	Try the operation again. If the error occurs again, contact the Pulse Secure administrator.

Introduction to the Pulse Mobile Client for Android API

The API for the Pulse Mobile Client for Android allows a third-party application to establish a VPN with the Pulse Secure Access Service. In addition to adding, changing, and deleting VPN connections, the API lets you check the current API version, verify whether a VPN already exists, and add a certificate to the Android key store.

API Access Requirement

To access the API, the package name and signing certificate of the calling application must be submitted to Pulse Secure for incorporation into Pulse Mobile Client. The client stores an MD5 checksum of the following string:

```
<package name> + ":" + <signing signature>
```

If the application has multiple signing signatures, only one of the signatures is needed.

Package Name

net.pulsesecure.pulsesecure

AIDL File

The AIDL file used to define the client interface is named `IVpnProfile.aidl` and contains the following:

```
package net.pulsesecure.pulsesecure.vpnprofile;

//Package name updated in API version 3.

interface IVpnProfile {
   int doCommand(String xml); //deprecated
   int getVersion();
    // Create a profile using data provided
    int createConnection(in String jsonProfile);
    // Delete a profile
    int removeConnection(String profileName);
    // List all profiles
    List<String> getAllConnections();
    // Access a profile's connection data
    String getConnection(String profileName);
```

```
    // Start and stop a VPN connection for a given profile
    int startConnection(String profileName);
    int stopConnection(String profileName);
    // State info for this profile
    int getState(String profileName);
    String getErrorString(String profileName);
}
```

This file must be included in the `src/` directory used to compile the application. For more information about the AIDL, see http://developer.android.com/guide/components/aidl.html

Service intent filter action name

Use the below action name to bind to a service implementing this API. Updated in API version 3.

"net.pulsesecure.pulsesecure.vpnprofile.IVpnProfile"

JSON String Format for createConnection

The `createConnection` method, new in version 2 of the API, accepts a VPN profile described in a JSON string format. The JSON strings vary depending on the type of authentication used in the VPN profile, as shown in the following examples.

Single Username/Password Authentication

The following sample JSON string is for a VPN profile that uses a username/password for authentication:

```
{
   "MDM_VPN_PARAMETERS": {
      "profile_attribute": {
         "profileName":"MyVpnProfile",
         "host":"https://vpn.xyz.com/auth",
         "vpn_type":"ssl",
         "isUserAuthEnabled":true
      },
      "ssl": {
         "basic": {
            "username":"testuser",
            "password":"secret123"
         }
      },
      "vendor": {
         "realm":"",
         "role":"",
         "certAlias":""
      }
   }
}
```

Certificate Authentication

The following sample JSON string is for a VPN profile that uses a certificate for authentication:

```
{
   "MDM_VPN_PARAMETERS": {
      "profile_attribute": {
         "profileName":"MyVpnProfile",
         "host":"https://vpn.xyz.com/auth",
         "vpn_type":"ssl",
         "isUserAuthEnabled":true
      },
      "ssl": {
         "basic": {
         }
      },
      "vendor": {
         "realm":"",
         "role":"",
         "certAlias":"myCertName"
      }
   }
}
```

Note that the certificate must be installed in Android's keystore before this profile can be used. The certificate name is specified in the `certAlias` field in the `vendor` section.

Username/Password and Certificate Authentication

The following sample JSON string is for a VPN profile that uses both certificate and username/password for authentication:

```
{
   "MDM_VPN_PARAMETERS": {
      "profile_attribute": {
         "profileName":"MyVpnProfile",
         "host":"https://vpn.xyz.com/auth",
         "vpn_type":"ssl",
         "isUserAuthEnabled":true
      },
      "ssl": {
         "basic": {
            "username":"testuser",
            "password":"secret123"
         }
      },
      "vendor": {
         "realm":"",
         "role":"",
         "certAlias":"myCertName"
      }
   }
}
```

Dual Username/Password Authentication

The following sample JSON string is for a VPN profile that uses two usernames/passwords for authentication:

```
{
   "MDM_VPN_PARAMETERS": {
      "profile_attribute": {
         "profileName":"MyVpnProfile",
         "host":"https://vpn.xyz.com/auth",
         "vpn_type":"ssl",
         "isUserAuthEnabled":true
      },
      "ssl": {
         "basic": {
            "username":"testuser",
            "password":"secret123"
         }
      },
      "vendor": {
         "realm":"",
         "role":"",
         "username2":"testuser2",
         "password2":"passwd123"
      }
   }
}
```

Note that the primary username/password are specified in the `ssl` section and the secondary username/password are specified in the `vendor` section.

UiLess and Per-App VPN

The following sample JSON string is for UiLess and Per-App VPN connections:

For UiLess Connection, set UiLess parameter to true

For Per-App VPN connection, :

1. `vpn_route_type` is set to1
2. `action` is set to 0 to allow and 1 to deny the application traffic through tunnel
3. Add applications to the `appList` in the specified format

```
{
    "MDM_VPN_PARAMETERS": {
        "profile_attribute": {
         "profileName":"MDM Profile",
         "host":"https://myvpn.net/vpn ",
         "vpn_type":"ssl",
         "vpn_route_type":1,
         "isUserAuthEnabled":true
      },
      "ssl": {
         "basic": {
```

```
                "username":"",
                "password":"",
                "authentication_type":1
            }
        },
        "vendor": {
              "uiLess":true,
              "realm":"",
              "role":"",
              "username2":"",
              "password2":"",
              "certAlias":"mycertalias",
              "appVpn": {
                    "action":0,
                    "appList":
                    [{
                          "appId":"1",
                          "appPkg":"com.android.chrome",
                          "appName":"Chrome"
                    },
                    {
                          "appId":"2",
                          "appPkg":"com.google.android.apps.plus",
                          "appName":"Google Plus"
                    }]
                    }
        }
    }
    }
```

API Functions

This chapter describes each of the function calls for the Pulse Mobile Client API.

getVersion()

Retrieves the version number of the client API.

Syntax:

```
getVersion()
```

Parameters:

None.

Returns:

An integer indicating the API version (the first version is 1). The current version is 3.

Since:

5.0R1

createConnection(String jsonProfile)

Creates a new VPN profile.

Syntax:

```
int createConnection(String jsonProfile)
```

Parameters:

The jsonProfile is the VPN profile in the JSON string format. See the examples in "JSON String Format for createConnection" on page 13.

Returns:

This method returns 0 on success and 1 on error. When this method returns 1, the caller can call `getErrorString()` to obtain a description of the error.

Since:

5.0R3

removeConnection(String profileName)

Removes an existing VPN profile.

Syntax:

```
int removeConnection(String profileName)
```

Parameters:

The `profileName` is the name of a VPN profile that was created with the functions addVPNConnection or "createConnection(String jsonProfile)" on page 17.

Returns:

This method returns 0 on success, 1 if the profile does not exist, and -1 on other errors. One of the other errors occurs when an active VPN session is using the profile. When this method returns 1 or -1, the caller can call `getErrorString()` to obtain a description of the error.

Since:

5.0R3

getAllConnections()

Gets the profile names for all connections added by the same caller.

Syntax:

```
List<String> getAllConnections()
```

Parameters:

None

Returns:

This method returns the profile names for all VPN profiles added by the same caller. VPN connections added by the device user or by other callers are not included. This method could return null or an empty list.

Since:

5.0R3

getConnection(String profileName)

Obtains the JSON profile for a VPN profile created with the function "createConnection(String jsonProfile)" on page 17.

Syntax:

```
String getConnection(String profileName)
```

Parameters:

The `profileName` is the name of the VPN profile.

Returns:

This method returns the VPN profile for the name specified. This profile must be added by the same caller. This method could return null.

Since:

5.0R3

startConnection(String profileName)

Starts a VPN session using the specified VPN profile.

Syntax:

```
int startConnection(String profileName)
```

Parameters:

The `profileName` is the name of the VPN profile.

Returns:

This method returns 0 on success, 1 if the profile does not exist, and -1 on other errors.

When `startConnection()` returns 0, it means that a VPN session is getting started. It is possible that a VPN session may not be established after `startConnection()` returns 0. The caller can call `getState()` to obtain the VPN state. When the VPN fails to start, the caller can call `getErrorString()` to obtain a description for the error.

Since:

5.0R3

stopConnection(String profileName)

Stops an active VPN session for the specified VPN profile.

Syntax:

```
int stopConnection(String profileName)
```

Parameters:

The `profileName` is the name of the VPN profile.

Returns:

This method returns 0 on success, 1 if the profile does not exist, and -1 on other errors.

When `stopConnection()` returns 0, it means that a VPN session is being shut down, but is not yet complete. The caller can call `getState()` to obtain the VPN state.

Since:

5.0R3

getState(String profileName)

Get the current VPN state for the specified VPN profile.

Syntax:

```
int getState(String profileName)
```

Parameters:

The profileName is the name of the profile.

Returns:

The possible return values are:

```
public enum KnoxVpnState {
   IDLE(1),
   CONNECTING(2),
   DISCONNECTING(3),
   CONNECTED(4),
   FAILED(5),
```

```
   DELETED(6),
   // State(s) defined by our app. To avoid conflict with KNOX state  // definitions, we
are starting the value from 101;  PS_USER_SESSION_CREATED(101);
}
```

When the return value is KnoxVpnState.FAILED, the caller can call `getErrorString()` to obtain a description of the VPN failure.

Since:

5.0R3

getErrorString(String profileName)

Get the error string for the last API call or for the VPN failure.

Syntax:

```
String getErrorString(String profileName)
```

Parameters:

The `profileName` is the name of the profile.

Returns:

```
public static final String ERR_PROFILE_NAME_EMPTY = "Profile name is empty";
public static final String ERR_PROFILE_NOT_FOUND = "VPN profile not found";
public static final String ERR_DATABASE_EXCEPTION = "Exception accessing profile DB";
public static final String ERR_NO_ERROR = "No Error";
public static final String ERR_NO_NETWORK = "There is no network connectivity to perform
the requested action";
public static final String ERR_NO_ACTIVE_VPN_SESSION = "An active VPN session with the
profile name does not exist";
public static final String ERR_DELETE_PROFILE_FAILED = "Failed to delete the profile
from DB";
```

Since:

5.0R3

Using Pulse Client with PZTA

Pulse Secure provides a PZTA-ready version of the Pulse Client software required for end-user devices to be able to connect to your secure applications and resources.

Pulse Client connects to PZTA services, by default, through an on-demand connection basis and can handle multiple simultaneous PZTA and non-PZTA connections. To learn more, see On-Demand and Simultaneous Connection Handling.

Pulse Client maintains communication with the PZTA Controller to continuously-enable synchronization of policy and configuration updates. Through this mechanism, user requests to access resources and applications are subject to continuous assessment for risk and authorization. For more details, see Dynamic Policy Update and CARTA.

To learn more about enrolling user devices for use with PZTA, see Enrolling a User Device.

On-Demand and Simultaneous Connection Handling

While active, Pulse Client maintains two connection channels for PZTA services, a control channel to the PZTA Controller, and a data channel to your PZTA Gateways. For more details on networking considerations when deploying Gateways, see Working with Gateways.

The control channel connection to the PZTA Controller is activated when Pulse Client is started up and remains in an always-on state, silently in the background. If Pulse Client is able to locate a valid session cookie from an earlier session, the connection is re-established automatically. If no valid cookie is present, Pulse Client requests re-authentication from the user. The PZTA Controller connection is terminated when Pulse Client is shut down.

Pulse Client creates data channel connections to PZTA Gateways as an on-demand service. That is, connections to resources and applications controlled by PZTA Gateways become active only when required, and the connection is suspended after a period of inactivity. The user remains unaware of the connection state, unless re-authentication becomes necessary. As a user makes a request for a resource, Pulse Client transitions automatically from disconnected to connected. The connection remains in this state for the duration of the session, or until one of the following events occurs:

- An idle time-out occurs (after 5 minutes)
- The connection is actively placed in a disconnected state
- Pulse Client is shut down

To avoid the data channel being reconnected unnecessarily, non-PZTA DNS traffic is redirected to the device's physical network adapter.

Applicable Pulse Client versions can manage simultaneous connections with the PZTA Controller, and with other Pulse Secure services such as Pulse Connect Secure (PCS). While PCS connections must be activated and deactivated by the user, connections to PZTA are provided on-demand, as mentioned. Therefore, a PZTA connection in the Pulse Client does not provide the same **Connect** and **Disconnect** controls. Instead, PZTA connections include only a **ZTA** button to provide access to the PZTA Applications page. If this button is active, the connection to the Controller has been established. If the button is inactive, the connection to the Controller has not yet been established, or a communication problem has occurred. In this case, access to your applications is prevented.

When running active connections to both PZTA and PCS simultaneously, note that the following PCS features are not supported:

- Route Monitoring
- Traffic Enforcement
- Stealth Mode
- Always on VPN/LockDown
- Location awareness
- IPv6 support

Disabling the PZTA Connection

Pulse Client additionally provides the ability to actively disable the on-demand connection feature. Use of this facility disables the PZTA connection, avoiding the scenario where Pulse Client attempts to repeatedly request authentication even after the user might be unable to authenticate due to too many failed attempts, or where the user just does not require access to any PZTA-controlled resources during that session.

If a user attempts to request a PZTA-controlled resource during the period a PZTA connection is disabled, the request fails. Other Pulse Client connections are unaffected.

For Pulse Client on Android devices, use the **Disconnect** option in the Pulse Client application. Open Pulse Client, locate the PZTA connection profile, and tap the **Disconnect** button.

By setting the PZTA connection to be disconnected, Pulse Client suspends both the control channel and the data channel (where either are active). If the control channel was previously logged-in to the PZTA Controller, this remains the case to facilitate session resumption through a subsequent reconnect.

The disconnect feature is not activated by clicking or tapping Cancel in the PZTA authentication dialog. Canceling an authentication request triggers a timeout interval, after which Pulse Client re-displays the authentication dialog. The disconnect feature instead disables the authentication request process until the user manually reinstates it.

To reinstate the PZTA connection on Android devices, tap the ZTA button in the PZTA connection profile in the Pulse Client application:

If the existing session cookie is still valid, the control channel is re-established. If the session is now invalid, Pulse Client prompts the user for their PZTA credentials as normal. On successful re-establishment of the PZTA session, the user is presented with the PZTA End User Portal in the default browser.

When restarting Pulse Client, PZTA connections default to being on-demand services. That is, a previously disabled PZTA connection is re-enabled when Pulse Client starts.

Dynamic Policy Update and CARTA

To complement the zero-trust approach, PZTA supports dynamic policy updates and CARTA (Continuous Adaptive Risk and Trust Assessment) for your end user devices. This framework establishes an approach of continuous assessment and updating of secure access policies on the client, without the requirement to disconnect and reconnect to establish an updated authorization posture.

As your policies, applications, and authentication configuration are updated by the administrator on the PZTA Controller, changes are synchronized out to client devices dynamically and take effect immediately. Pulse Client ensures that any application updates are applied and any new authentication requirements are met before continuing the session, providing the end user with a seamlessly-updated experience. This method ensures that Pulse Client is always updated at the point of change, and not just when establishing a connection to a PZTA Gateway to access an affected resource.

The CARTA implementation in Pulse Client means that the security posture of the end user is continuously assessed in conjunction with policies configured in the Controller, with allow or deny decisions enforced through dynamic assessment and updating of the current policy set. Where application access is denied or restricted, Pulse Client informs the user of any access restrictions or policy contravention at the point of use. For example, the PZTA Desktop Client Home page updates to provide visual cues with applicable error messages whenever a specific application becomes unavailable:

By hovering your pointer over the warning symbol in the inactive application, PZTA provides an explanatory message.

Furthermore, user attempts to access a restricted web resource in a browser trigger a CARTA response with Pulse Client presenting a pop-up resource blocked message:

Pulse Client implements a no-repeat interval for resource blocked messages of 2 minutes, to avoid a user repeatedly seeing the same pop-up message for every browser request for the same restricted resource. While the resource remains blocked to further access attempts, no further messages are displayed by Pulse Client until after 2 minutes has elapsed. You can force Pulse Client to continue hiding blocked resource messages indefinitely by right-clicking the connection in the Pulse Client dialog and selecting Disable Block Messages. To re-enable showing blocked resource messages, select Enable Block Messages.

Enrolling First Time Users

When enrolling a new device, an authorized user contacts the PZTA Controller to activate an initial first-time enrollment of their client device. The Controller responds to a valid enrollment request by activating a download of Pulse Client along with a suitable client certificate.

After Pulse Client is installed, a secure connection request is attempted with the Controller. The request is validated against the designated authentication policy applicable to that combination of user and device and, where successful, a connection profile is downloaded to the client. This profile enables Pulse Client to set up a secure tunnel directly to the PZTA Gateway serving the resource set the client is authorized to view.

Enrolling Existing PZTA Users

After you have enrolled the new device, Pulse Client is installed and configured with the policies and settings relevant to the device type. Your application and resource access rights should be duplicated to the new device.

Note: If a user device is currently using a Beta version of the PZTA-ready Pulse Client, Pulse Secure advises to remove the PZTA connection from Pulse Client and to re-perform the enrollment procedure through a web browser. For more details, see your support representative.