

Configuring HTTP or HTTPS and SSL for SBA Interface

- Feature Summary, on page 1
- Feature Description, on page 2
- How it Works, on page 2
- Configuring Support for HTTP or HTTPS and TLS, on page 3
- HTTP and SSL for SBA Interface OA&M Support, on page 4

Feature Summary

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	PCF
Applicable Platform(s)	SMI
Feature Default Setting	Enabled – Configuration required to disable
Related Documentation	Not Applicable

Revision History

Table 2: Revision History

Revision Details	Release
First introduced.	2020.03.0

Feature Description

In the SBA framework, the PCF exchanges data across the interconnected network functions and data repositories. The communication within this framework is established over a secure layer comprising of Hypertext Transfer Protocol (HTTP) or HTTPS using Transport Layer Security (TLS). TLS offers a secure network layer transportation of data between the components. However, PCF also offers support for HTTPS without TLS.

In this release, TLS provides a transport layer encryption between the nodes for the security compliance purposes. This feature does not support the NF security requirements as per the 3GPP specifications of 5G.

The PCF provides HTTP or HTTPS support for the N7, N15, N28, N36, and NRF interface. The information transmission between the client and server happens through the HTTPS requests.

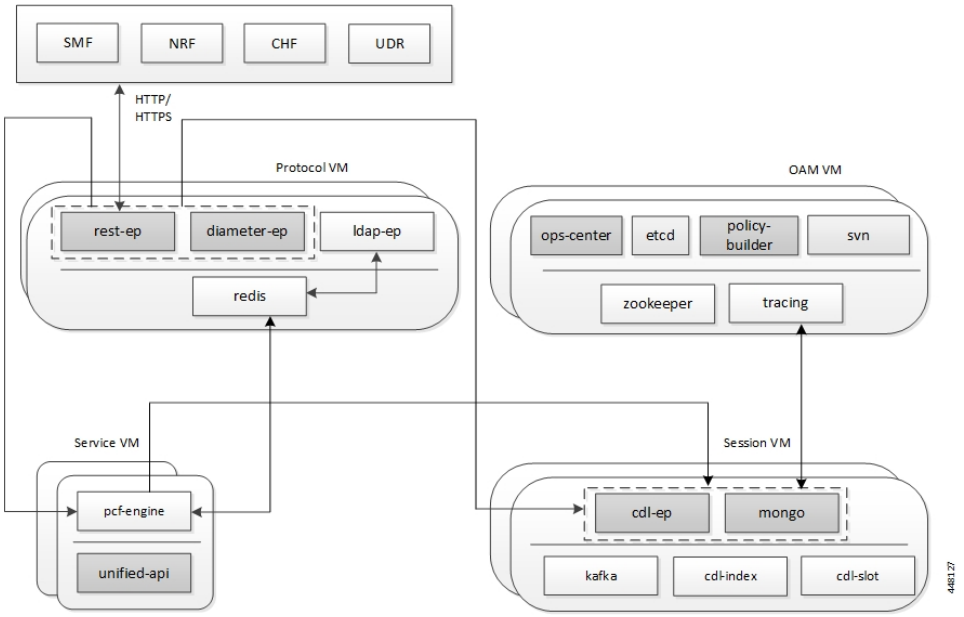
How it Works

This section describes how this feature works.

The implementation of HTTP or HTTPS with TLS in PCF requires you to configure the HTTP and HTTPs secure port for each interface. To enable a TLS handshake, import the signed certificate into the PCF Ops Center from a trusted source. The PCF supports both server and client HTTPS requests. By default, the PCF supports HTTP requests without TLS.

The following graphic illustrates the communication flow between NFs and REST endpoint.

Figure 1: HTTP or HTTPS Communication Flow



The support for HTTP and HTTPS in PCF involves the following steps:

1. Configure the ca-certificates which are required for TLS. The certificate data must be in the PEM format and residing in the Java KeyStore (JKS).

2. Configure the certificate and private key for establishing the TLS channel between the server and client. Obtain the private key from the certificate.
3. By default, the uri-scheme is associated with the HTTP. Enable HTTPS by associating the rest-endpoint uri-scheme with the HTTPS. PCF invokes the configured server certificate when starting up the pcf-rest-ep pod. This step ensures that the SSL context is set for the REST server. When PCF is a client that initiates N28, nNRF, or the N15 requests, the HTTP or HTTPS protocol is specified in the endpoint profile.

   The rest-endpoint server detects all the certificates from the Kubernetes secrets during a startup. An individual Kubernetes secret is created for each certificate. These secrets are mounted on the rest-endpoint pods, at /config/secrets location during its deployment. All the certificates are loaded into the keystore that is located at /opt/workspace/rest-ep/certs/server/keystore. If the HTTPS is configured as the uri-scheme, then the HTTP server initiates the SSL context with the certificate name configured. For messages initiated from the REST endpoint (PCF as client), the HTTP client loads all the certificates from the keystore.

Configuring Support for HTTP or HTTPS and TLS

This section describes how to configure the HTTP or HTTPS and TLS from the PCF Ops Center.

Configuration of HTTP or HTTPS and TLS involves:

- Configuring Server and Client Certificates

Configuring Server and Client Certificates

This section describes how to configure the certificates for the server and client.

To configure the certificates for the server and client, use the following configuration in the PCF Ops Center:

```
config
   pcf-tls
     ca-certificates [name]
       cert-data certificate_pem
     certificates [name]
       cert-data certificate_pem
       private-key certificate_private_key
       end
```

NOTES:

- **ca-certificates [name]** —Specify the certificate name. The list of certificates names is displayed based on the configured certificates.
- **certificates [name]**—Specify the certificate name. The list of certificates names is displayed based on the configured certificates.
- **cert-data** *certificate_pem*—Specify the cert-data value in the PEM format.
- **private-key** *certificate_private_key*—Specify the private key value in the Public-Key Cryptography Standards (PKCS) #8 format.

Obtaining the Private key

This section describes how to obtain the private key from a certificate.

To obtain the private key, perform the following procedure:

1. Convert the certificate from PEM to PKCS12 format using the following:

   openssl pkcs12 -export -out *pkcscertificate.p12* **-inkey** *certificatekey.pem* **-in** *inputcertificate.pem*

2. Extract the private key from the PKCS12 certificate created in the previous step by using the following:

   openssl pkcs12 -in *pkcscertificate.p12* **-nocerts -nodes -out** *privatekey.pem*

3. Convert the private key to a PKCS8 key using the following:

   openssl pkcs8 -in *privatekey.pem* **-topk8 -nocrypt -out** *privatekey.p8*

Verifying the Certificate Status

This section describes how to verify the configuration status of the certificates.

The following configuration is a sample output of the **show rest-endpoint certificate-status** command:

```
CERTIFICATE
NAME         TIME TO EXPIRE
-------------------------------------------
pcfserver    3649 days 10 hours 25 minutes
cacert       3610 days 13 hours 55 minutes
pcfclient    334 days 21 hours  26 minutes
```

HTTP and SSL for SBA Interface OA&M Support

This section describes the operations, administration, and maintenance information for this feature.

Statistics

This section provides gauge that is generated for computing the HTTP TLS certificate validity information.

- http_tls_cert_validity: This gauge fetches the duration (in milliseconds) after which the certificate expires. The cert_name label fetches the certificate name.

  An example of the Prometheus query:

  ```
  abs(http_tls_cert_validity)>0
  ```