PCoIP Connection Broker Protocol Overview

The PCoIP Connection Broker Protocol (hereafter referred to as the Broker Protocol) is a web application program interface (API) used by PCoIP clients to securely communicate with connection brokers that manage PCoIP sessions and resources. A PCoIP session provides access to a remote desktop or application.

This document provides an overview of the Broker Protocol, the devices and software commonly involved with Broker Protocol connections, and an example of a Broker Protocol session flow. This document is not a specification and should be only considered as an informal example.

Normative Broker Protocol specifications

Consult the **PCoIP Connection Broker Protocol Specification** for further details and as a normative reference. This document is part of Cloud Access Software, and is distributed by agreement. To request access to this document, contact your Teradici representative.

The connection broker is a portion of the overall solution in the Teradici Cloud Access Software. From the PCoIP client's perspective, the connection broker authenticates the user, provides a list of 'entitled resources' or inventory to the user for selection, and provides connection information to the selected resource. A minimal broker implementation would provide password authentication and inventory and connection information.

Components and architecture of Teradici Cloud Access Software

Refer to Teradici Cloud Access Software Architecture Guide for further information about the components in the Teradici Cloud Access Software.

The remote resource, being either a remote desktop or remote application, is provided by a virtual or physical host where an appropriate PCoIP agent is installed, licensed, and running.

The Broker Protocol facilitates the following functionality:

- Authenticating a user to access remote resources.
- Querying which resources are available to a user.

- Provisioning the host resource.
- Selecting the host resource to establish a PCoIP session.
- Establishing a PCoIP session.
- Creating network security deployment topologies involving gateways and proxies.
- Creating distributed authentication deployment topologies.
- Providing high availability deployments.

Partner Requirements

A Teradici partner implementing the Broker Protocol will implement different aspects of the system depending on which PCoIP components the partner is using.

One case is where the partner is implementing a custom deployment and brokering system, but generally consuming the rest of the platform—such as connection managers, PCoIP agents and PCoIP clients—‘out of the box’. In this case, the partner is responsible for implementing Broker Protocol API from the ‘broker’ or ‘server’ perspective.

Other partners, such as gateway, authentication, or client partners, may need to implement both from the broker or server perspective, as well as from the client side.

User Authentication

The connection broker is responsible for specifying which authentication methods, if any, must be employed to authenticate a user before requesting a list of resources to which the user is entitled.

User authentication is used both for ensuring the user has access to the overall system, as well as providing, when possible, single-sign-on functionality such that a user only needs to enter credentials once to access the brokering system and the resource.

The user authentication methods discussed next are supported by the Broker Protocol. Brokers implement the methods as appropriate for their systems and usage scenarios. Of these methods, the password method is the only required method in the protocol. Other methods are optional.

- Password method: The user is authenticated using the operating system (for example, Windows) username, password, and domain name.

- **Token authentication method**: The token authentication method allows the client to reconnect to a previously disconnected PCoIP session using user credentials cached on the client machine. The user does not need to re-enter credentials.
- **Disclaimer method**: The user is prompted to accept a disclaimer statement as one of the authentication steps/methods. The connection broker provides the disclaimer text to be accepted by the user.
- **Dialog method**: In this case, the broker sends information to the client to enable the client to create a dialog box for user input. This method is suitable for a large number of cases, including multi-factor and challenge-response authentication methods.
- **ID card method**: The ID card is a physical identification card that has been issued to the user. The PCoIP client reads the ID card to obtain the card number stored in the card, and provides it to the connection broker to identify and/or authenticate the user.

In addition, the following authentication methods are available. These cases require more involved system changes and so cannot be implemented without direct interaction with Teradici:

- **OAuth authentication method**: The client uses the OAuth protocol to authenticate a user. User credentials are not passed to the client nor sent over the Broker Protocol. After authenticating a user, the client is granted an OAuth authorization code or access token which is passed over the Broker Protocol to access the resource.
- **Kerberos authentication method**: The client uses the Kerberos protocol to authenticate a user. After authenticating a user, the client is granted a Kerberos Ticket-Granting Ticket which is passed over the Broker Protocol to access the resource.

Optional Features

The protocol has other optional features, which may be implemented by partner products. These include:

- Updating a user's password.
- Performing operations on resources such as operating system restart, sleep, resume, power off and on, and wake-on-LAN.
- Conveying resource state and health information to the user.
- Identifying client geographical location through IP address.

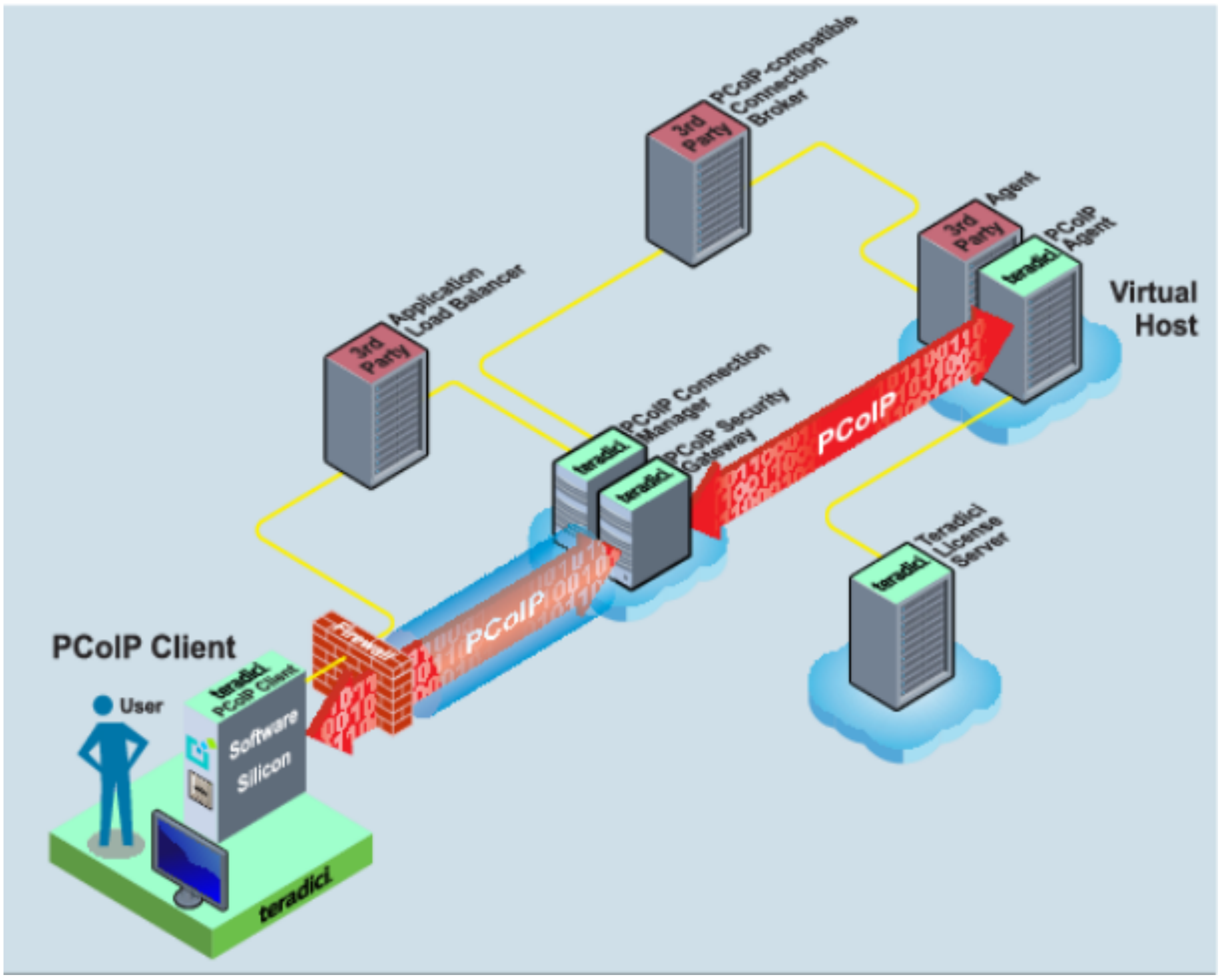
- Remapping IP spaces through network address translations (NATs) to support connections from outside the local network.
- Localizing output for languages other than US English. Teradici Zero Clients support a variety of locales. Consult the release notes for each zero client for an accurate list.

Deployment Scenarios

This section provides a few example deployments in which the Broker Protocol is used to establish a PCoIP session to remotely access a desktop or application.

External or WAN Deployments

The following diagram shows an example of a user using the Broker Protocol to establish a PCoIP session in a typical external or WAN deployment.



As seen in the diagram, the PCoIP Connection Manager enables the client and the agent to establish a remote desktop connection by creating a PCoIP session. The PCoIP Security Gateway enables WAN users to securely access their remote desktops via the Internet without setting up a

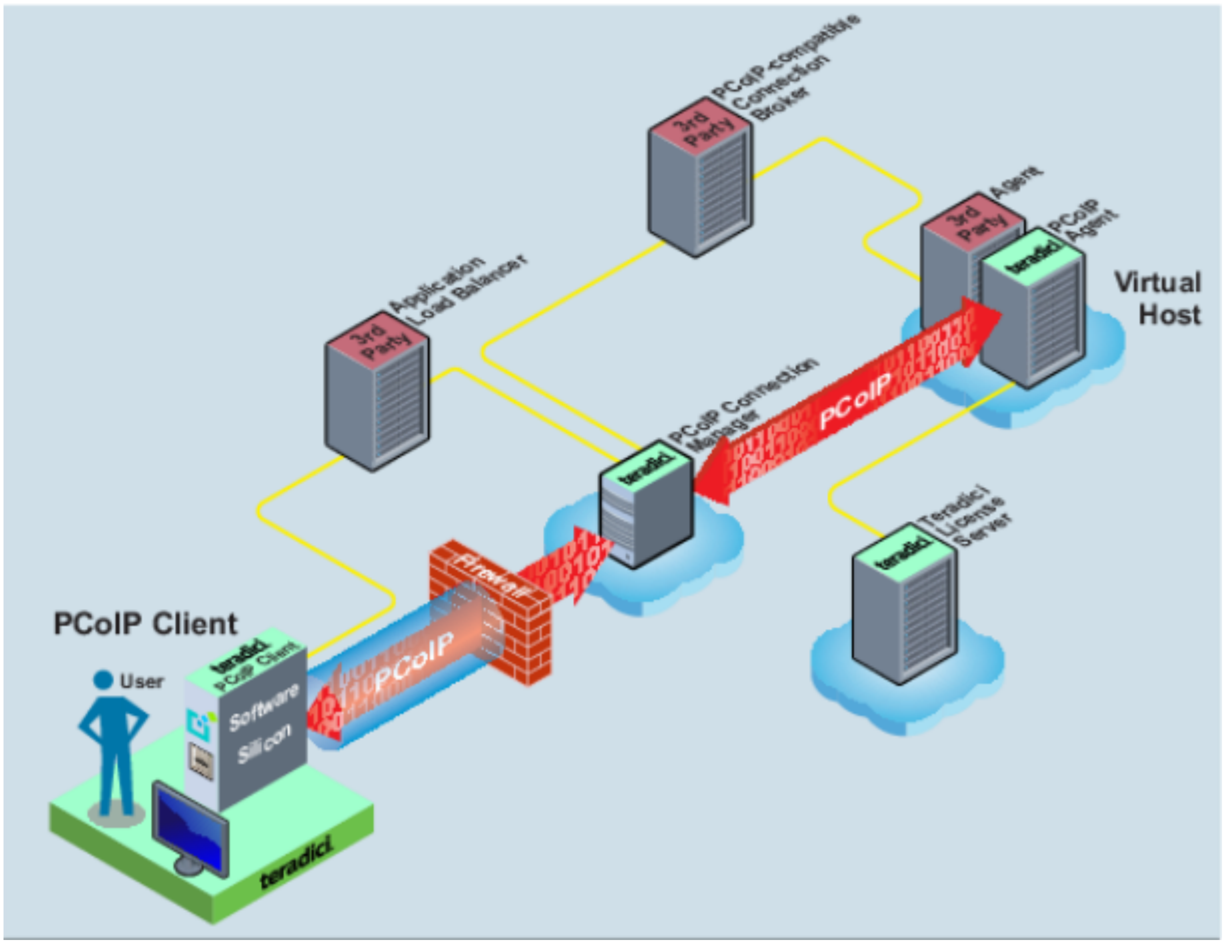
VPN connection, while the connection broker authenticates the user and queries the desktop and applications.

In larger WAN deployments, multiple connection managers and security gateways should be deployed to provide high availability. A load balancer may then be deployed to distribute PCoIP sessions over the pairs of connection managers and security gateways. The load balancer should be an HTTPS reverse-proxy that supports cookie-based session stickiness.

The third-party broker agent component in the host desktop is an optional component used by the connection broker to manage and monitor the desktop state. The broker agent and PCoIP agent do not directly interact, but the broker agent may sample information about PCoIP sessions. As an example, the broker agent may query PCoIP session status through the PCoIP Windows WMI statistics counter on Windows hosts.

Internal LAN or VPN Deployments

The following diagram shows an example of a user using the Broker Protocol to establish a PCoIP session where clients are connected to the datacenter through a LAN or VPN.



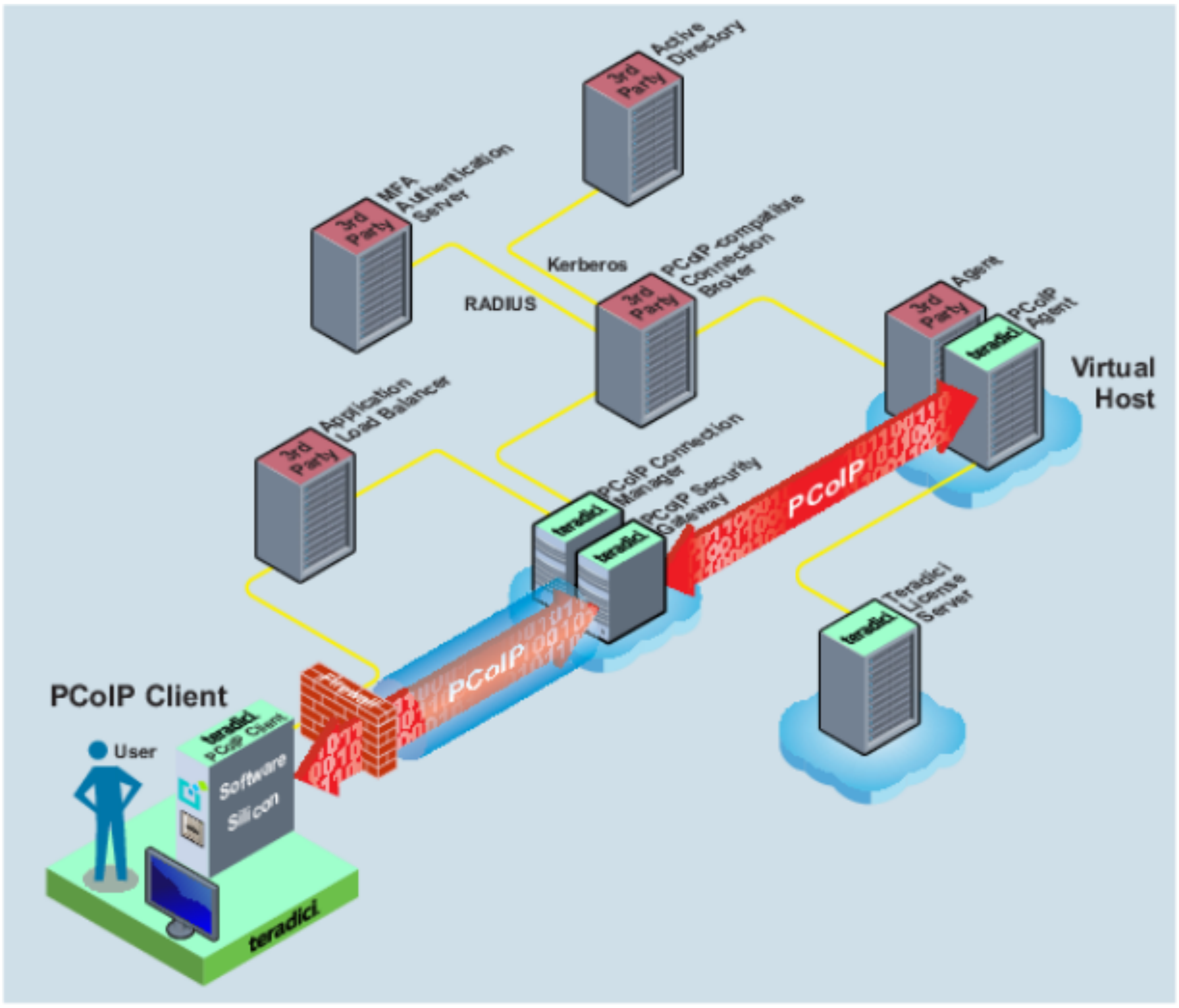
In this scenario, clients can directly communicate with the PCoIP agents, which eliminates the need for the PCoIP Security Gateway. In this case, the PCoIP Connection Manager would be deployed without a paired PCoIP Security Gateway, as shown in the diagram. The figure also shows an optional load balancer between the PCoIP client and the PCoIP Connection Managers.

When a deployment involves both internal and external deployments

In a typical deployment scenario involving both internal and external deployments, internal and external users may access the same data center by providing both a pool of external-facing connection managers paired with security gateways, and a pool of internal-facing connection managers without security gateways.

Multi-Factor Authentication Deployments

The broker is generally responsible for ensuring the connecting user has authorization to use the system and determining which resources the user should be presented with. With that principle, in a multi-factor authentication deployment, the broker is often responsible for performing the authentication, as shown in the following diagram.

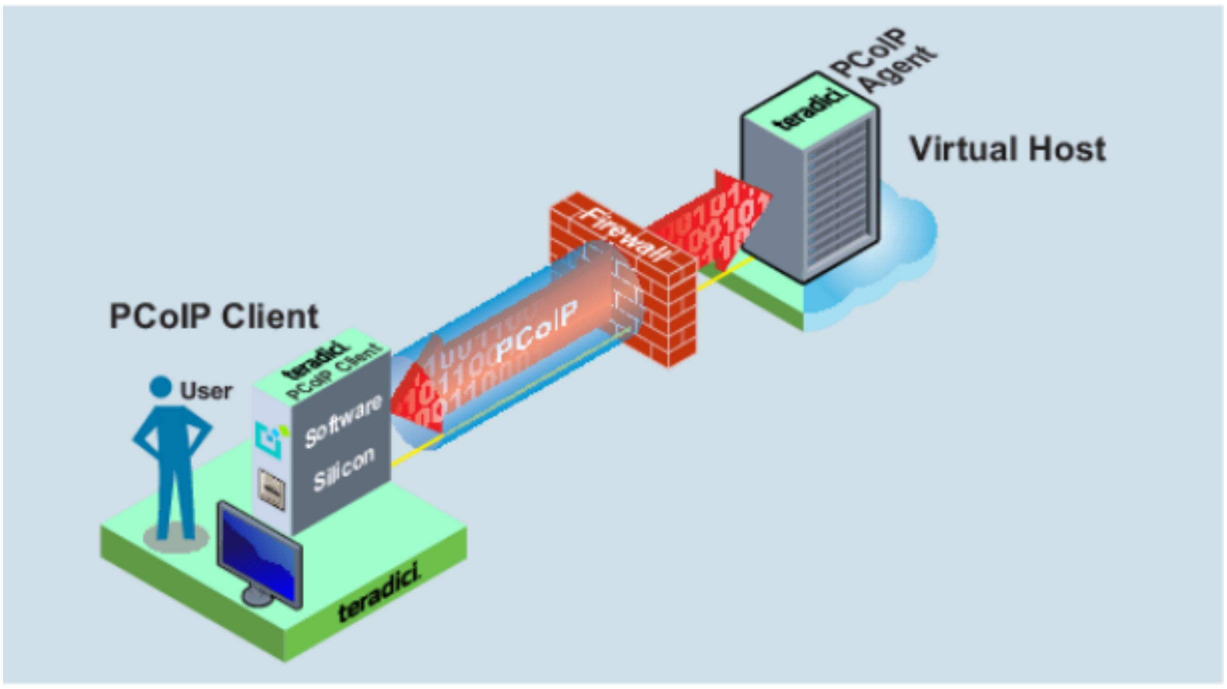

3rd Party Active Directory
3rd Party MFA Authentication Server
Kerberos
RADIUS
3rd Party PCoIP-compatible Connection Broker
3rd Party Agent
teradici PCoIP Agent
Virtual Host
3rd Party Application Load Balancer
teradici PCoIP Connection Manager
teradici PCoIP Security Gateway
PCoIP
Teradici License Server
PCoIP Client
User
teradici PCoIP Client
Software
Silicon
Firewall
teradici

This diagram shows an example broker which uses Kerberos and RADIUS communications to back-end authentication servers. The Broker Protocol and the Teradici Cloud Access Software do not prescribe any particular structure on the broker or how it authenticates users.

Alternately, gateway or authentication appliances may perform one phase of the user authentication before allowing access to the broker. These appliances, not shown, may be placed inline in the Broker Protocol flow with a function similar to that of the PCoIP Connection Manager.

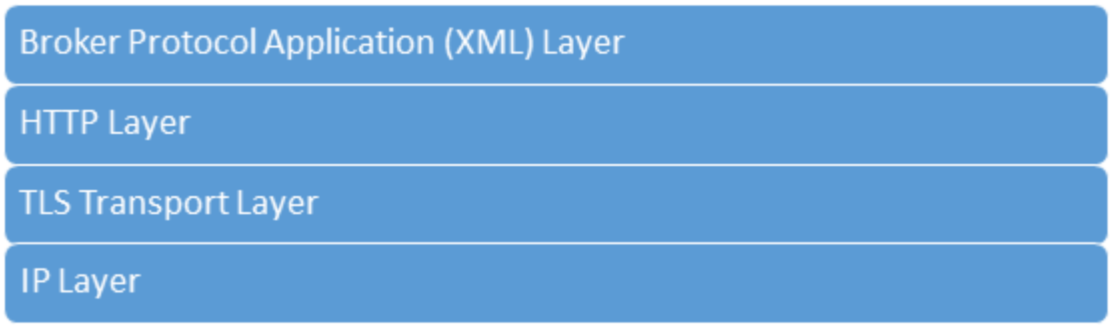
Direct-Connect Deployments

The Broker Protocol may also be used to establish a PCoIP session directly without using a connection broker. In the direct-connect deployment, the client uses the Broker Protocol to directly communicate with the remote resource to establish a PCoIP session. Direct-connect deployments are typically used in LAN-only scenarios. The following diagram illustrates a direct-connect deployment.



PCoIP Connection Broker Protocol Structure

Broker Protocol messages are XML messages embedded in the body-content of HTTP messages, using a strict client/server or API message flow where the PCoIP client is the 'client' and the broker is the 'server'.



The Broker Protocol uses HTTP encrypted by Transport Layer Security (TLS) as the transport mechanism to deliver messages. To ensure secure delivery of the Broker Protocol messages, specific aspects of the TLS and HTTP layers are specified.

To provide solution flexibility in a compatible way, most messages provide mechanisms for name-spaced optional variables to be tunneled between components.

Localization in the Broker Protocol is handled from the server end so the broker is responsible for creating messages in the languages it wishes to support.

Security Requirements

For security and interoperability, all sessions are secured over TLS 1.0 or above with a specified preferred cipher suite. Broker Protocol session metadata (cookie) handling is also specified to ensure uniform operation across the platform.

Broker Protocol HTTP Overview

All requests to and from the Broker Protocol are initiated by the HTTP client endpoint.

The Broker Protocol makes use of the following HTTP elements and functionality to transport the Broker Protocol messages:

- HTTP version: 1.1
- HTTP Requests: POST
- HTTP Responses: 200 OK and error responses
- https-URI: `https://<FQDN or ip-address>/pcoip-broker/xml`
- HTTP header content-type: application/xml charset=UTF-8
- Fixed-length or chunked HTTP payload
- Persistent and/or non-persistent HTTP connection
- X-Forwarded-For header
- Client-Log-Id header

As well, the Broker Protocol makes use of an HTTP cookie called JSESSIONID as per RFC 6265.

Broker Protocol Application (XML) Layer

Requests and responses are paired as much as possible with, for example, a client sending an `<example>` request and the server responding with an `<example-resp>` response. Each request-response pair is considered a message transaction. Only a single message transaction may be outstanding at any point in time for each Broker Protocol session.

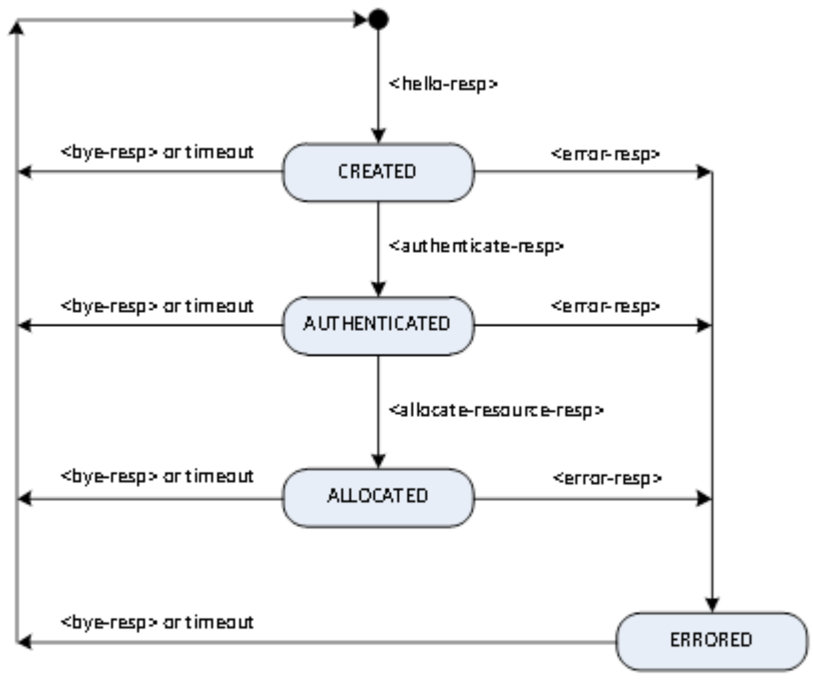
The Broker Protocol messages broadly fit into the following categories:

- Negotiate protocol version and capabilities (required)
- Authenticate the user (basic password required)
- Present the user with their inventory (required)
- Start a session (required)
- Perform operations on resources or machines (optional)
- Terminate the brokering session (optional)
- Broker Protocol application layer errors are indicated by sending an `<error-resp>` message towards the client.

Many flows and messages are optional. A standard flow that is required is a negotiate, authenticate, select and start session flow. See Overview of PCoIP Session Initiation Broker Protocol Messages provides an example of this flow.

Broker Protocol Session States

A Broker Protocol session is created when the `<hello>` message has been successfully processed and a `JSESSIONID` has been generated. The following diagramshows the state transition diagram for the Broker Protocol session.



State transition diagram for the Broker Protocol session

As shown in the preceding figure, the Broker Protocol session has four states:

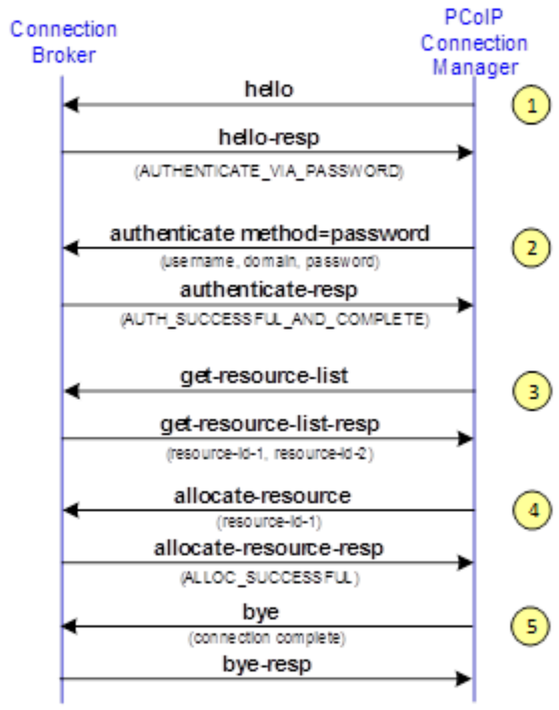
- CREATED: The session transitions into the `CREATED` state after successfully processing a `<hello>` request. A `JSESSIONID` value is generated when the session enters this state.
- AUTHENTICATED: The session transitions into the `AUTHENTICATED` state after successfully authenticating a user (after receiving an `<authenticate-resp>` message with `AUTH_SUCCESSFUL_AND_COMPLETE` result).

- ALLOCATED: The session transitions into the `ALLOCATED` state after successfully allocating a resource (after receiving an `<allocate-resource-resp>` message with `ALLOC_SUCCESSFUL` result).
- ERRORED: The session transitions into the `ERRORED` state after detecting an error and sending an `<error-resp>` message.

The Broker Protocol session terminates when a `<bye>` message has been processed or when the maximum session time elapses. The `JSESSIONID` is deleted/invalidated when the session terminates.

Overview of PCoIP Session Initiation Broker Protocol Messages

The following diagram shows the typical Broker Protocol messages exchanged between the PCoIP Connection Manager and a broker to initiate a PCoIP session.



Typical message transactions to launch a PCoIP session

The PCoIP client and PCoIP Connection Manager exchange an almost identical set of messages. The PCoIP client initiates the Broker Protocol session by sending a `<hello>` request to the PCoIP

Connection Manager. In this example, the PCoIP client authenticates the user using the username and password. Here are the steps to initiate/broker a PCoIP session:

1. The PCoIP Connection Manager, triggered by the client, initiates a Broker Protocol session by sending a `<hello>` request message. The connection broker responds with the `<hello-resp>` response message. The `<hello-resp>` message contains the `AUTHENTICATE_VIA_PASSWORD` method which tells the client to authenticate the user via the username and password.
2. The PCoIP Connection Manager sends an `<authenticate>` message and specifies "password" as the authentication method. The `<authenticate method="password">` message contains the username, domain, and password entered by the user. The connection broker responds with the `<authenticate-resp>` message and specifies `AUTH_SUCCESSFUL_AND_COMPLETE.`
3. The PCoIP Connection Manager sends the `<get-resource-list>` message to request the list of resources to which the authenticated user is entitled and the broker returns the list of the resources.
4. The PCoIP Connection Manager sends the `<allocate-resource>` message to launch a PCoIP session to the specified resource (desktop) and the connection broker responds with the `<allocate-resource-resp>` response messages and specifies `ALLOC_SUCCESSFUL` .

Example Broker Protocol Messages

The following sections provide example messages between the PCoIP Connection Manager and the connection broker for the flow shown in the preceding diagram. The parameters marked as 'xxxx' would need to be filled in during an actual message exchange.

hello message

```
POST /pcoip-broker/xml HTTP/1.1<CR><LF>
Host: <CR><LF>
User-Agent: xxxx<CR><LF>
Client-Log-Id: 4208fb66-e22a-11d1-a7d7-00a0c982c00d<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
```

```
<hello>
   <client-info>
      <product-name>Teradici Windows Soft Client</product-name>
      <product-version>3.1.5</product-version>
      <platform>Windows 7 64-bit</platform>
      <locale>en_US</locale>
      <hostname>my-client.teradici.local</hostname>
      <serial-number>123456</serial-number>
      <device-name>some-name</device-name>
   </client-info>
   <pcm-info>
      <product-name>Teradici PCoIP Connection Manager</product-name>
      <product-version>1.0.0.12345</product-version>
      <platform>Ubuntu 14.4 64-bit</platform>
      <ip-address>10.0.136.93</ip-address>
      <hostname>cm1.example.com</hostname>
   </pcm-info>
</hello>
</pcoip-broker>
```

hello-resp message

```
HTTP/1.1 200 OK<CR><LF>
Date: Sun. 1 Jul 2012 16:40:00 GMT<CR><LF>
Set-Cookie: JSESSIONID=aabbccddeeff00112233445566778899; HttpOnly; Secure<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<hello-resp>
   <brokers-info>
      <broker-info>
         <product-name>XYZ Broker</product-name>
         <product-version>5.1.0</product-version>
         <platform>Linux Ubuntu 10.4 64-bit</platform>
         <locale>en_US</locale>
         <ip-address>10.0.136.26</ip-address>
         <hostname>broker1.teradici.com</hostname>
      </broker-info>
   </brokers-info>
   <next-authentication>
      <authentication-methods>
         <method>AUTHENTICATE_VIA_PASSWORD</method>
      </authentication-methods>
      <domains>
```

```
          <domain>domain-1</domain>
          <domain>domain-2</domain>
       </domains>
    </next-authentication>
</hello-resp>
</pcoip-broker>
```

authenticate message

```
POST /pcoip-broker/xml HTTP/1.1<CR><LF>
Host: <CR><LF>
User-Agent: xxxx<CR><LF>
Cookie: JSESSIONID=aabbccddeeff00112233445566778899<CR><LF>
Client-Log-Id: 4208fb66-e22a-11d1-a7d7-00a0c982c00d<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<authenticate method="password" >
   <username>username</username>
   <password>password</password>
   <domain>domain</domain>
</authenticate>
</pcoip-broker>
```

authenticate-resp message

```
HTTP/1.1 200 OK<CR><LF>
Date: Sun. 1 Jul 2012 16:40:00 GMT<CR><LF>
Set-Cookie: JSESSIONID=aabbccddeeff00112233445566778899; HttpOnly; Secure<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<authenticate-resp method="password">
   <result>
      <result-id>AUTH_SUCCESSFUL_AND_COMPLETE</result-id>
      <result-str>Authentication completed successfully</result-str>
   </result>
</authenticate-resp>
</pcoip-broker>
```

get-resource-list message

```
POST /pcoip-broker/xml HTTP/1.1<CR><LF>
Host: <CR><LF>
User-Agent: xxxx<CR><LF>
Cookie: JSESSIONID=aabbccddeeff00112233445566778899<CR><LF>
Client-Log-Id: 4208fb66-e22a-11d1-a7d7-00a0c982c00d<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<get-resource-list>
</get-resource-list>
</pcoip-broker>
```

get-resource-list-resp message

```
HTTP/1.1 200 OK<CR><LF>
Date: Sun. 1 Jul 2012 16:40:00 GMT<CR><LF>
Set-Cookie: JSESSIONID=aabbccddeeff00112233445566778899; HttpOnly; Secure<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<get-resource-list-resp>
   <result>
      <result-id>LIST_SUCCESSFUL</result-id>
      <result-str>The user is entitled to resources</result-str>
   </result>
   <resource>
      <resource-name>My Desktop</resource-name>
      <resource-id>abcdef0123456789</resource-id>
      <resource-type session-type="VDI">DESKTOP</resource-type>
      <resource-state>UNKNOWN</resource-state>
      <protocols>
         <protocol is-default="true">PCOIP</protocol>
      </protocols>
   </resource>
   <resource>
      <resource-name>My Session Desktop</resource-name>
      <resource-id>abcdef9876543210</resource-id>
      <resource-type session-type="RDS">DESKTOP</resource-type>
```

```
        <resource-state>UNKNOWN</resource-state>
        <protocols>
            <protocol is-default="true">PCOIP</protocol>
        </protocols>
    </resource>
</get-resource-list-resp>
</pcoip-broker>
```

allocate-resource message

```
POST /pcoip-broker/xml HTTP/1.1<CR><LF>
Host: <CR><LF>
User-Agent: xxxx<CR><LF>
Cookie: JSESSIONID=aabbccddeeff00112233445566778899<CR><LF>
Client-Log-Id: 4208fb66-e22a-11d1-a7d7-00a0c982c00d<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<allocate-resource>
    <resource-id>abcdef0123456789</resource-id>
    <protocol>PCOIP</protocol>
    <client-info>
        <time-zone-windows>Pacific Standard Time</time-zone-windows>
        <ip-address>192.168.0.57</ip-address>
        <mac-address>00:67:a8:2d:84:7e</mac-address>
    </client-info>
</allocate-resource>
</pcoip-broker>
```

allocate-resource-resp message

```
HTTP/1.1 200 OK<CR><LF>
Date: Sun. 1 Jul 2012 16:40:00 GMT<CR><LF>
Set-Cookie: JSESSIONID=aabbccddeeff00112233445566778899; HttpOnly; Secure<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<allocate-resource-resp>
    <result>
        <result-id>ALLOC_SUCCESSFUL</result-id>
```

```
      <result-str>Successfully allocated resource</result-str>
   </result>
   <target>
      <ip-address>192.168.1.56</ip-address>
      <hostname>mydesktop.teradici.com</hostname>
   </target>
   <resource-id>desktop-id</resource-id>
   <protocol>PCOIP</protocol>
</allocate-resource-resp>
</pcoip-broker>
```

bye message

This is an optional message. The PCoIP client may send this message to end the broker session.

```
POST /pcoip-broker/xml HTTP/1.1<CR><LF>
Host: <CR><LF>
User-Agent: xxxx<CR><LF>
Cookie: JSESSIONID=aabbccddeeff00112233445566778899<CR><LF>
Client-Log-Id: 4208fb66-e22a-11d1-a7d7-00a0c982c00d<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<bye>
   <reason>Connection complete</reason>
</bye>
</pcoip-broker>
```

bye-resp message

```
HTTP/1.1 200 OK<CR><LF>
Date: Sun. 1 Jul 2012 16:40:00 GMT<CR><LF>
Set-Cookie: JSESSIONID=aabbccddeeff00112233445566778899; HttpOnly; Secure<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<bye-resp>
</bye-resp>
</pcoip-broker>
```

Errors

If an error occurs at the application layer, then the broker is responsible for generating an `<error-resp>` .

The following example request is a malformed request. The `<client-info>` element is not closed.

```
POST /pcoip-broker/xml HTTP/1.1<CR><LF>
Host: <CR><LF>
User-Agent: xxxx<CR><LF>
Client-Log-Id: 4208fb66-e22a-11d1-a7d7-00a0c982c00d<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
   <hello>
      <client-info>
   </hello>
</pcoip-broker>
```

In this case, the broker would create an `<error-resp>` message. The following is an example of an appropriate message.

```
HTTP/1.1 200 OK<CR><LF>
Date: Sun. 1 Jul 2012 16:40:00 GMT<CR><LF>
Set-Cookie: JSESSIONID=aabbccddeeff00112233445566778899; HttpOnly; Secure<CR><LF>
Client-Log-Id: 4208fb66-e22a-11d1-a7d7-00a0c982c00d<CR><LF>
Content-Type: application/xml charset=UTF-8<CR><LF>
Content-Length: xxxx<CR><LF>
<CR><LF>
<?xml version="1.0" encoding="UTF-8"?>
<pcoip-broker version="2.1">
<error-resp>
   <result>
      <result-id>ERR_INVALID_MSG_FORMAT</result-id>
      <result-str>Invalid XML message format</result-str>
   </result>

   <detected-by>BROKER</detected-by>

   <!-- Optional error details -->
```

```
    <err-detail>failed to close ‘client-info’ element </err-detail>


</error-resp>
</pcoip-broker>
```