Diameter Endpoint

Feature Description

You can enable the Diameter endpoint to dynamically create pods on a designated node or host. This feature might be a requirement when you want to ensure that the nodes are meeting specific security and regulatory parameters, or the node is closer to the datacenter in terms of geographical proximity. The node affinity determines the node where CPC creates the Diameter endpoint pods, which are based on the affinity towards a node or group of nodes. Node affinity is a set of rules that allows you to define the custom labels on nodes and specify the label selectors within the pods. Based on these rules, the scheduler determines the location where the pod can be placed.

Note If you do not specify a node, then the Kubernetes scheduler determines the node where the Diameter endpoint creates a pod.

CPC supports both IPv4 and IPv6 connectivity on its external interfaces/endpoints (inbound and outbound).

Configuring the Node for the Diameter Endpoint Pod

This section describes how to specify the node or host where the Diameter endpoint must spawn the pod.

Note Configuration changes to the diameter endpoint cause the endpoint to restart automatically. Cisco recommends making such changes only within the maintenance window.

To specify the node where you want Diameter endpoint to spawn the pod, use the following configuration:

```
config
   diameter group diameter_group_name
```

```
mode server server_name
stack stack_name
  application application_name
  bind-ip ipv4 host_address


  fqdn fqdn_address
  realm realm_address
  node-host node_host_address
  end
```

NOTES:

- **diameter group** *diameter_group_name*—Specify the Diameter group name.
- **mode server** *server_name*—Specify the server name that operates as the mode server.
- **stack** *stack_name*—Specify the stack name.
- **application** *application_name*—Specify the application name.
- **bind-ip** *host_address*—Specify the host address to bind the stack.
- **fqdn** *fqdn_address*—Specify the FQDN address.
- **realm** *realm_address*—Specify the realm address.
- **node-host** *node_host_address*—Specify the host IP address of the node.

Sample Configuration

The following is a sample configuration of the node configuration.

```
mode server
  stack cicdsite
  application rx
  bind-ip 192.0.2.18
  realm cisco.com
  node-host for-node-2a-worker39e1587354h
  exit
```

RX-AAR messages handling during site failover

Table 1: Feature History

Feature Name	Release Information	Description
Cross-Site Session Handling for Distributed CPC		This feature enables seamless cross-site session handling within a distributed Converged Policy (CPC) system. The feature enables CPC nodes to process Rx session-related messages for sessions created at other sites. When the system is configured to accept Diameter messages with unknown destination hostnames, session processing continues during site failovers, load balancing, or dynamic routing scenarios. **Command introduced:** **diameter properties**

Configuring diameter properties for cross-site session handling

This procedure describes how to enable CPC nodes to accept Diameter messages with unknown Destination-Host values, which is essential for cross-site session handling in a distributed CPC environment.

Before you begin

Ensure you meet these important prerequisites before you start with the Ops-Center configuration:

- Ensure you have administrative access to the system's Configuration Management Framework.
- Understand the impact of this configuration change on your distributed CPC system.

Follow these steps to configure Diameter properties for cross-site session handling.

Procedure

Step 1 Enter the Ops-Center to specify the Diameter properties.

Example:

```
PCF(config)# diameter properties
```

Step 2 Change the `jdiameter.accept.unknown_desthost` property value from false to True.

a) **false**: The Diameter protocol stack rejects messages where the Destination-Host does not match the hostname of the receiving CPC node, returning error code 3002 (DIAMETER_UNKNOWN_PEER).

b) **true**: The Diameter protocol stack accepts messages with unknown Destination-Host values and bypasses strict hostname validation. This allows the CPC to process the message locally or retrieve session state.

Example:

```
PCF(config)# diameter properties jdiameter.accept.unknown_desthost value true
Tue Sep  9  06:21:31.289 UTC+00:00
PCF(config-properties-jdiameter.accept.unknown_desthost)# commit
Tue Sep  9  06:21:34.615 UTC+00:00
```

After applying the changes, verify that the `jdiameter.accept.unknown_desthost` property is successfully set to `true` on the CPC nodes. You can verify the property by querying the configuration or by observing system behavior. Applying this property config redeploys the diameter pods.

Diameter message level overload handling

Feature description

Table 2: Feature description

Feature Name	Release Information	Description
Diameter message-level overload handling	2026.03.0	The Diameter message-level overload handling is an enhancement that provides ingress protection for the Converged Policy (CPC) Diameter endpoint. It mitigates traffic bursts that cause latency, request timeouts, and instability by enforcing traffic limits at the pod level before requests reach the engine. This ensures system reliability and protects resources from excessive traffic.

This enhancement enforces traffic limits at the `cps-diameter-ep` pod level for the Converged Policy (CPC) Diameter endpoint. It supports ingress traffic for Gx, Rx, and Sy interfaces. It mitigates traffic bursts that cause latency, request timeouts, and instability by enforcing traffic limits at the pod level before requests reach the `pcf-engine`.

- Global overload protection: Applies at the pod level, affecting all stacks within the `cps-diameter-ep` pod.
- Stack-specific overload protection: Applies to a specific stack within the `cps-diameter-ep` pod.

Applying overload protection globally and at the stack level for specific Diameter procedures protects system resources, prioritizes legitimate traffic, and minimizes the processing load on downstream components.

Configure global overload protection

Before you begin

Before you configure this feature, ensure that you meet these requirements:

- Ensure you have administrative access to the Converged Policy Ops-Center.
- Verify that the `cps-diameter-ep` pod is running and healthy.
- Confirm that you have the necessary permissions to modify the `advance-tuning` and `diameter` configuration hierarchies.

Use this procedure to configure global traffic limits and rejection actions for Diameter messages.

Procedure

Step 1 Log in to the Converged Policy Ops-Center and enter `config` mode.

Step 2 Set the global ingress request limit.

```
advance-tuning overload-control diameter global limits max-requests-per-sec <value>
```

Step 3 Define the global action for rejected requests.

```
advance-tuning overload-control diameter global action throttle-action <reject | drop>
```

Step 4 Specify the Diameter result code for rejected requests.

```
advance-tuning overload-control diameter global action threshold-reject-code <code_value>
```

Step 5 Configure message-specific threshold actions.

```
advance-tuning overload-control diameter global action threshold-action <command_name>
discard-action <reject | drop>
threshold-count <value>
exit
```

Step 6 `commit` the changes.

Configure stack-specific overload protection

Use this procedure to configure traffic limits and rules for specific Diameter stacks.

Procedure

Step 1 Enter `config` mode

Step 2 Define the stack-specific transactions per second (TPS) limit.

```
diameter group <group_name> stack <stack_name> overload-control limits max-requests-per-sec <value>
```

Step 3 Configure overload rules for the stack.

```
diameter group <group_name> stack <stack_name> overload-control rule <rule_name>
threshold-tps <value>
discard-action <reject | drop>
command-code [ <code_list> ]
request-type [ <type_list> ]
exit
```

Step 4 `commit` the changes.

Verify and troubleshoot configurations

Use this procedure to verify configurations and to troubleshoot potential issues.

1. Run the `show running-config` command in the Converged Policy Ops-Center to ensure the output displays the intended global and stack-specific parameters.
2. Monitor the `cps-diameter-ep` Prometheus counters to confirm that the system tracks throttled, threshold-exceeded, and within-threshold requests

The `cps-diameter-ep` pod evaluates inbound Diameter requests against the defined global and stack-specific limits. The system rejects or drops requests exceeding these thresholds to prevent them from reaching the `pcf-engine`.

Troubleshoot

- Verify the syntax of the `advance-tuning` or `diameter` commands by using the `show configuration` command in the Converged Policy OpsCenter.
- Ensure that the `threshold-tps` or `max-requests-per-sec` values are set appropriately for the expected traffic volume.
- Check the Prometheus counters to determine if the traffic hits the global ingress limit before reaching stack-specific rules, as global limits take precedence.
- Verify the `DiameterServerStackListener` logs for runtime errors or rule-based throttling events:

  ```
  kubectl logs <cps-diameter-ep-pod-name>
  ```