API Connect Gateway setup

1. In the search field, enter `api`.
2. From the search results, click **API Connect gateway service**.
3. Set the administrative state of the configuration.
4. In the **Comments** field, enter a brief, descriptive summary for the configuration.
5. Define the host and port through which API Connect connects to manage the configuration of the gateway service.
6. Specify the local IP address or a host alias.The default value is 0.0.0.0, which is all active IP4 addresses.
7. Specify the local TCP port.The default value is 3000. Beyond this port, the service uses four more consecutive ports. Therefore, ports 3000, 3001, 3002, 3003, and 3004 are in use.
8. Optional: When API Connect requires a secure connection, specify the TLS client profile.
9. Optional: Define the URL to retrieve JWKS data to validate the JWT token from API Manager. When you specify a URL, specify whether transactions fail when JWT validation fails.
10. Optional: When the DataPower Gateway requires a secure connection, specify the TLS server profile.
11. Define the host and port for the HTTPS handler that the API gateway accepts client transactions.
12. Specify the IP address or a host alias for the handler.The default value is 0.0.0.0, which is all active IP4 addresses.
13. Specify the port for the handler.The default value is 9443.
14. Specify whether to enable compatibility with API gateway version 5.
15. When V5 compatibility is enabled, define the following properties.
16. Specify the gateway-peering instance to synchronize distributed state and configuration data.
17. Specify the peer group mode to use for the SLM policy.**When unicast mode**Specify the IP unicast address or select a host alias.**When multicast mode**Specify the IP multicast configuration.
18. When V5 compatibility is not enabled, edit the `default` gateway-peering manager.

19. Specify the gateway-peering instance that synchronizes distributed state and configuration data.
20. Specify the gateway-peering instance that manages rate-limiting data.
21. Specify the gateway-peering instance that manages subscriber data.
22. Optional: Specify user-defined policies to advertise to API Connect.
23. Optional: When V5 compatibility is disabled, specify user-defined policies to advertise to API Connect.
24. Optional: Click the **API Manager proxy** tab to specify a proxy server to connect to API Manager.
25. Specify to enable the API Manager proxy.
26. Specify the hostname or IP address of the proxy to connect to the API Manager endpoint.
27. Specify the listening port on the proxy to connect to the API Manager endpoint.
28. Click **Apply** to save changes to the running configuration.
29. Click **Save** to save changes to the persisted configuration.

API Request Processing

The API gateway processes a request by interacting with the API context. For each transaction, a context is prepared before processing starts and is continuously updated during the transaction.

After an API collection is matched, the following processing occurs.

1. The API gateway triggers the API processing rule of the collection to match the target API.
2. The API gateway enforces the constraints that the target API operation defines.
3. The API gateway prepares the context for the API operation.
4. The API gateway runs the assembly rule of the API to respond to the request.

5. The API gateway uses an API processing rule to process the request. The API gateway achieves each step of the process through running the corresponding processing action. By default, the processing rule contains the following sequence of actions. You can add or remove these preflow actions or change the processing order. For more information, see Configuring an API rule.
6. The API gateway runs the API routing action to match the target API to call. When no API is matched, the request is rejected.
7. When CORS is enabled for an API, the API gateway runs the CORS API action to handle cross-origin resource sharing (CORS) requests for the API.
8. The API gateway runs the client identification API action to examine the API key credentials. These credentials are carried in the API request and match the API plan through which the target API is made available to the client.
9. The API gateway runs the rate limit API action to enforce the rate limit scheme that is configured for the matching API plan. When the rate limit is reached, the request is rejected.
10. The API gateway runs the security API action to run the security checks that the target API and operation require. When the security requirement is not fulfilled, the request is rejected.
11. The API gateway runs the context API action to populate the context variables for the assembly to access or manipulate.
12. The API gateway runs the execute API action to run the assembly of the target API.
13. The API gateway runs the result API action to prepare the final response to the client based on the result from the execute API action.

14. When error occurs during the API request processing, the API gateway runs the API error rule to handle the errors. A default API error rule comprises only the result API action.

API Gateway configuration

1. In the search field, enter `gateway`.
2. From the search results, click **API gateway**.
3. Click **Add**.
4. Define the basic properties - Name, administrative state, and comments.
5. Specify the protocol handlers to receive API requests.
6. Specify the maximum intra-transaction timeout in seconds for API gateway to client connections.This value is the maximum idle time to allow in a transaction on the API gateway to client connection. This timer monitors idle time in the data transfer process. If the specified idle time is exceeded, the connection is torn down. Enter a value in the range 1 - 86400. The default value is 120.
7. Specify the maximum inter-transaction timeout in seconds for API gateway to client connections.This value is the maximum idle time to allow between the completion of a TCP transaction and the initiation of a new TCP transaction. If the specified idle time is exceeded, the connection is torn down. Enter a value in the range 0 - 86400. The default value is 180. A value of 0 disables persistent connections.
8. Specify the list of API collections to include.
9. Optional: Specify whether to share the rate limit data among all APIs under an application.
10. Optional: Define the management of stylesheets.
11. Specify the URL refresh policy that defines the rules to refresh stylesheets in the stylesheet cache.

12. The capacity of the stylesheets cache, where capacity is reached based on maximum size or maximum count.
13. Whether to cache stylesheets with their SHA1 message digest value.
14. Whether multiple calls from the same stylesheet return the same result.
15. Optional: Specify the load balancer group, or server pool, to provide redundancy among target resources.
16. Optional: Specify the LDAP connection pool to connect to an LDAP server.
17. Optional: Click the **Rate limiting** tab to add assembly limits. Follow these steps to add assembly burst limits, assembly rate limits, or assembly count limits.
18. Click **Add**.
19. Specify the name for the limit scheme.Assembly limits defined in different configurations, including an API gateway, API collection, or API plan, must have unique names. If two limits with the same name are in an assembly and both are applied, then information about only one is included in the limit response headers.
20. Specify the maximum rate to allow.
21. Optional: Specify the time interval.
22. Optional: Specify the time unit.
23. Optional: Specify to enable hard limit.
24. Optional: Specify to use the local cache first to enforce the limit. In peer group mode, the local cache first can prevent transaction delays if communication problems arise across the peer group. However, the transaction count is less precise when this setting is enabled.
25. Optional: Specify to enforce the limit on the client rather than on an internal component. When exceeded, client limits return a 429 error code, and nonclient limits return

a `503` error code. When set to **off**, limit information is not included in the response header.

26. Optional: Specify to include the API name in the limit key.
27. Optional: Specify to include the application ID in the limit key.
28. Optional: Specify to include the client ID in the limit key.
29. Optional: Specify the dynamic value string for the limit, which contains one or more context variables.The dynamic value makes it possible to use a context variable to enforce the limit based on parameters that are not defined in the scheme. The context variable can be set in a GatewayScript action and then included in the dynamic value. The default value is an empty string.The following example uses the `context` object in a GatewayScript action to add the `my.server` variable to the API context. You can then provide the `$(my.server)` variable that resolves to the `server34` server.context.set("my.server", "server34")
30. Optional: Specify a JSONata expression that assigns a weight value to the transaction.You use a subset of JSONata notation to define the expression. For more information, see JSONata and assembly actions.For each API call, the applied limit is the value that the weight expression computes. The default setting is `1`. If the weight expression evaluates to a value that is less than or equal to `0`, it is set to `1`. An empty string results in an error.
31. For an assembly count limit, specify whether to autodecrement the count limit for each transaction. When enabled, the count limit is decremented at the end of the assembly. The decrement amount is the total incremented amount minus any decrement actions in the rate limit assembly action. You do not need to define decrement actions explicitly unless you want the decrements to occur at specific points in the assembly. When set to **off**, you

must explicitly define a decrement action for each corresponding increment action. For more information, see Count limits guidelines.

32. Click **Apply** to add the entry to the list.
33. Repeat this step to add another limit scheme.
34. Optional: Click the **Document cache** tab to enable document caching, define capacity, and current write behavior.
35. Specify the maximum number of documents that is allowed in the cache. By default the value is 0, which disables caching.
36. Specify the maximum size for document cache in bytes. Regardless of size, no document that is greater than 1073741824 bytes is cached. This restriction applies even when the cache has available space.
37. Specify the maximum number of concurrent write requests to create documents or refresh expired documents in the document cache. Enter a value in the range 1 - 32768. The default value is 32768. After the maximum number is reached, requests are forwarded to the target server. Responses to forwarded requests are not written to the cache.
38. Optional: Click the **Connection policy** tab to define proxy policies. For each policy, complete the following steps.
39. Click **Add**.
40. Specify a shell-style match pattern that defines a URL set.
41. Whether to forward the request to the remote server.
42. Specify the name or IP address and port of the remote server.
43. Specify credentials to authenticate with the remote server.
44. Click **Apply** to add the entry to the list.
45. Click the **Document cache policy** tab to define document cache policies, which define how documents are cached

and serve requests. For each policy, complete the following steps.

46. Click **Add**.
47. Specify the shell-style match pattern that identifies documents.
48. Select the type of the caching policy.**Fixed**Caches documents that match the expression. The TTL specifies the validity period.**No-cache**Does not cache documents that match the expression.**Protocol-based**Caches documents that match the expression in compliance with HTTP caching rules.
49. When the policy type is fixed, specify the validity period in seconds for a document in the cache.
50. Specify the priority of a document to add to or remove from the cache. The greater the value, the higher its priority.
51. Optional: When the protocol type is fixed or protocol-based, define the following behaviors.
52. Select the data grid for document caching.
53. Whether to cache the response to an HTTP GET request.
54. Whether the document cache operates on HTTP cache validation requests from the client.
55. Whether to return expired, or stale, content.
56. Whether to invalidate the document cache entry when a PUT, POST, or DELETE request matches the entry.

57. Optional: When the protocol type is fixed, whether to cache the response to POST and PUT requests.
58. Click **Apply** to add the entry to the list.
59. Optional: Click the **Scheduled processing rule** tab to define the schedule to run specific processing rules.
60. Optional: Click the **OpenTelemetry** tab to define OpenTelemetry integration.An instance of the

OpenTelemetry object defines an integration point for OpenTelemetry (OTel), which is an observability framework. Each instance of an OpenTelemetry object defines an exporter, a sampler, and a vector of resource attributes to use for OTel integration. For more information, see OpenTelemetry integration.To augment resource attributes that the OpenTelemetry instance associated with the API gateway defines, use the **OpenTelemetry resource attributes** area to define the augmentation vector of OpenTelemetry resource attributes. For more information, see OpenTelemetry resources.The following table shows the logic that creates the operational resources attributes for OTel integration.

61. Table 1. Logic for OTel resource attributes

62. OpenTelemetry instance	63. Augmentation vector	64. Operational resource attribute
65. Yes	66. No	67. OpenTelemetry instance
68. No	69. Yes	70. Augmentation v
71. Yes	72. Yes	73. Augmentation v

74.

75. Associate an instance of the OpenTelemetry object as the integration point.
76. Define the vector of OpenTelemetry resource attributes to augment the vector in the associated OpenTelemetry instance.
77. Click **Apply** to save changes to the running configuration.
78. Click **Save** to save changes to the persisted configuration.