



WSO2 API Manager 3.2.0 Developer Fundamentals

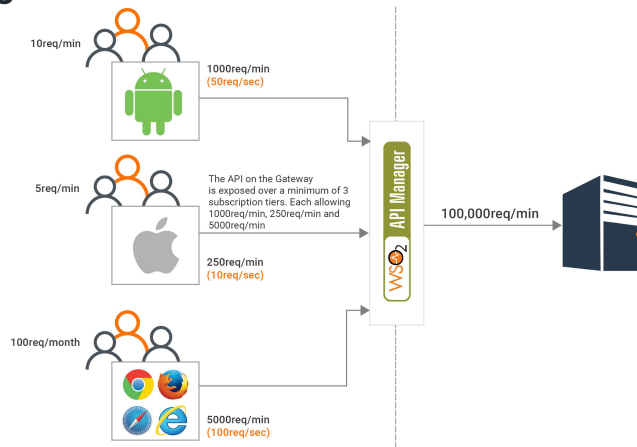
Rate Limiting Policies



WSO2 Training

CC BY 4.0

Rate Limiting Policies

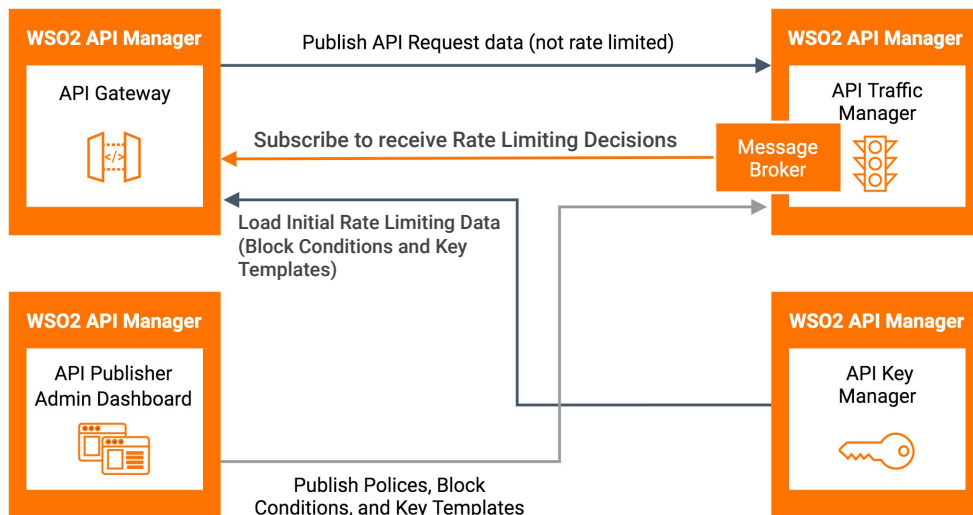


Link - [Working with Rate Limiting](#)

Why is Rate limiting important ?

- Rate limiting is a key functionality of API Management
- API backends can't serve unlimited requests
- Rate limiting plays an important part in monetizing an API
- Ensuring that your business APIs can be exposed to the public with a regulation and ensure that each user gets his fair share
- Shaping the traffic as your business changes
- Allow more traffic to your business partners and distribute the rest of it among other users
- Control Unusual API invocation of End Users

Introducing Central Policy Server : Traffic Manager



The Traffic Manager processes data of each API request and makes rate limiting decisions based on the applicability of available rate limiting policies.

Rate limiting decisions are made by the Siddhi runtime and published to the JMS topic.

The gateways subscribed to this JMS topic get instantly notified about the decisions.

Other features :

- Includes powerful Siddhi runtime based decision engine
- Extensible and flexible to define advanced rules based on API properties such as headers, users, JWT Claims, etc...
- Flexibility to design rate limiting policies based on both request count and bandwidth
- Supporting Instantaneous request blocking based on User, IP Address, Application and API

Maximum Backend Throughput Limit

The screenshot displays the 'Runtime Configurations' page for the 'PizzaShackAPI :1.0.0' in a state of 'PUBLISHED'. The page is divided into two main sections: 'Request' and 'Backend'. The 'Request' section includes settings for 'Transport Level Security', 'Application Level Security', 'CORS Configuration', 'Schema Validation', and 'Message Mediation'. The 'Backend' section, which is highlighted with a red box, contains the 'Backend Throughput' configuration. Under 'Maximum Throughput', the 'Specify' radio button is selected. Below this, there are two input fields: 'Max Production TPS' and 'Max Sandbox TPS', both currently set to 'TPS'. A note at the bottom of the 'Backend' section states 'Maximum backend transactions per second in integers'. The page also includes a 'Go To' button, a 'View in Dev Portal' link, and 'Create New Version' and 'Delete' buttons. The status 'Last updated: 12 minutes ago' is shown in the top right corner.

Link - [Maximum Backend Throughput Limit](#)

The Maximum backend throughput setting limits the total number of calls a particular API in the API Manager is allowed to make to the backend.

We can set the maximum throughput under rate limiting settings when publishing the API through UI.

Alternatively, you can go to the synapse configuration of the API in `<APIM_HOME>/repository/deployment/server/synapse-configs` and specify the maximum backend throughput there*.

*** Note: The changes made in the synapse file will be overridden if the API is republished.**

Levels of Rate Limiting

Subscription Level Rate Limiting
(API Publisher)

Subscription Level Rate Limiting
(API Subscriber)

Application Level Rate Limiting
(Application Developer)

Advanced Level Rate Limiting
(API Publisher)

☒ Request Count ☐ Request Bandwidth

Request Count *

Number of requests allowed

Unit Time

Time configuration

Minute(s) ▼



Why do we need Rate Limiting?

- To protect your APIs from common types of security attacks such as denial of service (DOS)
- To regulate traffic according to infrastructure availability
- To make an API, application or a resource available to a consumer at different levels of service, usually for monetization purpose

It is possible to rate limit requests for each tier based on the request count per unit time or the amount of data (bandwidth) transferred through the Gateway per unit time as shown in the last diagram.

Subscription Rate Limiting Policies (API Publisher)

Selecting the subscription tier/s when publishing the API

- Bronze: 1000 requests per minute
- Silver: 2000 requests per minute
- Gold: 5000 requests per minute
- Unlimited: Allows unlimited access

The screenshot shows the 'Business Plans' configuration page for the 'PizzaShackAPI :1.0.0'. The left sidebar contains a navigation menu with options: Overview, Design Configurations, Runtime Configurations, Resources, Endpoints, Subscriptions (highlighted), Lifecycle, and API Definition. The main content area has a 'BACK TO APIs' link and the API name 'PizzaShackAPI :1.0.0' with a 'PUBLISHED' state and 'Created by: admin@carbon.super'. Below this, the 'Business Plans' section is titled 'Attach business plans to API'. It lists four plans with checkboxes: Bronze (1000 requests per minute), Gold (5000 requests per minute), Silver (2000 requests per minute), and Unlimited (unlimited requests). The 'Unlimited' plan is selected. At the bottom are 'SAVE' and 'CANCEL' buttons.

Plan	Rate Limit	Selected
Bronze	Allows 1000 requests per minute	<input type="checkbox"/>
Gold	Allows 5000 requests per minute	<input type="checkbox"/>
Silver	Allows 2000 requests per minute	<input type="checkbox"/>
Unlimited	Allows unlimited requests	<input checked="" type="checkbox"/>

We can add this subscription level rate limiting to the API when managing the API to publish in the Publisher portal.

Subscription Rate Limiting Policies (API Publisher)

Rate Limiting (Burst Control)

- Define combined tiers
- Control the usage of APIs within smaller time durations
- Protect the backend from sudden request bursts



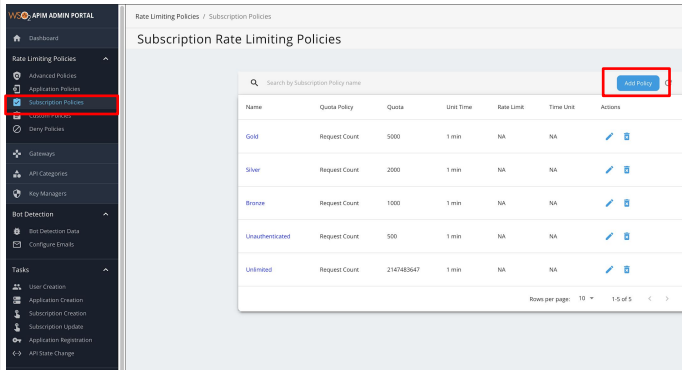
With rate limiting, you can define tiers with a combination of, for example, a 1000 requests per day and 10 requests per second. Users are then rate limited at two layers. Enforcing a rate limit protects the backend from sudden request bursts and controls the usage at a subscription and API level.

For instance, if there's a subscription level policy enforced over a long period, you may not want users to consume the entire quota within a short time span. Sudden spikes in usage or attacks from users can also be handled via rate limiting. You can define a spike arrest policy when the subscription level tier is created.

For each subscription level throttle key, a WS policy is created on demand. The request count is calculated and rate limiting occurs at the node level. If you are using a clustered deployment, the counters are replicated across the cluster.

Adding a New Subscription Rate Limiting Policy

Admin Portal : <https://localhost:9443/admin>



The 'Add Policy' form is shown with the following sections:

- General Details:** Includes fields for Name (set to 'Platinum'), Allowed requests per min (set to 1000), and Description.
- Quota Limits:** Includes fields for Request Count (set to 10000), Number of requests allowed (set to 1), and Unit Time.
- Burst Control (Rate Limiting):** Includes fields for Request Rate (set to 10) and Number of requests for burst control.
- GraphQL:** Includes fields for Max Complexity and Max Depth.
- Policy Flags:** Includes a Billing Plan dropdown (set to 'New') and a Stop On Quota Reach checkbox (checked).
- Custom Attributes:** Includes a dropdown for custom attributes (set to 'Add Custom Attribute').
- Permissions:** Includes a dropdown for permissions (set to 'Inherit everywhere') and radio buttons for Allow and Deny (Allow is selected).

Link - [Subscription Rate Limiting Policy](#)

We can specify the bandwidth per unit time by adding a new subscription level tier. This can be done through the Admin Portal (Rate Limiting Policies → Subscription Policies → Add Policy)

To make changes in the rate limiting configurations, set the enable_advanced_throttling parameter in the <APIM_HOME>/repository/conf/deployment.toml file. This parameter is set to true by default. If you set it to false, you only see the available tiers.

```
[apim.throttling]
enable_advanced_throttling = true
```


Subscription Rate Limiting (API Subscriber)

Selecting the Rate Limiting tier when subscribing to the API

Generate Credentials

Use the Key Generation Wizard. Create a new application -> Subscribe -> Generate keys and Access Token to invoke this API.

SUBSCRIBE WITH A NEW APPLICATION

Subscribe to an application and generate credentials

Application

DefaultApplication

Throttling Policy

Unlimited

Select an Application to subscribe

Available Policies - Unlimited

SUBSCRIBE

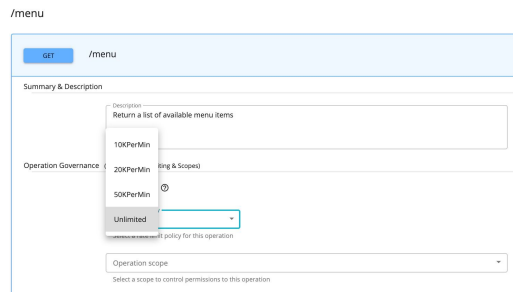
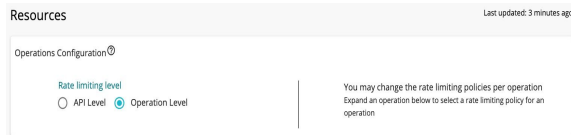


After subscription-level rate limiting tiers are set and the API is published, at subscription time, the consumers of the API can log in to the API Developer Portal and select which tier (out of those enabled for subscribers) they are interested in when subscribing to the API.

Advanced Rate Limiting (API Publisher)

Applicable in API-Level as well as Resource Level rate limiting when publishing the API

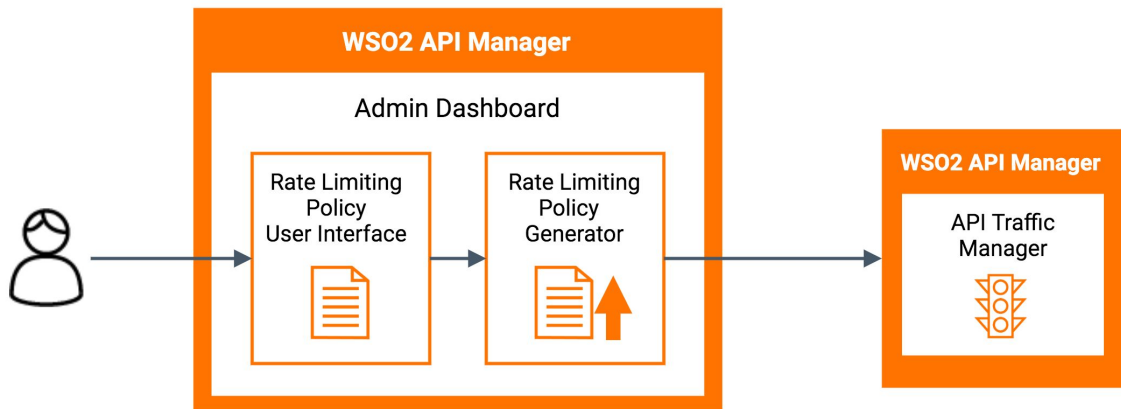
- 10KPerMin: 10,000 requests per minute
- 20KPerMin: 20,000 requests per minute
- 50KPerMin: 50,000 requests per minute
- Unlimited: Unlimited access



Advanced rate limiting policies are applicable on the API and also for Resources. They are defined when managing the API in the Publisher portal. Resource-level rate limiting tiers are set to HTTP verbs of an API's resources as shown.

Adding New Advanced Rate Limiting Policy

Admin Portal : <https://localhost:9443/admin>



We can add new advanced Rate limiting policies through the Admin Portal by adding a new advanced rate limiting tier.

We can also set Conditional groups for the tier, such as IP conditions, header conditions, etc.

Adding New Advanced Rate Limiting Policy

Admin Portal : <https://localhost:9443/admin>

General Details
Provide name and description of the policy. The policy can be referred from the name.

Name of the throttle policy.

Description of the throttle policy.

Default Limits
Request Count and Request Bandwidth are the two options for default limits. You can use the option according to your requirement.

Default Limit Option
☒ Request Count ☐ Request Bandwidth

Number of requests allowed

Time configuration

Conditional groups
To add throttling limits with different parameters base on IP, Header, Query Param, and JWT Claim conditions, click Add Conditional Group.

Conditional groups
To add throttling limits with different parameters base on IP, Header, Query Param, and JWT Claim conditions, click Add Conditional Group.

Hide group ^

Condition Policies

IP Condition Policy
This configuration is used to throttle by IP address.

Header Condition Policy
This configuration is used to throttle based on Headers.

Query Param Condition Policy
This configuration is used to throttle based on query parameters.

JWT Condition Policy
This configuration is used to define JWT claims conditions

Default Limit Option
☒ Request Count ☐ Request Bandwidth

Number of requests allowed

Time configuration

Description

Description of this group

Some Conditional Groups :

- IP Condition** - Allows you to set a rate limit for a specific IP address or a range of IP addresses.
- Header Condition** - Allows you to set a rate limit to specific headers and parameters.
- Query Param Condition** - Allows you to set a rate limit to specific query parameters.
- JWT Claim Condition** - Allows you to set a rate limit to specific claims.

Rate Limiting Policy Definition

```
@Plan:name('carbon.super_sub_Gold')
```

Policy Name

```
@Plan:description('ExecutionPlan for sub_gold')
```

Policy Description

```
@Import('org.wso2.throttle.processed.request.stream:1.0.0')
define stream RequestStream (messageID string, appKey string, appTier string,
subscriptionKey string, apiKey string, apiTier string, subscriptionTier string,
resourceKey string, resourceTier string, userId string, apiContext string, apiVersion
string, appTenant string, apiTenant string, appId string, apiName string,
propertiesMap string);
```

Request Stream Definition

```
@Export('org.wso2.throttle.globalThrottle.stream:1.0.0')
define stream GlobalThrottleStream (throttleKey string, isThrottled bool,
expiryTimeStamp long);
```

Global Throttle Stream Definition

<http://wso2.com/library/articles/2016/09/article-introducing-wso2-api-managers-throttling-implementation-architecture/>

Once the Traffic Manager receives a request, Siddhi runtime will process the deployed policies in the traffic manager instance.

These are the sections of rate limiting policy definition.

- Policy name: name of the rate limiting policy
- Policy description: description of rate limiting policy
- Request stream definition: mapping between incoming data from rate limiting request stream to a Siddhi stream named RequestStream that can be understood by Siddhi runtime.
- Global throttle stream definition: mapping of Siddhi output stream named GlobalThrottleStream to outgoing global throttle stream.

Rate Limiting Policy Definition

```
FROM RequestStream
SELECT messageID, ( apiTenant == 'carbon.super' and subscriptionTier == 'Gold') AS
isEligible, subscriptionKey AS throttleKey
INSERT INTO EligibilityStream;
```

Eligibility Query

```
FROM EligibilityStream[isEligible==true]#throttler:timeBatch(1 min, 0)
select throttleKey, (count(messageID) >= 5000) as isThrottled, expiryTimeStamp
group by throttleKey
INSERT ALL EVENTS into ResultStream;
```

Check Rate Limit State

```
from ResultStream#throttler:emitOnStateChange(throttleKey, isThrottled)
select *
insert into GlobalThrottleStream;
```

Notify Rate Limiting Decision

<http://wso2.com/library/articles/2016/09/article-introducing-wso2-api-managers-throttling-implementation-architecture/>

These are the sections of rate limiting policy definition.

- Eligibility query: this query will decide whether the policy will be further processed or not.
- Check Rate limit state: query that checks whether the current request is rate limited or not.
- Notify rate limiting decision: the operation that sends the rate limiting decision to the JMS Topic. Query in this section sends a message to the JMS Topic if there is a change in the Rate limiting state. Moreover, if the current request is rate limited, then the query will send a message to the JMS Topic. If the request is rate limited, gateways won't publish any data to the traffic manager until the rate limit time interval is over.

Application Rate Limiting (Application Developer)

Defined when creating the application in the API Developer Portal

Subscription & Key Generation Wizard

1 Create application 2 Subscribe to new application 3 Generate

Application Name *
PizzaApp

Enter a name to identify the Application. You will be able to pick this application when subscribing to APIs

Per Token Quota *
10PerMin

Assign API request quota per access token. Allocated quota will be shared among all the subscribed APIs of the application.

Application Description

(512) characters remaining

CANCEL NEXT

Subscription & Key Generation Wizard

1 Create application 2 Subscribe to new application 3 Generate

Application Name *
PizzaApp

Enter a name to identify the Application. You will be able to pick this application when subscribing to APIs

Per Token Quota *
10PerMin
20PerMin
50PerMin
Unlimited

(512) characters remaining

CANCEL NEXT

An application is available to a consumer at different levels of service. For example, if you have infrastructure limitations in facilitating more than a certain number of requests to an application at a time, the rate limiting tiers can be set accordingly so that the application can have a maximum number of requests within a defined time.

Adding New Application Rate Limiting Policy (Per Token Quota)

The screenshot displays the WSO2 API Manager Admin Portal interface. On the left, a sidebar menu shows 'Application Policies' highlighted. The main content area is titled 'Application Rate Limiting Policies' and contains a table of existing policies. A 'Add Policy' button is visible in the top right corner of the table. An 'Add Policy' dialog box is open on the left, showing the 'General Details' and 'Quota Limits' sections.

Add Policy Dialog - General Details:

- Name: 100PerMin
- Description: Allows 100 requests per min

Add Policy Dialog - Quota Limits:

- Request Count: ☒ Request Count, ☐ Request Bandwidth
- Request Count: 100
- Unit Time: 1 Minute(s)

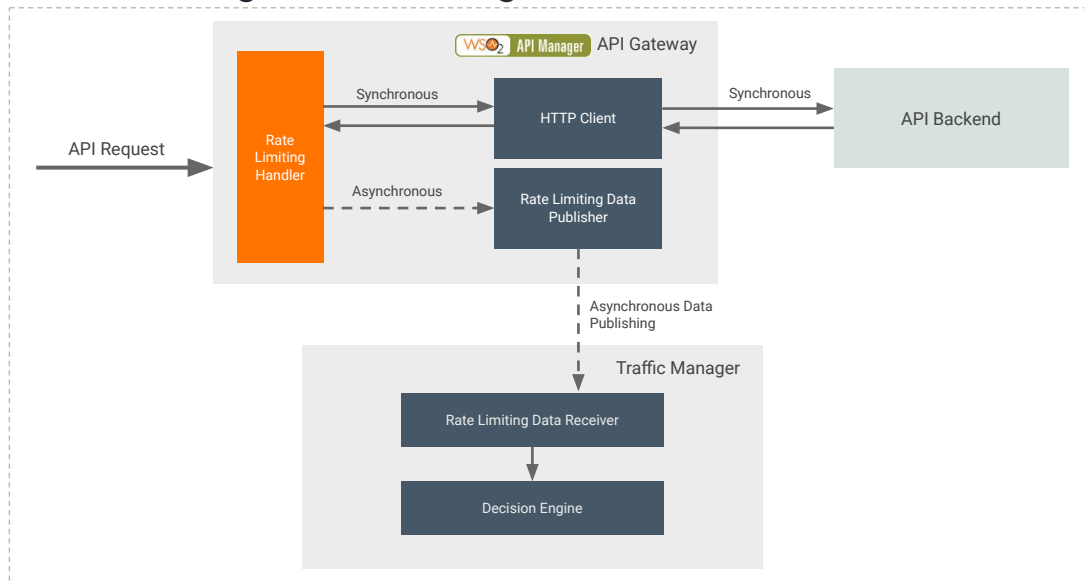
Application Rate Limiting Policies Table:

Name	Quota	Unit Time	Actions
50PerMin	50	1 min	Edit Delete
20PerMin	20	1 min	Edit Delete
10PerMin	10	1 min	Edit Delete

Admin Portal : <https://localhost:9443/admin>

Application-level rate limiting policies are applicable per access token generated for an application.

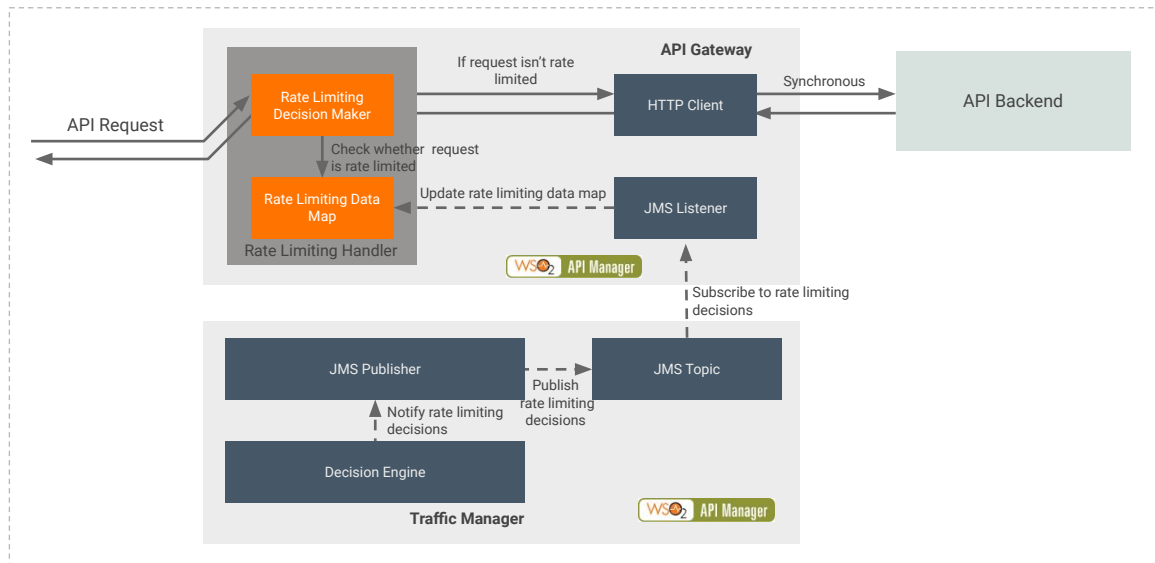
Rate Limiting Data Publishing



Traffic Manager has the responsibility of making rate limiting decisions

- Data required to make rate limiting decisions need to be published to the Traffic Manager
- Each Gateway in a deployment asynchronously publishes data required to make rate limiting decision for every API request to the Traffic Manager

Policy Evaluation and Notifications



- Traffic Manager has the responsibility of making rate limiting decisions
- The Siddhi Runtime in Traffic Manager processes events from gateways
- Policies deployed in traffic manager are executed on each event
- An event that triggers a condition in a policy will be notified to gateways through a JMS topic
- Each gateway maintains a rate limiting data map to check whether a request is within the allowed quota.
- Gateways update the rate limiting data map from the JMS Topic which is notified by the Traffic Manager

Custom Rate Limiting and Denying Requests

Custom Rate Limiting Policy - Define Policy

Name*
CustomRule

Description
This is a custom Policy

Key Template*
\$userId:\$apiContext:\$apiVersion

Siddhi Query:

The following sample query will allow 5 requests per minute for an Admin user.
Key Template : \$userId

```
1 FROM
2 RequestStream
3 SELECT
4   userId,
5   (userId == 'admin@carbon.super') AS isEligible,
6   $keyTemplate('admin@carbon.super', '') AS throttleKey
7 INSERT INTO
8   EligibilityStream;
9 FROM
10  EligibilityStream [isEligible=true] #throttle:timeBatch(1 min) SELECT throttleKey, (count(userId))
```

Add **Cancel**

Select Item to Deny

Condition Type

☒ API Context ☐ Application ☐ IP Address ☐ IP Range ☐ User

Value *

Format : \${context}

Eg : /test/1.0.0

☒ Enable Condition

Cancel **Deny**

Link - [Deny Policies](#)

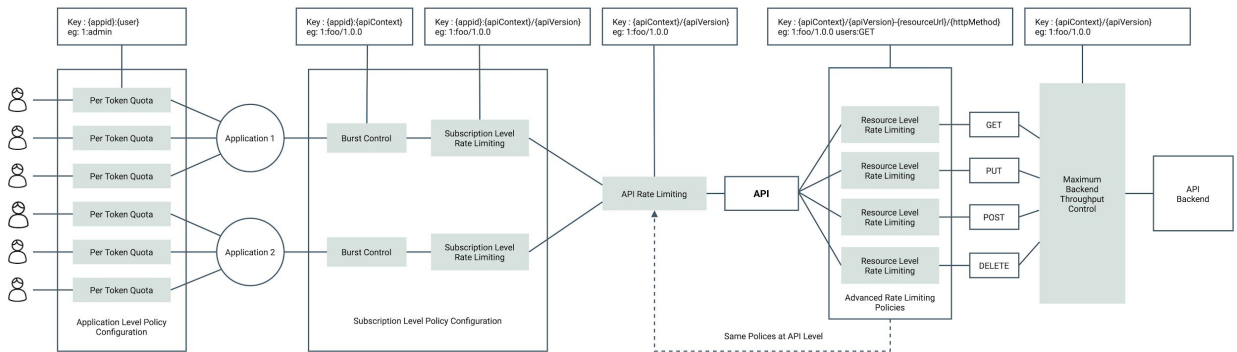
Link - [Custom Throttling](#)

Custom policies allow users to write custom Siddhi queries. Depending on the data received from gateway API calls, these policies will be executed and will send rate limiting decisions through JMS topic to gateways. Users may write Siddhi queries to limit activities of certain users based on their API invocation data that comes to the traffic manager. The throttle key is an important part of defining the policy. The key template definition available in the policy configuration will be sent to the gateway and then replaced with actual values of the request and will decide if the current request is rate limited or not. These policies extend the flexibility of rate limiting by allowing users to write custom rate limiting policies.

Deny policy conditions provide functionality to:

- instantly block a user who invokes APIs by username
- block API calls coming from a specific IP address by specifying IP
- block an API by API context
- block API calls coming from an application by application name.


Rate Limiting Policies Applicable at Different Levels



<http://wso2.com/library/articles/2016/09/article-introducing-wso2-api-managers-throttling-implementation-architecture/>



This is how the above discussed policies are applied in different levels of rate limiting.
First check the blocking conditions



Let's try it out!

Working with Rate Limiting Policies

