An Oracle White Paper
May 2011

Concurrency and Throttling in the Oracle Enterprise Gateway

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Concurrency

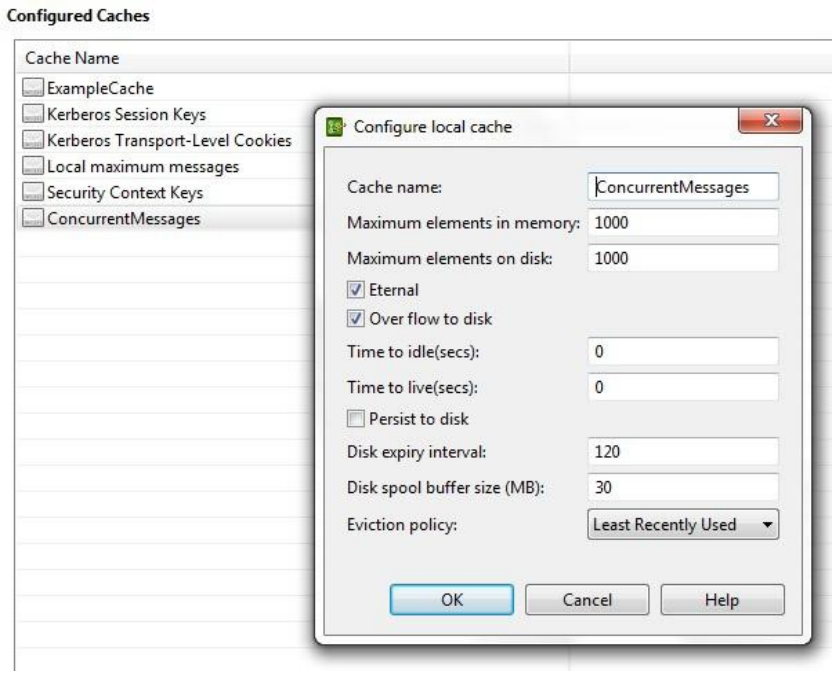
Scenarios for Limiting Concurrency

OEG can limit concurrent requests to services. The typical scenarios for this are:

- A server can only handle a certain number of concurrent messages. Therefore OEG is placed in front of the server to keep the number of concurrent messages below this limit. Note that the "SR" (Service Request) command-line load-testing tool provided with OEG Service Explorer can be used to test and find concurrent messages processing limits for a particular server, which can then be enforced by OEG itself.
- Traffic above a certain level may indicate a data-harvesting attack. Therefore OEG would be configured to keep traffic below this level.
- An organization wishes to spread processing across a number of servers. OEG can be used to limit the concurrent messages to a server, but then distribute the overflow messages to a secondary server, which itself may have a similar rule. In this way, traffic is distributed across servers.

How it works

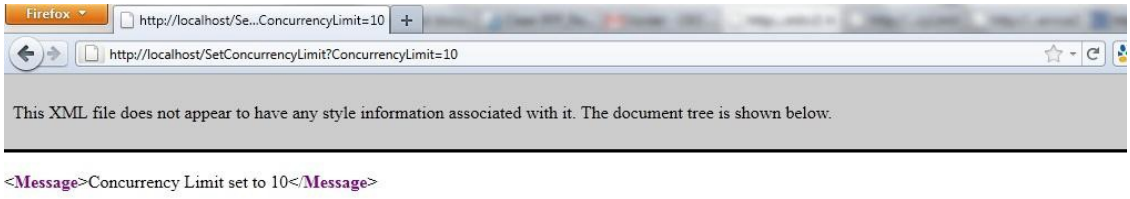
In order to keep track of the current number of concurrent messages, a counter is used. OEG uses its in-memory cache for this purpose. In the following screenshot we see this cache in place:



The Maximum Concurrent Requests value may be read from a variety of places, including:

- Database
- LDAP directory
- Web Service call
- File read from the file system
- Note that this is not an exhaustive list.

It is also possible to set the Maximum Concurrent Requests value via a REST API, as shown below, if a customer wishes to do this. This REST API may be protected by a policy which ensures it can only be run by certain authenticated users.



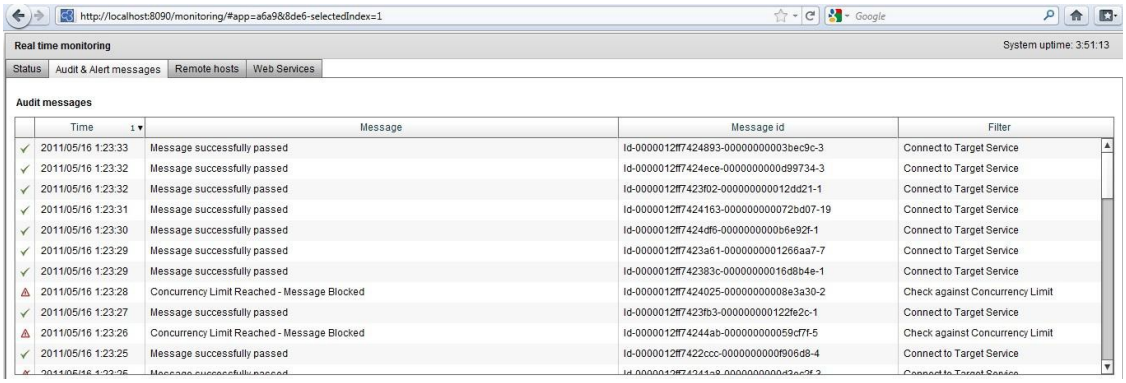
Actions taken once Concurrent Request Limit is reached

OEG administrators may set a variety of actions to take place once the concurrent limit is reached. These actions are triggered once the counter increments to the Maximum Concurrent Requests value. OEG administrators may configure a variety of actions to occur once the maximum concurrent messages limit is reached. The actions can include:

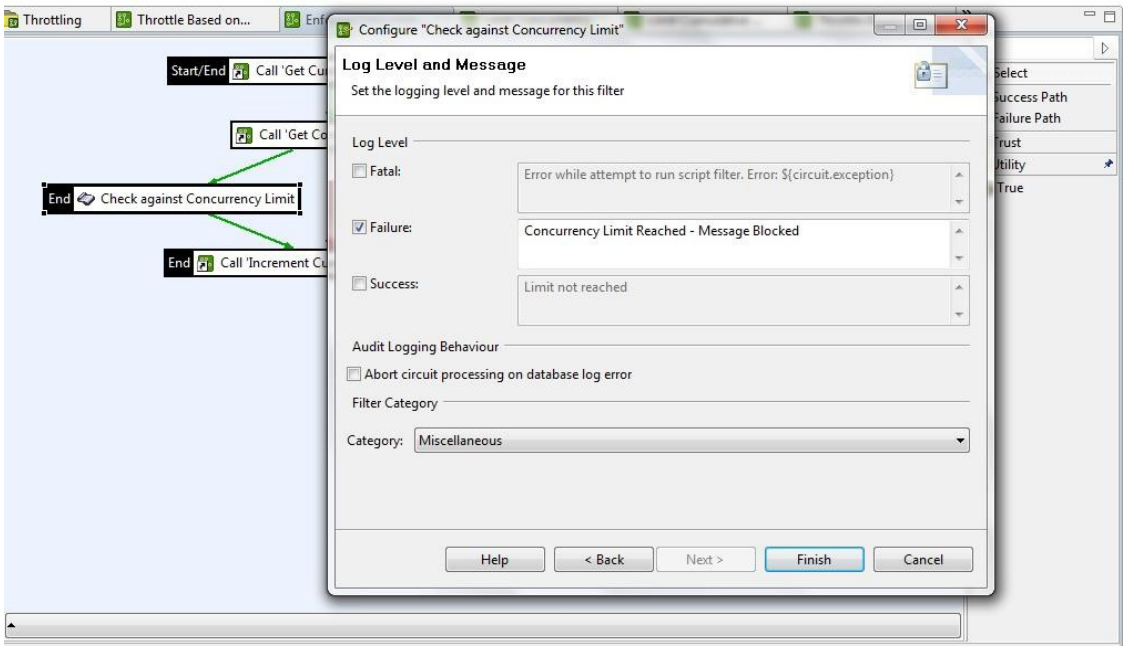
- Raise an alert [email, SNMP, syslog, etc]
- Route overflow requests to a secondary server [for scenarios where a particular server can only process a certain number of concurrent requests]
- Return a particular message to the client (e.g. "Server Limit has been reached, please try again") with a configurable HTTP code
- Place the message on a message queue to be read at a later point by OEG's message queue reader. In this way, the message may be processed at a later point when the concurrent message count is less.

Alerting

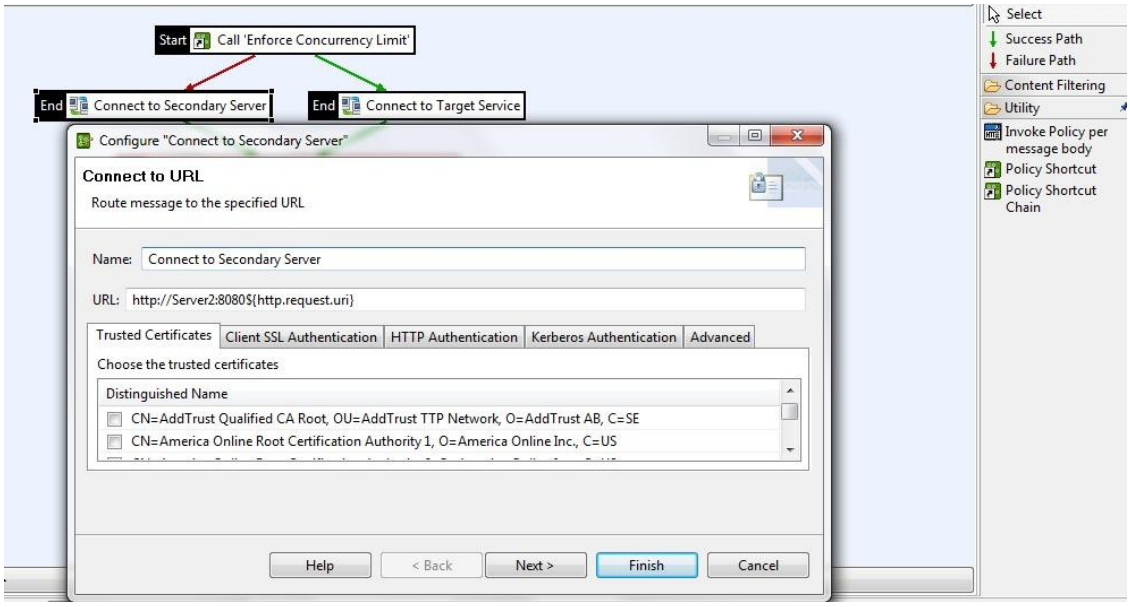
In the screenshot below we see real-time monitoring information about messages which are being blocked because the maximum concurrent messages limit has been exceeded:



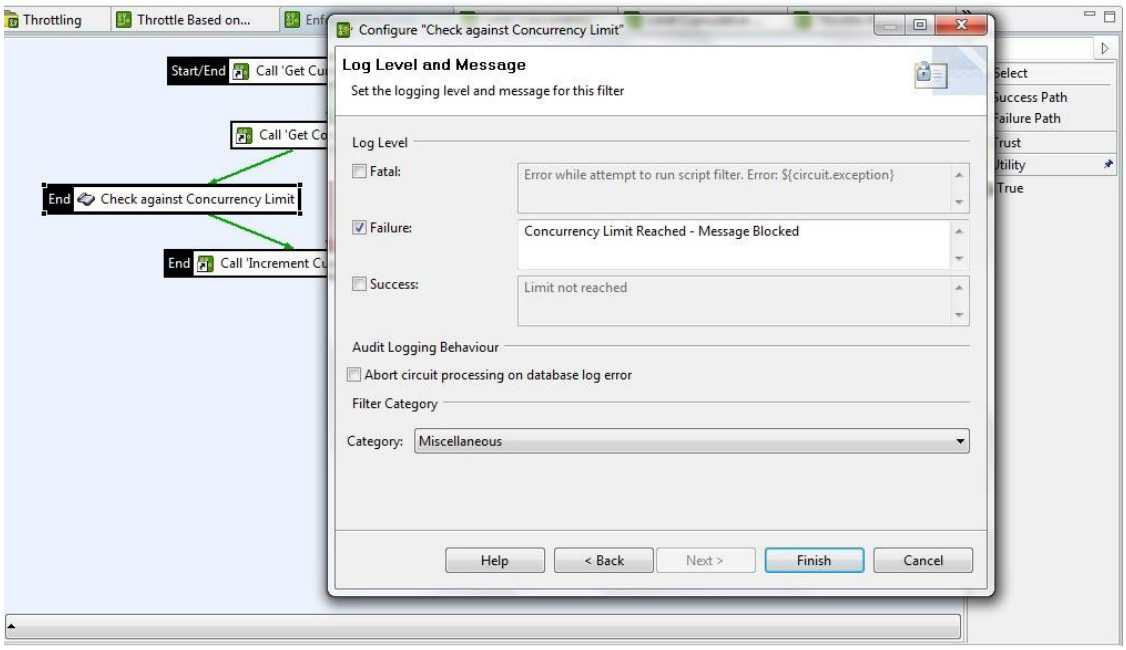
In the following screenshot we see the policy which is checking the concurrency limit. We see the filter which checks if the current number of messages processed has reached the concurrency limit. If it has, then it goes down the "Failure Path" and its failure log message is recorded. Here we see the "Concurrency Limit Reached" message which is logged, and which matches the text shown in the screenshot above.



As mentioned above, a common reason to limit concurrency is because it has been found that the target server cannot process more than a certain number of requests per second. In the screenshot below, we are catching the "failure case" of the policy which checks against the concurrency limit, and we are then routing the overflow requests (those above the concurrency limit) to a secondary server.



In a similar manner, an alert can be generated based on messages reaching the concurrency level. This is shown below. An "Alert" filter is placed after the "Check against Concurrency Limit" policy call. Note that a number of actions can be done in sequence after the concurrency limit has been reached (e.g. raise an alert and then route the overflow of messages to a secondary server).



Throttling

OEG performs throttling for a number of purposes:

- Blocking Denial-of-Service Attacks
- Ensuring quality of service
- Providing different levels of service to different types of clients (e.g. "Gold customers can do 1000 requests/second, Silver customers can do 100 requests/second").

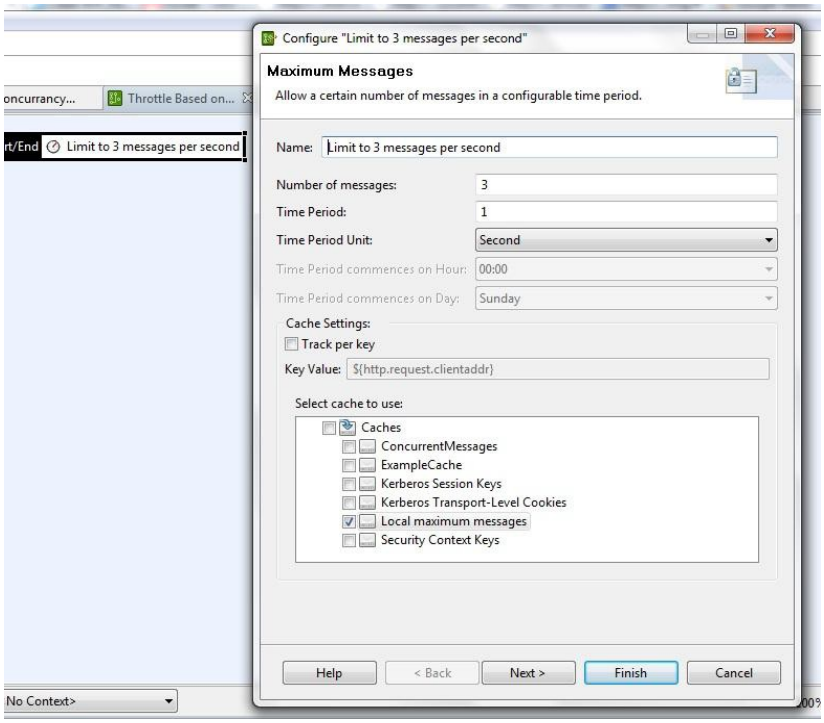
This is achieved through the Throttling feature in OEG. The filter used for this feature is the Maximum Messages filter, which is configured in Policy Studio. In this section we will see how this is configured. We will walk through a use case where a PartnerID value, taken from the incoming HTTP header, is used as the "key" to throttle against, so that each PartnerID can only do a certain amount of requests per hour.

Throughput – Limiting the number of requests per second irrespective of the concurrency.

OEG uses the Maximum Messages filter to limit traffic throughput. Next we see some examples of the Maximum Messages filter in action.

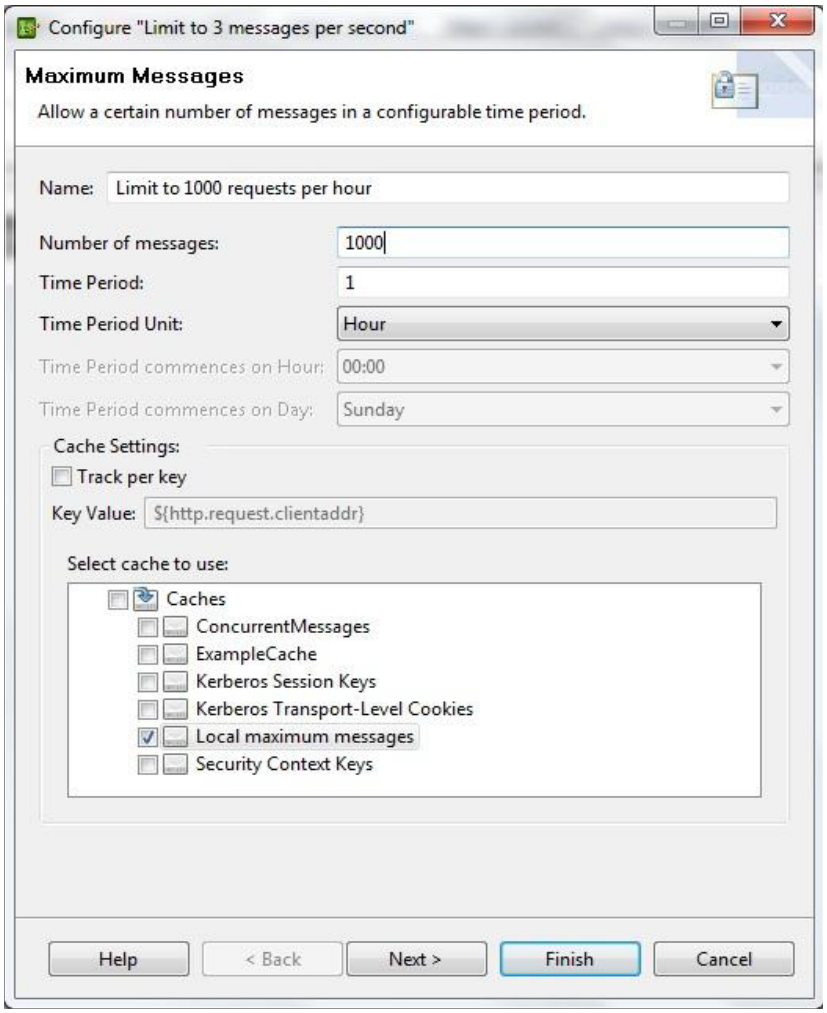
Limiting Traffic per second

OEG limits traffic-per-second using its Maximum Messages filter. In the example below, we configure the filter to limit request to three per second. Notice that a cache must be chosen. This is because the accumulating count must be stored between requests, and so it is stored in an in-memory cache. If a distributed cache is chosen, then the count is used across multiple Gateways.



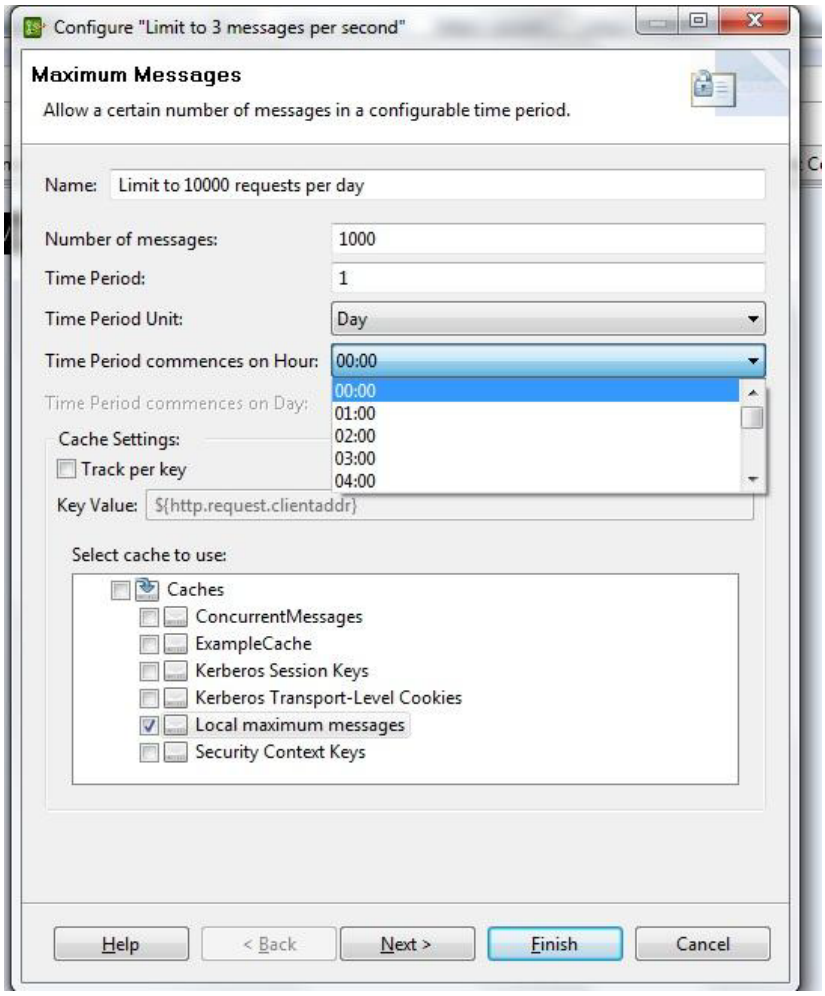
Limit requests services per hour

The "Maximum Messages" filter is also used for longer times than seconds. In the example below, it is limiting requests to 1000 requests per hour.



Limit per day

Finally, we see how the Maximum Messages filter may be used to limit requests by the day. Note that the start hour of the day may be chosen.



Testing Throttling

The SR tool provided with OEG Service Explorer provides a convenient method to test throttling. The following SR command-line invocation will connect to the local machine on port 80 (assuming this where OEG is listening) and request "/Service2" with ten parallel connections for ten seconds. This generates traffic to trigger throttling rules, depending of course on what values have been set for the throttling rules.

sr -h localhost -u /Service2 -s 80 -p10 -d10

How to Configure Throttling across multiple services

Throttling in OEG is configured using message filters. These are dragged and dropped onto the Policy Studio canvas to create a policy. The Policy may be applied across many services by, for example, mapping it to the "/" relative path on OEG.

What parameters may be used for throttling?

Throttling rules may be applied across a wide variety of parameters. These include:

- HTTP Headers
- HTTP QueryString values
- XML message content (e.g. if a partner ID is located into the incoming XML message)
- A value looked up from a database
- A value looked up from LDAP (for example, a user's role may be looked up from an LDAP directory, and then throttling may be applied per-role)
- Single Sign-On cookies (e.g. the obsso cookie used by Oracle Access Manager)
- Client Identity
- Client IP address
- Client browser type

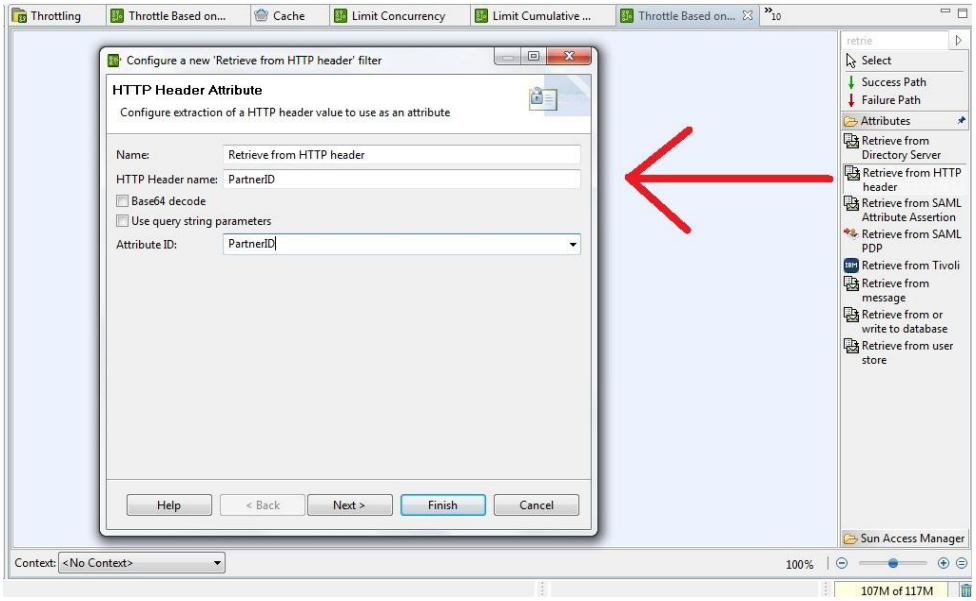
In fact, any variable available to OEG may be used in a throttling rule.

Step by Step Configuration

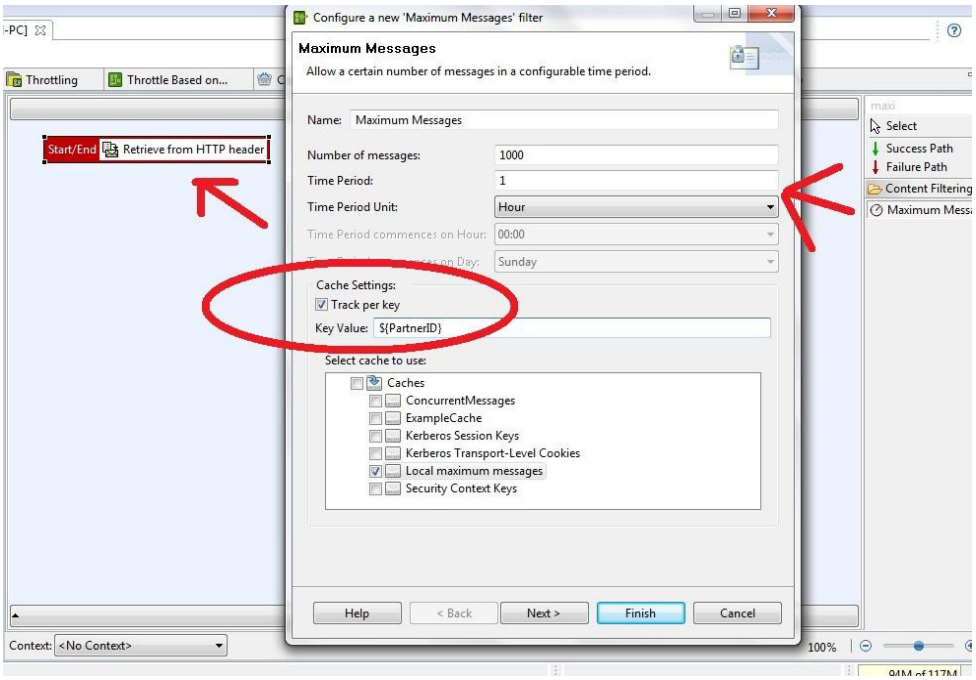
In the following example, we will configure throttling across services, based on a HTTP header called "PartnerID". We will configure a rule that any given PartnerID value can only be used for 100 messages a minute.

To do this, first create a new blank policy. This is done in OEG Policy Studio by right-clicking on "Policies" and selecting "New Policy".

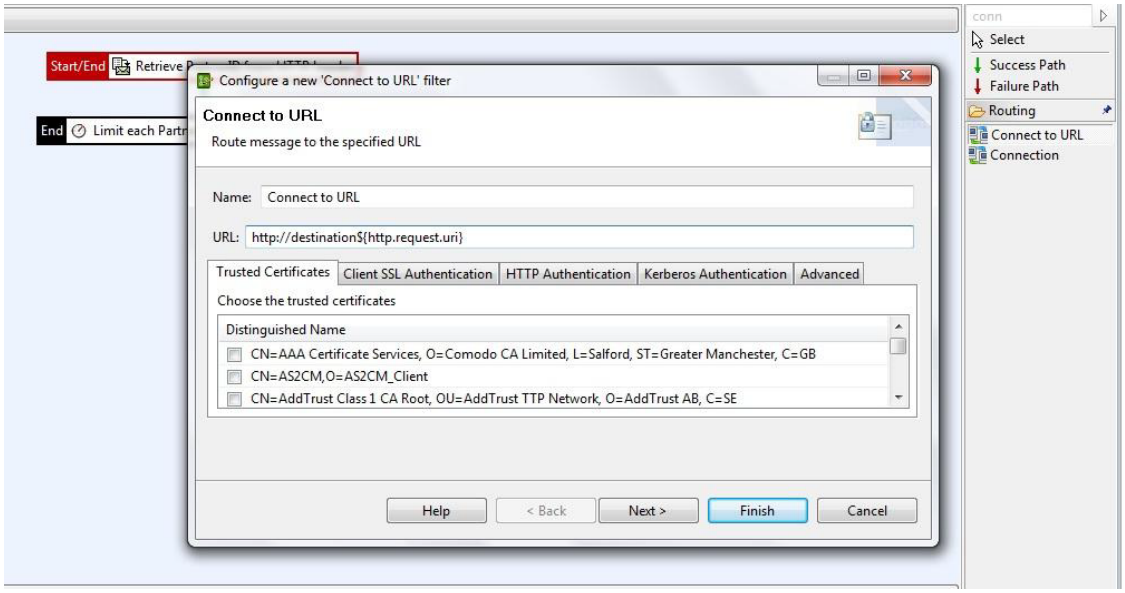
We first drag a "Retrieve from HTTP Header" filter into the policy canvas. We configure this, as shown below, to look for a HTTP header value called "PartnerID".



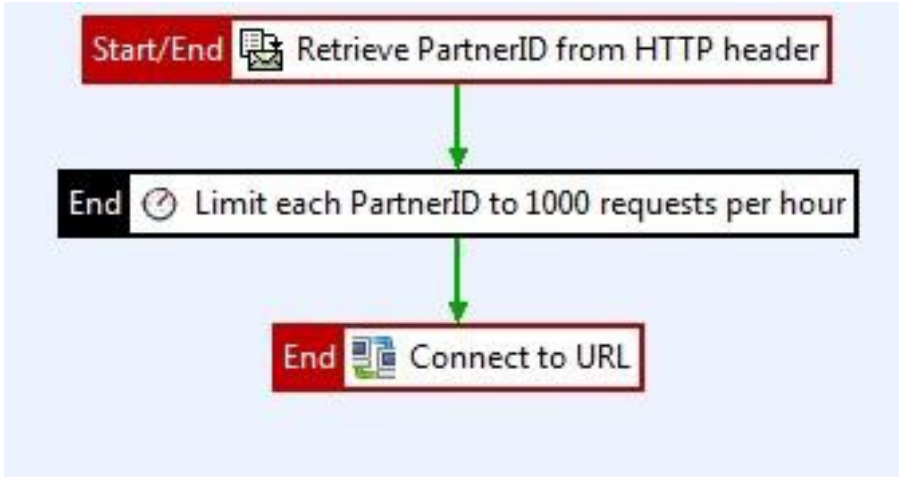
Next we use a Maximum Messages filter, and ensure that it is throttling based on PartnerID. The configuration below ensures that any given PartnerID value can only be used for 1000 requests per hour. After that amount is reached, this filter will return "false".



Finally we place a “Connect to URL” filter after the throttling filter. Note that in this example we are using a wildcard value for relative path (${http.request.uri}). This means that if a client connects to OEG with a URL like http://OEG_address/ServiceName , then OEG will route automatically to http://DestinationServer/ServiceName . In the example below, we simple call the destination server “destination”, but this should be the name of the server behind OEG, which is hosting the services it is protecting.



Our policy now looks like the following:



If mapped to the path “/” on OEG, this policy will work across-the-board for all downstream services.

How can throttling violations be handled?

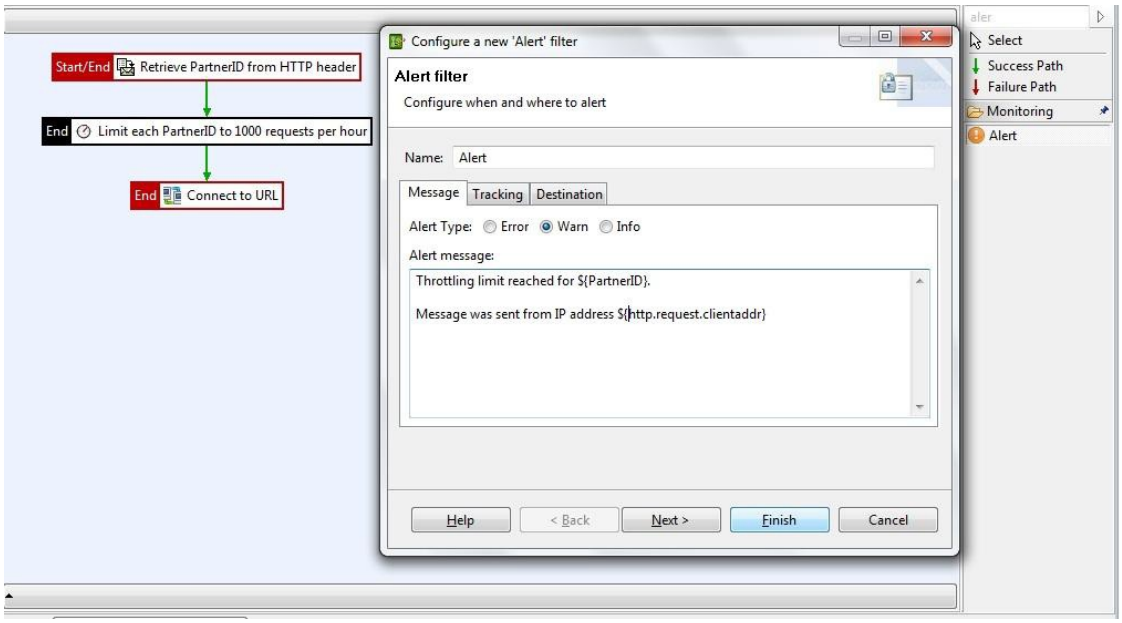
The list of possibly actions when a throttling limit is very similar to the list of actions which can be taken if a concurrent message limit is hit. These are:

- Raise an alert [email, SNMP, syslog, etc]
- Route overflow requests to a secondary server [for scenarios where a particular server can only process a certain number of concurrent requests]
- Return a particular message to the client (e.g. "Server Limit has been reached, please try again") with a configurable HTTP code
- Place the message on a message queue to be read at a later point by OEG's message queue reader. In this way, the message may be processed at a later point when the concurrent message count is less.

How to configure actions once the throttling limit is reached

Handling the situation where throttling limit is reached is performed by handlig the *failure case* of the throttling (Maximum Messages) filter. Let's take the example of raising an alert.

Drag in an "Alert" filter onto the canvas, and configure it as shown below. Under the "destination" tab, this is where an alert destination is chosen. This can be email, SNMP, syslog, or other destinations. Notice in the example below we are including details of which PartnerID reached its throttling limit. Note also that you can take advantage of the Alert Filter's "tracking" tab to limit the number of alerts.



After the Alert filter, which now follows the red “failure” path of the throttling filter, a further “Connect to URL” filter may be added, if the overflow requests are to be routed to a secondary server. In this way, multiple actions can be taken if a throttling filter reaches its limit.

Oracle Enterprise Gateway
May 2011
Author:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

SOFTWARE. HARDWARE. COMPLETE.