API Reference

Pacing REST API

5/2/2025

Pacing REST API

Overview

The Pacing API gives external components access to your pacing information, using two different methods:

- The Reactive State method returns a number that indicates the probability of whether additional reactive traffic will displace proactive traffic. **It does not return the number of agents who are ready to accept chat interactions.**
- You can, however, use the Channel Capacity method to figure out how many agents are ready to process incoming requests.

> **Important**
> Please read the following information carefully before attempting to use the Pacing API!

Reactive State

Use this method to determine whether reactive traffic is displacing proactive traffic. If so, you may want to take action, such as limiting the number of chat interactions that are initiated in response to the reactive requests. To use this method, you must configure the Pacing Algorithm to use a type that predicts reactive engagements, that is, either `SUPER_PROGRESSIVE_DUAL` or `PREDICTIVE_B_DUAL`.

> **Important**
> As noted below, a request using this method returns the probability that a new reactive engagement should be allowed for a visitor. **It does not contain information about the number of agents who can accept an incoming interaction.**

Request

Method	GET
URL	http://<backend_host:backend_port>/backend/data/pacing/reactiveState?channel=<channelName>&groups=[<names>]

Method	GET
	The HTTPS schema is also allowed, if configured.

Parameters

Name	Value	Mandatory	Description
channel	string	yes	The name of a media channel, which determines if a reactive engagement is possible. Valid values are `chat` or `webcallback`.
groups	string	no	The list of agent group names. If this parameter is not defined, then the reactive pacing result is consolidated over all groups. **Note:** If you want to specify more than one group, you must use the following syntax: **&groups=*Group1_name*&groups=*Group2_na*

Response

Response	`{reactiveState : <value>}` The request returns the value of the pacing reactive state. This value is the probability that a new reactive engagement should be allowed for a visitor.
Response Type	JSON

Example

```
<script>
    $.ajax({url: 'http://{server}:{port}/backend/data/pacing/reactiveState?channel=chat'})
    .done(function( result ) {
        console.log('result: ' + JSON.stringify(result.reactiveState));
                var rnd = Math.random();
        if(rnd <= result.reactiveState) {
            // Do something
        }
    });
</script>
```

> **Important**
>
> jQuery is required for the example above.

Channel Capacity

> **Important**
> The channel capacity functionality was introduced in Genesys Web Engagement 8.1.200.41

Use this method to figure out how many agents are ready to process incoming requests.

Note: This method does not take into account the extent to which reactive traffic is displacing proactive traffic. Because of this, if you use its results without taking other factors into account, you may reduce the effectiveness of your proactive campaigns.

Request

Method	GET
URL	http://<backend.host:backend.port>/backend/data/pacing/channelCapacity?channel=<channelName>&groups=[<names>] The HTTPS schema is also allowed, if configured.

Parameters

Name	Value	Mandatory	Description
channel	string	yes	The name of a media channel for which you want to determine the count of ready agents. Valid values are `chat` or `webcallback`.
groups	string	no	The list of agent group names. If this parameter is not defined, then the resulting agent count is consolidated over all groups. **Note:** If you want to specify more than one group, you must use the following syntax: **&groups=*Group1_name*&groups=*Group2_na*

Response

Response	`{capacity : <value>}` The request returns the count of ready agents in the specified group or groups—or for the entire channel, if no group is specified—according to statistics provided by Stat Server.
Response Type	JSON

Example

```
<script>
    $.ajax({url: 'http://{server}:{port}/backend/data/pacing/channelCapacity?channel=chat'})
    .done(function( result ) {
        console.log('Ready agents capacity is: ' + JSON.stringify(result.capacity));
    });
</script>
```

> **Important**
> jQuery is required for the example above.