Queue Overview - Realtime Data

Let's have a look at the steps needed to get access to Queue Overview data similar to the data under the Real-time tab shown in Connect Control. In any use case, one of the first things to do is authenticate and get an 'accesstoken' which you will use for all subsequent REST calls.

Step 1 Authenticate:

https://auth.puzzel.com/api/Authenticate/LogIn

```
{
  "customerKey": "nnnnn",
  "userName": "xxxxx",
  "password": "yyyyy",
  "clientName": "zzzzz",
  "clientVersion": "ccccc"
}
```

Result gives you the "Access Token", referred to as accessToken from hereon (the value of "result").

```
{
  "accessToken": "YourAccessToken",
  "refreshToken": {
    "customerKey": "xxxxx",
    "sessionId": "YourSessionID",
    "expiryDate": "2022-08-31T23:14:25.98"
  },
  "languageCode": "SE",
  "twoFactorRequired": false,
  "twoFactor": null,
  "logoutFromExternalIdentityProvider": false
}
```

Step 2 AccessTokenInformationGet:

We need to get the details for this accessToken to determine the expiry and the userId of the authenticated user for subsequent REST calls.

https://api.puzzel.com/ContactCentre5/accesstokeninformation

Sample response

```
{
  "result": {
    "customerKey": "string",
    "customerId": 0,
    "userGroupId": 0,
    "userId": 0,
    "languageId": 0,
    "languageCode": "string",
    "passwordValidDaysLeft": 0,
    "accessTokenExpiry": "2024-08-12T07:58:21.841Z"
  },
  "code": 0,
  "id": "string",
  "message": "string"
}
```

Result gives us the accessTokenExpiry and userId values. You should re-use the accessToken until it expires and when required use the refreshToken returned in Step1 to get a new accessToken using the below Request URL.

https://auth.puzzel.com/api/Authenticate/GetAccessToken

Refer to https://auth.puzzel.com/swagger/ui/index#!/Authenticate/Authenticate_GetAccessToken

Step 3 VisualQueueStateAndTickerList:

We are now ready to request the data for all queues.

https://api.puzzel.com/ContactCentre5/90031/visualqueues/stateinformation/All?userId=userId&visualQueueId=visualQueueId

In this request we send the the 'userId' along with an 'visualQueueId'.

Notice that there are several visualQueueResult options we can ask for (All, UserRelevant, UserMember). This determines the scope of the data we get back.

(Refer to the API Explorer for descriptions).

We now have details of the current status of each queue as well as the metrics for requests so far (data for Today, starting from midnight, upto the moment we made this request).


```
{
  "result": [
    {
      "id": 0,
      "description": "string",
      "waitTimeMaxSeconds": 0,
      "waitTimeAverageSeconds": 0,
      "queueSize": 0,
      "agentsLoggedOn": 0,
      "agentsInPause": 0,
      "agentsReady": 0,
      "agentsUnavailable": 0,
      "sla": 0,
      "queueSizeCiq": 0,
      "unblockedLoggedIn": 0,
      "unblockedUnavailable": 0,
      "queueSizePreferred": 0,
      "callsOfferedToday": 0,
      "callsAnsweredToday": 0,
      "ciqsOfferedToday": 0,
      "ciqsAnsweredToday": 0,
      "callsAnsweredWithinSla": 0,
      "alarmFlag": 0,
      "ciqScheduled": 0,
      "callsAbandonedToday": 0,
      "callsAbandonedWithinSla": 0,
      "silentCallsToday": 0
    }
  ],
  "code": 0,
  "id": "string",
  "message": "string"
}
```


So at this stage you can decide on how to present the data. A simple table view might look like this:

Queue Name	Queueing Now	Agents LoggedOn	Requests Offered	Requests Answered	%Answered within SLA

"description" = the name of the queue.

"waitTimeMaxSeconds" = maximum wait time for the requests in queue right now.

"queueSize" = the number of requests in queue night now.

"callsOfferedToday" = number of requests offered to Queue so far today.

"callsAnsweredToday" = number of requests answered so far today.

"callsAnsweredWithinSla" = number of requests answered within the set SLA.