3.16 User Management API Specification

Bright Pattern Documentation

Generated: 5/20/2024 6:07 pm

Table of Contents

Purpose

The Bright Pattern Contact Center *User Management API Specification* describes the methods and responses of the User Management API, which can be used for automated user provisioning with third-party systems. For example, if a new agent is hired, the agent's account is created in a corporate system, and at that time, the system can invoke the API and automatically create the user in the Bright Pattern system. When an agent is terminated, for example, the User Management API can be used to automatically delete or disable the user.

Audience

This guide is intended for the IT personnel responsible for the data infrastructure of Bright Pattern Contact Center-based contact centers. Readers of this guide are expected to have expertise in web application development as well as a solid understanding of contact center operations and resources that are involved in such operations.

General Information

The User Management API is a RESTful API. The HTTP POST method is used for all requests to insert or update information, while the HTTP GET method is used to retrieve information. Request and response bodies are encoded using JSON.

This specification describes only additional or different meanings of standard HTTP response codes. For more information, see RFC 2616, Section 10.

Request URLs

The request URL contains information about the required action and the object name.

The base part of the URL is:

http[s]://<host>/configapi/v2

Request Body

The request body carries either the object to be inserted or the unique ID fields of the object to be queried.

Tenant ID

In a multi-tenant deployment, the tenant ID is derived from authentication login information.

For more information, see the Security and Authentication section of this specification.

Security and Authentication

HTTP digest access authentication is used to authenticate the access attempts. The provided username is checked against the list of users configured at the contact center (tenant) level. Moreover, the session is established provided that the supplied credentials are authenticated and that the user's role indicates the user's authorization to perform operations.

Standard HTTP response codes whose meaning conforms to the original specification (RFC 2616) are not discussed in this guide. For specification of such responses, see section 10 of http://www.ietf.org/rfc/rfc2616.txt. This document only specifies the response codes whose description deviates from the original specification (e.g., is defined more narrowly or has a different meaning).

Example

Here is an example of how to authenticate in Python.

```
auth=HTTPDigestAuth('Username', 'Password')
```

Get User Data

Returns user data, such as name, team, extension, skills, and so forth.

Request

URL

http[s]://<host>/admin/ws/t/<tenant_url>/user/<login_id>

HTTP Method

GET

Response


```
{
 "loginId": "test008",
 "firstName": "test",
 "lastName": "008",
 "team": "Maintenance Renewal",
 "extension": "2072",
 "workPhone": "123456",
 "mobilePhone": "78910",
 "email": "test008@nowhere.net",
 "disabled": false,
 "changePassword": true,
 "skills": {
 "Maintenance Renewal": 100,
 "English": 33
 },
 "roles": [
   "Agent",
```

```
   "Supervisor"
 ]
}
```

Create User

Creates a new user in the system.

Request

URL

http[s]://<host>/admin/ws/t/<tenant_url>/user

HTTP Method

POST

Response

```
{
 "loginId": "test008",
 "password": "top secret",
 "firstName": "test",
 "lastName": "008",
 "team": "Maintenance Renewal",

 "extension": "2072",
 "workPhone": "123456",
 "mobilePhone": "78910",
 "email": "test008@nowhere.net",
 "disabled": false,
 "changePassword": true,
 "skills": {
 "Maintenance Renewal": 100,
 "English": 33
 },
 "roles": [
 "Agent",
 "Supervisor"
 ]
}
```

Update User

Changes a user's password, system status (i.e., active or inactive), and skills.

Request

URL

http[s]://<host>/admin/ws/t/<tenant_url>/user/<login_id>

HTTP Method

PUT

Response


```
{
 "password": "top secret",
 "disabled": false,
 "changePassword": true,
 "skills": {
  "Maintenance Renewal": 100,
  "English": 33
 }
}
```


Get User Lock State

Checks whether a user's account has been temporarily locked out after too many invalid login attempts.

Request

URL

http[s]://<host>/admin/ws/t/<tenant_url>/user/lock/<login_id>

HTTP Method

GET

Response


```
{
   "lockedOut":true
}
```


Clear User Lock State

Resets a temporary lockout, if a user's account has been temporarily locked out after too many invalid login attempts.

Request

URL

http[s]://<host>/admin/ws/t/<tenant_url>/user/lock/<login_id>

HTTP Method

PUT

Response


```
{
   "lockedOut":false
}
```