



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Predictive Routing Deployment and Operations Guide

[Code Samples](#)

Contents

- 1 Code Samples
 - 1.1 cURL Examples
 - 1.2 Python Examples

Code Samples

The code samples included in this topic are for use with the Predictive Routing API.

cURL Examples

The *Predictive Routing API Reference* includes cURL examples. The *Reference* requires a password for access. Please contact your Genesys representative if you need to view this document.

Important

If you copy and past the cURL example code into a Windows-based code editor, you must remove the \ characters at the ends of the lines.

Python Examples

The following examples provide a guideline for how to implement GPR API calls using Python. The examples are presented in the same groupings as in the *Predictive Routing API Reference*.

Authenticate Example

```
#POST / authenticate
response = requests.post(
    '%s/api/v2.0/authenticate' % HOST,
    data={
        'username': USERNAME,
        'password': PASSWORD,
        'api_key': API_KEY
    }
)
token = response.json()['token']
print token
```

Datasets Examples

```
# GET /datasets (get presigned URL) Get Dataset presigned URL for uploading file to S3
def get_presigned_url(token):
    response = requests.get(
        '%s/api/v2.0/datasets' % HOST,
        json={
            'token': token,
            'command': 'presigned_url',
            'filename': 'dataset.csv.zip'
        }
    )
```

```
)
    return response.json()['signed_url']

# PUT /presigned URL (upload to S3) Upload dataset to S3
def upload_dataset_s3(presigned_url):
    response = requests.put(presigned_url,
        data = open(DATASET_PATH, 'rb').read()
    )
    return response.json()

# POST /datasets (create using S3) Create a dataset using S3
def create_dataset_s3(token):
    response = requests.post(
        '%s/api/v2.0/datasets' % HOST,
        json={
            'token': token,
            'name': 'DatasetName',
            's3_separator': 'TAB',
            's3_filename': 'dataset.csv.zip',
            'encoding': 'shift-jis'
        }
    )
    return response.json()

# PUT /datasets/{id}/ (using s3) Append a dataset using a file uploaded to S3.
def append_dataset_s3(token):
    response = requests.put(
        '%s/api/v2.0/datasets/%s/append' % (HOST, DATASET_ID),
        json={
            'token': token,
            's3_separator': 'TAB',
            's3_filename': 'dataset.csv.zip',
            'encoding': 'shift-jis'
        }
    )
    return response.json()

# POST /datasets (create using file system)
def create_from_csv(token):
    response = requests.post(
        '%s/api/v2.0/datasets' % HOST,
        data={
            'token': token,
            'sep': 'TAB',
            'name': 'TestDataset ' + str(datetime.datetime.now())
        },
        files={
            'csv_file': open(SMALL_FILE, 'rb')
        }
    )
    return response.json()

# GET /datasets Get all datasets details
def get_all_ds_details(token):
    response = requests.get(
        '%s/api/v2.0/datasets' % HOST,
        json={
            'token': token,
            'fields2show': ['name', 'id', 'sync_status']
        }
    )
    return response.json()
```

```
# GET /datasets/{id} Get Dataset details.
def get_dataset_details(token):
    response = requests.get(
        '%s/api/v2.0/datasets/%s' % (HOST, DATASET_ID),
        json={
            'token': token,
            'include_cardinalities': 'false'
        }
    )
    return response.json()

# PUT /datasets/{id} Edit dataset.
def edit_dataset(token):
    response = requests.put(
        '%s/api/v2.0/datasets/%s' % (HOST, DATASET_ID),
        data={
            'token': token,
            'name': 'NR TestDataset 2019-07-02 16:33:12.116667',
        }
    )
    return response.json()

# DELETE /datasets/{id} Delete dataset.
def delete_dataset(token):
    response = requests.delete(
        '%s/api/v2.0/datasets/%s' % (HOST, DATASET_ID),
        data={
            'token': token,
        }
    )
    return response.json()

# GET /datasets/{id}/data Get Dataset details using a filter or search.
def filter_ds(token):
    response = requests.get(
        '%s/api/v2.0/datasets/%s/data' % (HOST, DATASET_ID),
        json={
            'token': token,
            'skip': 0,
            'limit': 100,
            "filter": "Agent_Age>30&Agent_Skill2=Billing",
            "search": "Buyer"
        }
    )
    return response.json()

# PUT /datasets/{id}/append (using file system) Append a dataset using a CSV file or a zipped
# CSV file.
def append_data_csv(token):
    response = requests.put(
        '%s/api/v2.0/datasets/%s/append' % (HOST, DATASET_ID),
        data={
            'token': token,
            'sep': 'TAB',
        },
        files={
            'csv_file': open(SMALL_FILE, 'rb')
        }
    )
    return response.json()

# PUT /datasets/{id}/append (using JSON) Append a dataset using JSON batch data.
```

```
def append_data_batch(token, dataset_id):
    response = requests.put(
        '%s/api/v2.0/datasets/%s/append' % (HOST, dataset_id),
        json={
            'token': token,
            'batch_data': [
                {
                    "a": "1",
                    "b": "abc",
                    "ts": "12345"
                }
            ]
        }
    )
    return response.json()

# PUT /datasets/{id}/apply_sync Sync dataset.
def sync_dataset(token):
    response = requests.put(
        '%s/api/v2.0/datasets/%s' % (HOST, DATASET_ID),
        json={
            'token': token,
            'command': 'apply_sync'
        }
    )
    return response.json()

# PUT /datasets/{id}/accept_sync Accept dataset.
def accept_sync(token):
    response = requests.put(
        '%s/api/v2.0/datasets/%s' % (HOST, DATASET_ID),
        json={
            'token': token,
            'command': 'accept_sync'
        }
    )
    return response.json()

# PUT /datasets/{id}/cancel_sync Cancel dataset synchronization.
def cancel_sync(token):
    response = requests.put(
        '%s/api/v2.0/datasets/%s' % (HOST, DATASET_ID),
        json={
            'token': token,
            'command': 'cancel_sync'
        }
    )
    return response.json()
```

Schemas Examples

```
# GET /schemas Get presigned URL
def get_presigned_url(token):
    response = requests.get(
        '%s/api/v2.0/schemas' % HOST
        json={
            'token': token,
            'command': 'presigned_url',
            'schema_type': 'customers',          # or 'agents'
            'filename': 'customers.csv'
        },
    )
    tuplex = (response.json()['signed_url'], response.json()['file_path'])
```

```
    return tuplex

# PUT /presigned_url          Upload source file to s3(minio)
def upload_profile_s3(presigned_url):
    response = requests.put(
        presigned_url,
        data=open(PROFILE_PATH, 'rb').read()
    )
    return response

# POST /schemas              Upload from s3 to the GPR platform
def upload_from_s3(token, file_path):
    response = requests.post(
        '%s/api/v2.0/schemas' % HOST,
        data = {
            'token': token,
            'schema_type': 'customers',
            'command': 'finish_s3_upload',
            's3_separator': 'TAB',
            's3_filename': file_path
        }
    )
    return response.json()

def upload_from_s3(token, file_path):
    response = requests.post(
        '%s/api/v2.0/schemas' % HOST,
        data = {
            'token': token,
            'schema_type': 'customers',
            'command': 'finish_s3_upload',
            's3_separator': 'TAB',
            's3_filename': file_path
        }
    )
    return response.json()

# PUT /schemas (create schema from a sample)
def create_schema_from_template(token):
    response = requests.put(
        '%s/api/v2.0/schemas' % (HOST),
        json={
            'token': token,
            'schema_type': 'agents',
            'sample_data': {
                'string_field': 'string',
                'integer_field': 1,
                'boolean_field': False
            }
        }
    )
    return response.json()

# PUT /schemas (create final schema)
def create_final_schema(token):
    response = requests.put(
        '%s/api/v2.0/schemas' % (HOST),
        json={
            'token': token,
            'schema_type': 'agents',
            'schema_info': [
```

```
        {
            'name': 'boolean_field',
            'type': 'boolean'
        },
        {
            'name': 'string_field',
            'type': 'string',
            'is_id': True
        },
        {
            'name': 'integer_field',
            'type': 'integer'
        }
    ]
}
)
return response.json()

# POST /schemas Sync, Cancel sync, Accept schema
def manage_schemas(token):
    response = requests.post(
        '%s/api/v2.0/schemas' % HOST,
        data={
            'token': token,
            'command': 'accept_sync',
            'schema_type': 'agents'
        }
    )
    return response.json()

# DELETE /schemas Delete schema
def delete_schema(token):
    response = requests.delete(
        '%s/api/v2.0/schemas' % (HOST),
        data={
            'token': token,
            'schema_type': 'agents'
        }
    )
    return response.json()

# GET /schemas Get schema details
def get_schema_details(token):
    response = requests.get(
        '%s/api/v2.0/schemas' % HOST,
        json={
            'token': token,
            'schema_type': 'agents'
        }
    )
    return response.json()

# GET /schemas Get schema details
def get_schema_details(token):
    response = requests.get(
        '%s/api/v2.0/schemas' % HOST,
        json={
            'token': token,
            'schema_type': 'agents'
        }
    )
    return response.json()
```

```
# POST /schemas (change fields visibility)
def change_schema_visibility(token):
    response = requests.post(
        '%s/api/v2.0/schemas' % HOST,
        json={
            'token': token,
            'command': 'fields_visibility',
            'schema_type': 'agents',
            'visible_fields': ['Agent_First']
        }
    )
    return response.json()
```

Agents Examples

```
# POST /agents (create using file)
def upload_agents_csv(token):
    response = requests.post(
        '%s/api/v2.0/agents' % HOST,
        data={
            'token': token,
            'sep': 'COMMA',
            'encoding': 'shift-jis'
        },
        files={
            'csv_file': open(AGENTS_FILE, 'rb'),
        }
    )
    return response.json()

# POST /agents (create using json)
def upload_agents_batch(token):
    response = requests.put(
        '%s/api/v2.0/agents' % (HOST),
        json={
            'token': token,
            'batch_data': [
                {
                    'Agent_AgentID': '1000',
                    'Agent_First': 'Jane',
                    'Agent_Last': 'Doe'
                }
            ],
            'return_objects': True,
            'include_complex_attributes': True
        }
    )
    return response.json()

# GET /agents Get Agent details
def get_agent_details_id(token):
    response = requests.get(
        '%s/api/v2.0/agents' % HOST,
        json={
            'token': token,
            'Agent_AgentID': '1000',
            'include_complex_attributes': True
        }
    )
    return response.json()

def get_agent_details_search(token):
```

```
response = requests.get(
    '%s/api/v2.0/agents' % HOST,
    json={
        'token': token,
        'filter': 'Agent_Experience>10&Agent_Location_Country=US',
        'search': 'John',
        'include_complex_attributes': True
    }
)
return response.json()

#PUT /agents Edit agent record
def edit_agent(token):
    response = requests.put(
        '%s/api/v2.0/agents' % (HOST),
        json={
            'token': token,
            'batch_data': [
                {
                    'Agent_AgentID': '1000',
                    'Agent_First': 'Joe',
                    'Agent_Last': 'Doe'
                }
            ]
        }
    )
    return response.json()

#DELETE /agents Delete agent record.
def delete_agent(token):
    response = requests.delete(
        '%s/api/v2.0/agents' % (HOST),
        data={
            'token': token,
            'Agent_AgentID': '1000'
        }
    )
    return response.json()

# POST /agents/compute_cardinalities
def agent_cardinalities(token):
    response = requests.post(
        '%s/api/v2.0/agents/compute_cardinalities' % HOST,
        data={
            'token': token
        }
    )
    return response.json()
```

Customers Examples

```
#POST /customers (create using file)
def upload_customers_csv(token):
    response = requests.post(
        '%s/api/v2.0/customers' % HOST,
        data={
            'token': token,
            'sep': 'TAB',
            'encoding': 'shift-jis'
        },
        files={
            'csv_file': open(SMALL_FILE, 'rb'),
        }
    )
```

```
)
return response.json()

#POST /customers (create using JSON)
def upload_customers_batch(token):
    response = requests.put(
        '%s/api/v2.0/customers' % (HOST),
        json={
            'token': token,
            'batch_data': [
                {
                    'Customer_CustomerID': 'eafa71e9-85b9-4fa1-9395-2f6f3d7f1d65',
                    'Customer_AccountValue': 759696320,
                    'Customer_Age': 37,
                    'Customer_AgeBucket': 1,
                    'Customer_Location_Country': 'Canada',
                    'Customer_Location_State': 'BC',
                    'Customer_Intent': 'Opening Account',
                    'Customer_Income': 798884,
                    'Customer_Escalations': 3
                }
            ],
            'return_objects': True,
            'include_complex_attributes': True
        }
    )
    return response.json()

# GET /customers Get Customer details
def get_customer_details_id(token):
    response = requests.get(
        '%s/api/v2.0/customers' % HOST,
        json={
            'token': token,
            'Customer_CustomerID': 'eafa71e9-85b9-4fa1-9395-2f6f3d7f1d65',
            'include_complex_attributes': True
        }
    )
    return response.json()

def get_customer_details_search(token):
    response = requests.get(
        '%s/api/v2.0/customers' % HOST,
        json={
            'token': token,
            'filter': 'Customer_Age>20&Customer_Location_Country=US',
            'search': 'male',
            'include_complex_attributes': True
        }
    )
    return response.json()

#PUT /customers Edit Customer record
def edit_customer(token):
    response = requests.put(
        '%s/api/v2.0/customers' % (HOST),
        json={
            'token': token,
            'batch_data': [
                {
                    'Customer_CustomerID': 'eafa71e9-85b9-4fa1-9395-2f6f3d7f1d65',
                    'Customer_Age': '35'
                }
            ]
        }
    )
```

```
)
return response.json()

#DELETE /customers Delete customer record.
def delete_customer(token):
    response = requests.delete(
        '%s/api/v2.0/customers' % (HOST),
        data={
            'token': token,
            'Customer_CustomerID': 'eafa71e9-85b9-4fa1-9395-2f6f3d7f1d65'
        }
    )
    return response.json()

# POST /customers/compute_cardinalities
def customer_cardinalities(token):
    response = requests.post(
        '%s/api/v2.0/customers/compute_cardinalities' % HOST,
        data={
            'token': token
        }
    )
    return response.json()
```

Predictors Examples

```
# POST /predictors (simple)
def create_predictor(token):
    response = requests.post(
        '%s/api/v2.0/predictors' % HOST,
        json={
            "token": token,
            "name": "TestPredictor2",
            "from_dt": int(time.mktime((2018, 1, 31, 0, 0, 0, 0, 0, 0))), # Dummy range to
fit my sample set
            "to_dt": int(time.mktime((2020, 1, 31, 0, 0, 0, 0, 0, 0))),
            "action_type": "agents",
            "context_type": "customers",
            "dataset": DATASET_ID,
            "metric": "NPS",
            "action_id_expression": "Agent_AgentID",
            "kpi_type": "Sales",
            "action_features_schema": [
                {
                    "label": "Agent_Skill4",
                    "type": "string",
                    "field_expr": "Agent_Skill4"
                },
                {
                    "label": "Agent_SalesConversionRate",
                    "type": "integer",
                    "field_expr": "Agent_SalesConversionRate"
                },
                {
                    "label": "Agent_Skill4_Level",
                    "type": "integer",
                    "field_expr": "Agent_Skill4_Level"
                }
            ],
            "context_features_schema": [
                {
                    "label": "Customer_Age",
```

```
        "type": "integer",
        "field_expr": "Customer_Age"
    },
    {
        "label": "Customer_Location_Country",
        "type": "string",
        "field_expr": "Customer_Location_Country"
    },
    {
        "label": "Customer_Intent",
        "type": "string",
        "field_expr": "Customer_Intent"
    }
],
}
)
return response.json()

# POST /predictors (composite)
def create_composite_predictor(token):
    response = requests.post(
        '%s/api/v2.0/predictors' % HOST,
        json={
            "token": token,
            "name": "composite_predictor",
            "predictor_type": "Composite Predictor",
            "predictors_list": [PREDICTOR_ID, PREDICTOR_ID2],
            "raw_expression": "($predictor1Name1 + $predictor2Name)/2"
        }
    )
    return response.json()

# GET /predictors Get all predictors details
def get_predictors_info(token):
    response = requests.get(
        '%s/api/v2.0/predictors' % (HOST),
        json={
            'token': token,
            'fields2show': ['name', 'id']
        }
    )
    return response.json()

# GET /predictors/{id} Get predictor details
def get_predictor_info(token):
    response = requests.get(
        '%s/api/v2.0/predictors/%s' % (HOST, PREDICTOR_ID),
        json={
            'token': token,
        }
    )
    return response.json()

# PUT /predictors/{id} Edit predictor record
def edit_predictor(token):
    response = requests.put(
        '%s/api/v2.0/predictors/%s' % (HOST, PREDICTOR_ID),
        json={
            'token': token,
            'name': 'RenamedTestPredictor'
        }
    )
    return response.json()
```

```
#DELETE /predictors/{id} Delete predictor record.
def delete_predictor(token):
    response = requests.delete(
        '%s/api/v2.0/predictors/%s' % (HOST, PREDICTOR_ID2),
        data={
            'token': token
        }
    )
    return response.json()

# PUT /predictors/{id}/generate_data Generate predictor data.
def generate_training_data(token, from_timestamp, to_timestamp):
    response = requests.put(
        '%s/api/v2.0/predictors/%s' % (HOST, PREDICTOR_ID),
        json={
            'command': 'generate_data',
            'from_dt': from_timestamp,
            'to_dt': to_timestamp,
            'token': token
        }
    )
    return response.json()

# PUT /predictors/{id}/purge_data Purge predictor data
def purge_training_data(token):
    response = requests.put(
        '%s/api/v2.0/predictors/%s' % (HOST, PREDICTOR_ID),
        json={
            'command': 'purge_data',
            'token': token
        }
    )
    return response.json()

# GET /predictors/{id}/check_status Check status of predictor generate and purge jobs
def check_predictor_status(token):
    response = requests.get(
        '%s/api/v2.0/predictors/%s/check_status' % (HOST, PREDICTOR_ID),
        json={
            'token': token
        }
    )
    return response.json()

# GET /predictors/{id}/data Get predictor data
def get_predictor_data(token):
    response = requests.get(
        '%s/api/v2.0/predictors/%s/data' % (HOST, PREDICTOR_ID),
        json={
            'token': token,
            'filter': 'act.Agent_Skill4_Level=1&ctx.Customer_Age>30'
        }
    )
    return response.json()

# POST /predictors/{id}/copy_predictor Copy predictor
def copy_predictor(token):
    response = requests.post(
        '%s/api/v2.0/predictors/%s/copy_predictor' % (HOST, PREDICTOR_ID),
        json={
            'token': token
        }
    )
```

```
    return response.json()

# POST /predictors/{id}/score Score actions for predictor context
def predictor_score(token):
    response = requests.post(
        '%s/api/v2.0/predictors/%s/copy_predictor' % (HOST, PREDICTOR_ID),
        json={
            'token': token,
            'context': {
                'Customer_Location_Country': 'US'
            },
            'action_filters': "Agent_Skill4_Level in ['4','5']",
            'format_as_map': True,
            'warnings': True
        }
    )
    return response.json()

# POST /predictors/{id}/reset Reset predictor
def reset_predictor(token):
    response = requests.post(
        '%s/api/v2.0/predictors/%s/reset' % (HOST, PREDICTOR_ID),
        json={
            'token': token
        }
    )
    return response.json()

# POST /predictors/{id}/feedback Predictor feedback
def predictor_feedback(token):
    response = requests.post(
        '%s/api/v2.0/predictors/%s/feedback' % (HOST, PREDICTOR_ID),
        json={
            'token': token,
            'action': {
                'action_id': 'UUID_2',
                'skill': 'testing2',
                'age': '28',
                'fluency': 'good2',
                'seniority': 'veteran2'
            },
            'context': {
                'age': '36',
                'gender': 'M',
                'location': 'San Francisco2',
                'n_subs': '16',
                'intention': 'closing an account',
                'seniority': 'ancient2'
            },
            'reward': 0
        }
    )
    return response.json()

# POST /ab_testing/join A/B Test predictor
def ab_test(token):
    response = requests.post(
        '%s/api/v2.0/ab_testing/join' % (HOST),
        json={
            'token': token,
            'prpPredictor': 'RenamedTestPredictor',
            'from_date': '01/31/2018',
            'to_date': '01/31/2020'
        }
    )
```

```
    }  
  )  
  return response.json()
```

Score Log Examples

```
# POST /score_log Log your predictor score log data  
def score_log(token):  
    response = requests.post(  
        '%s/api/v2.0/score_log' % (HOST),  
        json={  
            'token': token,  
            'prpPredictor': 'RenamedTestPredictor',  
            'context': {  
                'Customer_Location_Country': 'US'  
            },  
            'action_filters': "Agent_Skill4_Level in ['4','5']",  
            'format_as_map': True,  
            'warnings': True  
        }  
    )  
    return response.json()  
  
# GET /score_log View your previously logged score data  
def get_score_log(token):  
    response = requests.get(  
        '%s/api/v2.0/score_log' % (HOST),  
        json={  
            'token': token,  
            'prpPredictor': 'RenamedTestPredictor',  
            'context': {  
                'Customer_Location_Country': 'US'  
            }  
        }  
    )  
    return response.json()
```

Models (PredictorsModels) Examples

```
# POST /predictor_models Create predictor model.  
def create_predictive_model(token):  
    response = requests.post(  
        '%s/api/v2.0/predictor_models' % HOST,  
        json={  
            "token": token,  
            "display_name": "TestModel_DISJOINT",  
            "predictor_id": PREDICTOR_ID,  
            "model_type": "DISJOINT",  
            "context_features": [  
                {  
                    "label": "Customer_Age",  
                    "type": "integer",  
                    "field_expr": "Customer_Age"  
                },  
                {  
                    "label": "Customer_Location_Country",  
                    "type": "string",  
                    "field_expr": "Customer_Location_Country"  
                },  
                {  
                    "label": "Customer_Age",  
                    "type": "integer",  
                    "field_expr": "Customer_Age"  
                }  
            ]  
        }  
    )  
    return response.json()
```

```
        "label": "Customer_Intent",
        "type": "string",
        "field_expr": "Customer_Intent"
    }
],
"action_features": [
    {
        "label": "Agent_Skill4",
        "type": "string",
        "field_expr": "Agent_Skill4"
    },
    {
        "label": "Agent_SalesConversionRate",
        "type": "integer",
        "field_expr": "Agent_SalesConversionRate"
    },
    {
        "label": "Agent_Skill4_Level",
        "type": "integer",
        "field_expr": "Agent_Skill4_Level"
    }
],
"train_data_percentage": 80,
)
return response.json()

# GET /predictor_models Get all predictor Models details
def get_models_info(token):
    response = requests.get(
        '%s/api/v2.0/predictor_models' % (HOST),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()

# GET /predictor_models/{id} Get predictor Model details
def get_model_info(token):
    response = requests.get(
        '%s/api/v2.0/predictor_models/%s' % (HOST, MODEL_ID),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()

# PUT /predictor_models/{id} Edit predictor model record
def edit_model(token):
    response = requests.put(
        '%s/api/v2.0/predictor_models/%s' % (HOST, MODEL_ID),
        json={
            "token": token,
            "display_name": "NewTestModel",
            "description": "Test model",
            "predictor_id": PREDICTOR_ID,
            "model_type": "HYBRID",
            "context_features": [
                {
                    "label": "Customer_Age",
                    "type": "integer",
```

```
        "field_expr": "Customer_Age"
    },
    {
        "label": "Customer_Location_Country",
        "type": "string",
        "field_expr": "Customer_Location_Country"
    },
    {
        "label": "Customer_Intent",
        "type": "string",
        "field_expr": "Customer_Intent"
    }
],
"action_features": [
    {
        "label": "Agent_Skill4",
        "type": "string",
        "field_expr": "Agent_Skill4"
    },
    {
        "label": "Agent_SalesConversionRate",
        "type": "integer",
        "field_expr": "Agent_SalesConversionRate"
    },
    {
        "label": "Agent_Skill4_Level",
        "type": "integer",
        "field_expr": "Agent_Skill4_Level"
    }
],
"train_data_percentage": 80,
}
)
return response.json()

# DELETE /predictor_models/{id} Delete predictor model
def delete_model(token):
    response = requests.delete(
        '%s/api/v2.0/predictor_models/%s' % (HOST, MODEL_ID3),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID,
            'hard': True
        }
    )
    return response.json()

# DELETE /predictor_models (several models) Delete predictor models in batch
def delete_models_batch(token):
    response = requests.delete(
        '%s/api/v2.0/predictor_models' % (HOST),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID,
            'hard': True,
            'model_id': [
                MODEL_ID1,
                MODEL_ID2
            ]
        }
    )
    return response.json()
```

```
# PUT /predictor_models/{id}/train Train predictor model
def train_model(token):
    response = requests.put(
        '%s/api/v2.0/predictor_models/%s/train' % (HOST, MODEL_ID),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()

# PUT /predictor_models/{id}/retrain Retrain predictor model
def retrain_model(token):
    response = requests.put(
        '%s/api/v2.0/predictor_models/%s/retrain' % (HOST, MODEL_ID),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()

# PUT /predictor_models/{id}/activate Activate predictor model
def activate_model(token):
    response = requests.put(
        '%s/api/v2.0/predictor_models/%s/activate' % (HOST, MODEL_ID),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()

# PUT /predictor_models/{id}/deactivate Deactivate predictor model
def deactivate_model(token):
    response = requests.put(
        '%s/api/v2.0/predictor_models/%s/deactivate' % (HOST, MODEL_ID),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()

# PUT /predictor_models/{id}/copy Copy predictor model
def copy_model(token):
    response = requests.put(
        '%s/api/v2.0/predictor_models/%s/copy' % (HOST, MODEL_ID),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()
```

Analysis Examples

```
# GET /analysis Get analysis reports data
def get_analysis(token):
    response = requests.get(
        '%s/api/v2.0/analysis' % HOST,
        json={
```

```
        'token': token
    }
)
return response.json()

# GET /analysis/{id} Get analysis report data by ID
def get_analysis_by_id(token):
    response = requests.get(
        '%s/api/v2.0/analysis/%s' % (HOST, REPORT_ID),
        json={
            'token': token
        }
    )
    return response.json()

# DELETE /analysis/{id} Delete analysis report
def delete_analysis_report(token):
    response = requests.delete(
        '%s/api/v2.0/analysis/%s' % (HOST, REPORT_ID),
        json={
            'token': token
        }
    )
    return response.json()

# POST /analysis/datasets/feature_analysis Generate Feature Analysis report for dataset
def generate_fa_report_dataset(token):
    response = requests.post(
        '%s/api/v2.0/analysis/datasets/feature_analysis' % HOST,
        json={
            'token': token,
            'dataset_id': DATASET_ID,
            'target_metric': 'HandleTime'
        }
    )
    return response.json()

# POST /analysis/predictors/feature_analysis Generate Feature Analysis report for predictor
def generate_fa_report_predictor(token):
    response = requests.post(
        '%s/api/v2.0/analysis/predictors/feature_analysis' % HOST,
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()

# POST /analysis/predictors/lift_estimation Generate Lift Estimation report for predictor
def generate_le_report(token):
    response = requests.post(
        '%s/api/v2.0/analysis/predictors/lift_estimation' % HOST,
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID,
            'model_id': MODEL_ID
        }
    )
    return response.json()
```

Index Management Examples

```
# GET /index_management Get existing indexes on a collection
def get_indexes(token):
    response = requests.get(
        '%s/api/v2.0/index_management' % HOST,
        json={
            'token': token,
            'coll_type': 'agents'
        }
    )
    return response.json()

# PUT /index_management Edit indexes on a collection
def edit_indexes(token):
    response = requests.put(
        '%s/api/v2.0/index_management' % HOST,
        json={
            "token": token,
            "coll_type": "agents",
            "fields": [
                "Agent_Status",
                "Agent_Location_Country"
            ]
        }
    )
    return response.json()

# PUT /index_management Delete indexes on a collection
def delete_indexes(token):
    response = requests.delete(
        '%s/api/v2.0/index_management' % HOST,
        json={
            "token": token,
            "coll_type": "agents",
            "fields": [
                "Agent_Status",
                "Agent_Location_Country"
            ]
        }
    )
    return response.json()
```

PURGE APIs

```
# GET /purge/{purge_job_id} Get purge job details
def get_purge_status(token, purge_job_id):
    response = requests.get(
        '%s/api/v2.0/purge/%s' % (HOST, purge_job_id),
        json={
            'token': token
        }
    )
    return response.json()

# POST /purge/agents Add a purge job for agents
def purge_agents(token):
    response = requests.post(
        '%s/api/v2.0/purge/agents' % HOST,
        json={
            'token': token,

```

```
        'data_filter': '(Agent_Age>30)'
    }
)
return response.json()

# POST /purge/customers Add a purge job for customers
def purge_customers(token):
    response = requests.post(
        '%s/api/v2.0/purge/customers' % HOST,
        json={
            'token': token,
            'data_filter': '(Customer_Income<200000)'
        }
    )
    return response.json()

# POST /purge/datasets/{id} Add a purge job for dataset with given id
def purge_dataset(token):
    response = requests.post(
        '%s/api/v2.0/purge/datasets/%s' % (HOST, DATASET_ID),
        json={
            'token': token,
            'data_filter': '(CSAT=5)'
        }
    )
    return response.json()

# POST /purge/predictors/{id} Add a purge job for predictor with given id
def purge_predictor(token):
    response = requests.post(
        '%s/api/v2.0/purge/predictors/%s' % (HOST, PREDICTOR_ID),
        json={
            'token': token,
            'data_filter': '(ctx.Customer_Age=56)'
        }
    )
    return response.json()

def train_model(token):
    response = requests.put(
        '%s/api/v2.0/predictor_models/%s/train' % (HOST, MODEL_ID),
        json={
            'token': token,
            'predictor_id': PREDICTOR_ID
        }
    )
    return response.json()
```

Predictor Scoring Example

```
# POST /predictors/{id}/score Score actions for predictor context
def get_scores(token):
    response = requests.post(
        '%s/api/v2.0/predictors/%s/score' % (HOST, PREDICTOR_ID),
        json={
            'token': token,
            'context': {
                'Customer_Location_Country': 'US'
            },
            'action_filters': "Agent_Skill4_Level in ['3','4','5']",
            'format_as_map': True,
            'warnings': True
        }
    )
```

```
    }  
  )  
  return response.json()
```