MARIGOLD™

API Technical Guide: Search Records

Cheetah Messaging

Table of Contents

1 Introduction

Purpose

The purpose of this document is to provide an overview of the SEARCH RECORDS API endpoint within the Cheetah Messaging platform. This document discusses the intended use of the SEARCH RECORDS endpoint, and provides technical details for how to implement the endpoint.

Overview

The SEARCH RECORDS endpoint provides a simple and flexible database query tool, similar to the Record Lookup screen within the Messaging application. The endpoint allows you to submit a variety of different search criteria in order to identify the desired record (or records). You also have the ability to customize the response message, by indicating what fields you want to see returned in the response.

Note

This endpoint returns a maximum of 250 records in the response message.

This endpoint requires authentication using OAuth 2.0, and supports JSON and XML messages.

The URLs for this endpoint are:

- **North America**: https://api.eccmp.com/services2/api/SearchRecords
- **Europe**: https://api.ccmp.eu/services2/api/SearchRecords
- **Japan**: https://api.marketingsuite.jp/services2/api/SearchRecords

Methods

The SEARCH RECORDS endpoint supports the following HTTP methods:

  - **GET**: Retrieve a single record from a specified table by providing the Record ID (i.e., the Primary Key ID); you can specify the table by either:
- The table's Object Reference ID
- The table's system name
  - **GET**: Retrieve records from a specified table by providing a value, mathematical operator, and field to search; you can specify the table by either:
- The table's Object Reference ID
- The table's system name

Authentication

Access to the SEARCH RECORDS endpoint requires that you first be authenticated within the platform. Within Messaging, authentication is handled by OAuth 2.0. To authenticate with OAuth 2.0, you must first obtain a "Consumer Key" and a "Consumer Secret." Both of these values are managed at the user level, and can be obtained from within the Messaging application.

Next, you'll use your Consumer Key and Consumer Secret to request a "token." A token is a text string that, when provided in a request message, will allow the user access to the requested service. Tokens are valid only for a certain period of time.

For more details on how to authenticate your API request, please see the *Messaging: API How-to Guide*.

2 Retrieve Records

Overview

This section describes how to retrieve one or more records from a table in your database using the SEARCH RECORDS endpoint.

Retrieve a Record

Using a GET method, you can retrieve all of the requested information about a single record in a single table by specifying the record's Record ID.

recordId

This integer parameter is required.

The **recordId** parameter represents the Record ID (or "Primary Key ID") of the desired record.

When submitting a GET request to the SEARCH RECORDS endpoint, the request message must include the Record ID as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?recordId=2311113&vie
wId=1002&prop=name_first,name_last
```

viewId / viewName

One of the these two parameters is required.

The SEARCH RECORDS endpoint is limited to searching only one table per request message. The endpoint supports two different ways of indicating the table you want to search (by Object Reference ID or by Table System Name). You must provide one of these two identifiers in the request message.

The **viewId** parameter represents the **Object Reference ID** of the table you want to search. The **viewName** parameter represents the **Table System Name** of the table you want to search.

When submitting a GET request to the SEARCH RECORDS endpoint, the request message must include the Object Reference ID or Table System Name as a query type parameter within the URL.

Below is an example of searching by the table's Object Reference ID:

```
https://api.eccmp.com/services2/api/SearchRecords?recordId=2311113&vie
wId=1002&prop=name_first,name_last
```

Below is an example of searching by the Table System Name:

```
https://api.eccmp.com/services2/api/SearchRecords?recordId=2311113&vie
wName=recipient&prop=name_first,name_last
```

prop

This string parameter is required.

The SEARCH RECORD endpoint lets you customize the response message by indicating what field (or fields) you want returned in the response. The **prop** parameter must include at least one **Column Name** value for a field you want returned. If you want to return multiple fields, separate the Column Name values with a comma (with no spaces).

When submitting a GET request to the SEARCH RECORDS endpoint, the request message must include the desired Column Name (or Names) as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?recordId=2311113&vie
wId=1002&prop=name_first,name_last
```

Search for Records

Using a GET method, you can provide search criteria, a mathematical operator (equal to, greater than, starts with, etc.), and the column that you want to search. The system will return up to 250 records that match the provided search conditions.

For example, let's say you wanted to look up the first and last names of all your consumers in your Recipient table that have a Gmail email address. Your search logic would look like this:

- Find records in Recipient table that contain the value "gmail" anywhere within the "email" field

viewId / viewName

One of the these two parameters is required.

The SEARCH RECORDS endpoint is limited to searching only one table per request message. The endpoint supports two different ways of indicating the table you want to search (by Object Reference ID or by Table System Name). You must provide one of these two identifiers in the request message.

The **viewId** parameter represents the **Object Reference ID** of the table you want to search. The **viewName** parameter represents the **Table System Name** of the table you want to search.

When submitting a GET request to the SEARCH RECORDS endpoint, the request message must include the Object Reference ID or Table System Name as a query type parameter within the URL.

Below is an example of searching by the table's Object Reference ID:

```
https://api.eccmp.com/services2/api/SearchRecords?viewId=1002&prop=nam
e_first,name_last,email&columnName=email&operation=like&param=gmail&co
unt=100&page=1
```

Below is an example of searching by the Table System Name:

```
https://api.eccmp.com/services2/api/SearchRecords?viewName=recipient&p
rop=name_first,name_last,email&columnName=email&operation=like&param=g
mail&count=100&page=1
```

prop

This string parameter is required.

The SEARCH RECORD endpoint lets you customize the response message by indicating what field (or fields) you want returned in the response. The **prop** parameter must include at least one Column Name value for a field you want returned. If you want to return multiple fields, separate the Column Name values with a comma (with no spaces).

When submitting a GET request to the SEARCH RECORDS endpoint, the request message must include the desired Column Name (or Names) as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?viewId=1002&prop=nam
e_first,name_last,email&columnName=email&operation=like&param=gmail&co
unt=100&page=1
```

columnName

This string parameter is required.

The **columnName** parameter represents the Column Name for the field that you want to search. The SEARCH RECORDS endpoint is limited to searching only a single field per request message.

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?viewId=1002&prop=nam
e_first,name_last,email&columnName=email&operation=like&param=gmail&co
unt=100&page=1
```

operation

This string parameter is required.

The **operation** parameter represents the mathematical operator for your search condition. This parameter supports the following valid values:

- "=" -- Equal to
- "lt" -- Less than
- "lte" -- Less than or equal to
- "gt" -- Greater than
- "gte" -- Greater than or equal to
- "like" -- Contains this value
- "starts" -- Starts with this value

- "ends" -- Ends with this value
- "between" -- Between two values (this operator requires you to provide two values in the **param** parameter)

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?viewId=1002&prop=nam
e_first,name_last,email&columnName=email&operation=like&param=gmail&co
unt=100&page=1
```

param

This string parameter is required.

The param parameter represents the value for which you are searching. The system will search for this value within the field indicated in **columnName**, using the mathematical operator indicated in **operation**.

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?viewId=1002&prop=nam
e_first,name_last,email&columnName=email&operation=like&param=gmail&co
unt=100&page=1
```

If you're using the value of "between" in the **operation** parameter, then you must provide two values in **param**, and separate the two values with a comma.

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?viewId=1002&prop=nam
e_first,name_last,email&columnName=purchase&operation=between&param=50
,100
```

count / page

These two integer parameters are optional; if you don't provide a value for these parameters, the system will default them to blank.

These parameters are used to control how many records, and which records, should be included within the response message. By default, the system will return all records that match the search criteria, sorted by Record ID in ascending order.

However, you can use the **count** and **page** parameters to limit the response message to only a certain quantity and selection of records.

The **count** parameter will split the response message up into "pages" of the designated size. The **page** parameter then tells the system which page you want to see in the response message.

For example, let's say your search conditions resulted in 500 matches, and you want to see only the first 100 in the response message. You could set **count** to "100" so that the system splits the response into pages of 100 records each. You would also set **page** to "1," so that you receive only the first page of records in the response.

For example:

```
https://api.eccmp.com/services2/api/SearchRecords?viewId=1002&prop=nam
e_first,name_last,email&columnName=email&operation=like&param=gmail&co
unt=100&page=1
```

3 Response

This section describes the possible response messages sent back from the SEARCH RECORDS endpoint.

Success

A successful response to a GET method will generate a response code of "200," followed by the record (or records) that matched the search conditions in the request message.

For each record in the response, the message will include the Primary Key ID, along with the field (or fields) indicated in the request message.

Note

This endpoint returns a maximum of 250 records in the response message.

If your search conditions resulted in no matches found, the endpoint will generate a response code of "404," with the message "No results for input."

Errors

If Messaging encounters a problem with a SEARCH RECORDS request message, the platform will send an "error" message with details of the problem. Below is a list of error codes and their descriptions.

Response Code	Error message	Description
400	Search column does not exist.	Field specified in columnName is invalid.

Respons e Code	Error message	Description
400	Invalid operation provided	Operator specified in **operation** is invalid.

4 Sample Messages

Request #1

In this sample message, the user is searching the Recipient table for a specific Record ID (350).

The user has indicated that the response should contain the 'first name,' 'last name,' and 'email' address fields.

```
https://api.eccmp.com/services2/api/SearchRecords?recordId=350&viewNam
e=recipient&prop=name_first,name_last,email
```

Response #1

This sample message shows the response to the above request message. The response includes the Record ID (i.e., the Primary Key ID), followed by the fields indicated in the request message.

```
{
  "id": 350,
  "properties": [
    {
      "propName": "name_first",
      "value": "John"
    },
    {
      "propName": "name_last",
      "value": "Smith"
    },
    {
      "propName": "email",
      "value": "john.smith@cheetahdigital.com"
    }
  ]
}
```

Request #2

In this sample message, the user is searching the Recipient table for all consumers who have a "cheetahdigital" email address.

The user has indicated that the response should contain the 'first name,' 'last name,' and 'email' fields.

```
https://api.eccmp.com/services2/api/SearchRecords?viewName=recipient&p
rop=name_last,name_first,email&columnName=email&operation=like&param=c
heetahdigital
```

Response #2

This sample message shows the response to the above request message. For each record that met the search criteria, the response includes the Record ID, followed by the fields indicated in the request message.

```
[
  {
    "id": 2296513,
    "properties": [
      {
        "propName": "name_last",
        "value": "John"
      },
      {
        "propName": "name_first",
        "value": "Smith"
      },
      {
        "propName": "email",
        "value": "john.smith@cheetahdigital.com"
      }
    ]
  },
  {
    "id": 2309775,
    "properties": [
      {
        "propName": "name_last",
        "value": "Bruce"
      },
      {
        "propName": "name_first",
        "value": "Wayne"
      },
      {
        "propName": "email",
        "value": "bruce.wayne@cheetahdigital.com"
      }
    ]
  },
  {
    "id": 2309789,
    "properties": [
      {
        "propName": "name_last",
        "value": "Mary"
      },
      {
        "propName": "name_first",
        "value": "Jones"
      },
      {
        "propName": "email",
        "value": "mary.jones@cheetahdigital.com"
      }
    ]
  }
]
```

5 Appendix A -- Identifiers

Messaging uses several different types of IDs when referencing assets, such as tables, fields, folders, Filters, and so forth. This appendix describes these different types of IDs, and provides steps on how to look up the value of an ID.

Record ID

The Record ID is a unique, system-generated identifier for every record in every table in your database. This identifier is also referred to as the Primary Key ID.

The value for this identifier can be found on the Record Lookup screen within the Messaging application:

1. From the System Tray, navigate to *Data Management > Management > Record Lookup.*
2. Select a table, and enter the search criteria. Click **Search**. The search results are displayed.
3. Click the "Edit" link next to the desired record. The Record Details screen is displayed, showing the values for every field in this table, including the Primary Key ID. The default name of the Primary Key ID field is "pk_<table name>_id."

You can also look up a Record ID using the **Search for Records** version of the SEARCH RECORDS endpoint. Enter your search conditions in the request message; the resulting response message will include the Record ID for all records that matched the search conditions. The Record ID can be found in the **id** parameter. For example:

```
{
  "id": 350,
  "properties": [
    {
      "propName": "name_first",
      "value": "John"
    },
    {
      "propName": "name_last",
      "value": "Smith"
    },
    {
      "propName": "email",
      "value": "john.smith@cheetahdigital.com"
    }
  ]
}
```

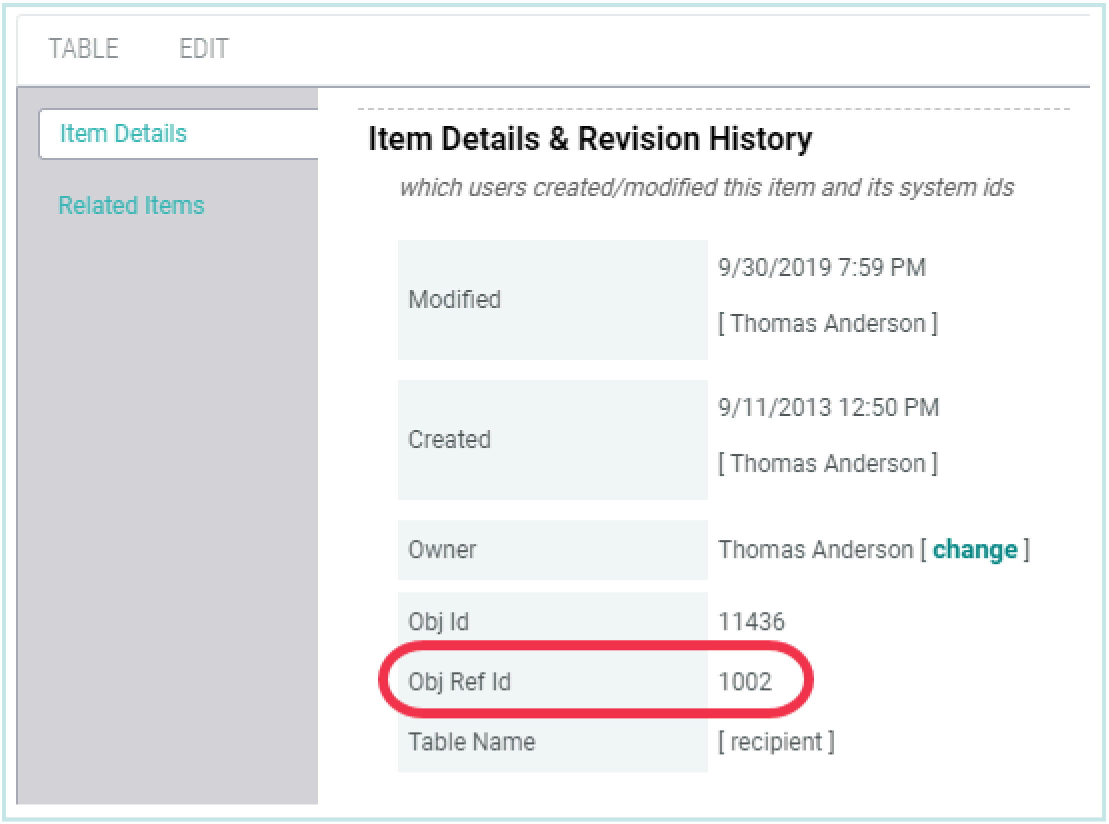
Object Reference ID

The Object Reference ID is a system-generated identifier for every item and asset in your account.

For Tables, the value for this identifier can be found within the Messaging application, or by using the TABLE endpoint.

To look up the Object Reference ID within the application:

1. From the System Tray, navigate to *Data Management* > *Structures* > *Tables*.
2. In the Tool Ribbon, click the Table tab.
3. The "Item Details" screen is displayed. The Object Reference ID is listed on this screen.



Item Details & Revision History

which users created/modified this item and its system ids

Modified	9/30/2019 7:59 PM [Thomas Anderson]
Created	9/11/2013 12:50 PM [Thomas Anderson]
Owner	Thomas Anderson [change]
Obj Id	11436
Obj Ref Id	1002
Table Name	[recipient]

Optionally, you can use the **Retrieve All Tables** method of the TABLE endpoint. In the response message, the Object Reference ID is contained within the **viewId** parameter. For example:

```
{
  "viewId": 1002,
  "viewName": "Recipient",
  "entityId": 100,
  "tableName": "recipient"
}
```

Table System Name

Tables in Messaging have a user-friendly display name, and a corresponding system-generated name. For example, let's say you have a table with a display name of "Order Item Table." By default, the platform will automatically generate the system name for this table as "order_item_table." When you're using the SEARCH RECORDS endpoint to search by table name, you must use the table's system name.

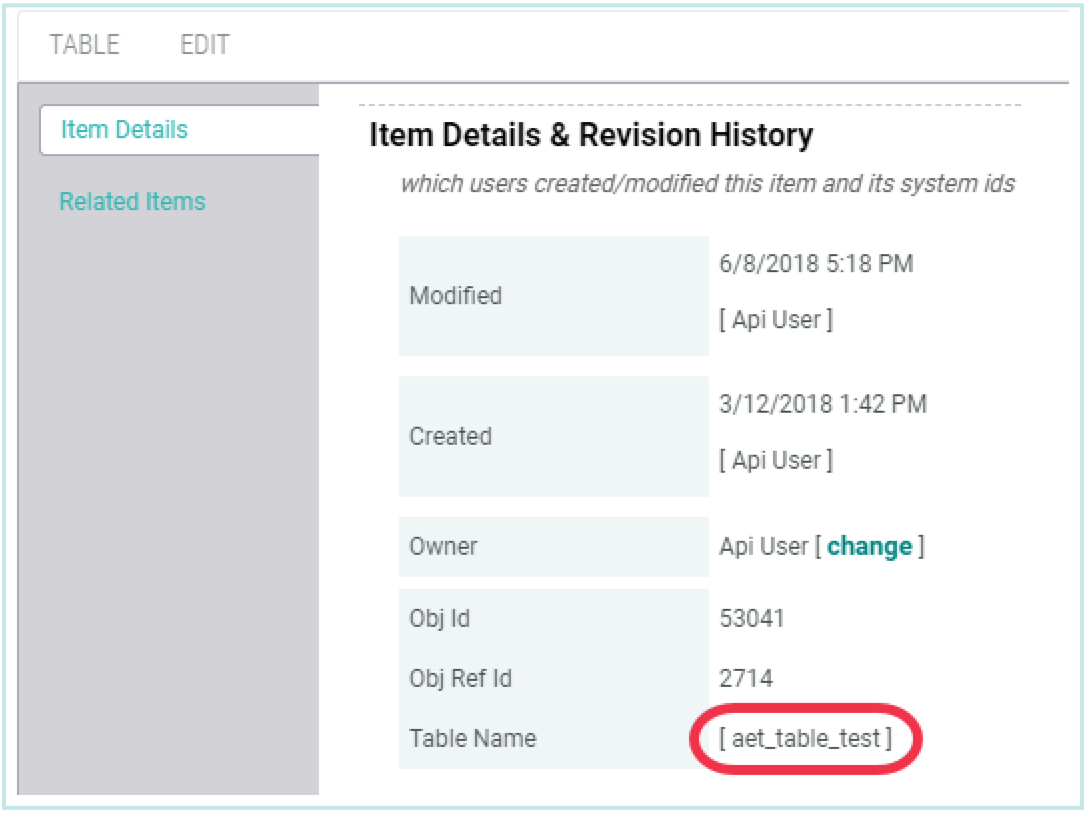
You can look up a table's system name within the application, or by using the TABLE endpoint.

To look up the system name within the application:

1. From the System Tray, select *Data Management* > *Structures* > *Tables*. The system displays a list of all the Tables in your account.

2. Select the desired Table. The Table Details screen is displayed.

3. In the Tool Ribbon, click the "Table" tab. The "Item Details" screen is displayed. The system name for this table is displayed.



Item Details & Revision History

which users created/modified this item and its system ids

Modified	6/8/2018 5:18 PM [Api User]
Created	3/12/2018 1:42 PM [Api User]
Owner	Api User [**change**]
Obj Id	53041
Obj Ref Id	2714
Table Name	[aet_table_test]

You can also retrieve the table's system name using the TABLE API endpoint.

To retrieve the table's system name:

1. Submit a request to the TABLE API endpoint. The simplest method is to use the version of the TABLE endpoint that allows you to retrieve information for all tables. Please note that depending on the number of tables in your account, you may need to increase the response message size (by default, the system returns the first 20

tables). Within the URL, add the **count** query type parameter, and enter the number of tables you want to return. For example:

```
https://api.eccmp.com/services2/api/Table?count=50
```

2. Within the API response message, the system returns the table details. The table's system name is provided in the **tableName** parameter.

Sample Response:

```
{
  "viewId": 2714,
  "viewName": "AET table test",
  "entityId": 816,
  "tableName": "aet_table_test"
}
```

Column Names

For the field names in the **SEARCH RECORDS** payload, you must use the system "Column Name," and not the viewer-friendly "Display Name." The Column Names for fields can be found on the Tables screen within the Messaging application, or by using the **TABLE** endpoint.

To look up the Column Name within the Messaging application:

1. From the System Tray, select *Data Management* > *Structures* > *Tables*. The system displays a list of all the tables in your account.
2. Select the desired table. The Table Details screen is displayed.
3. Within the list of fields in this table, the Column Name is displayed on the far-right of the screen.

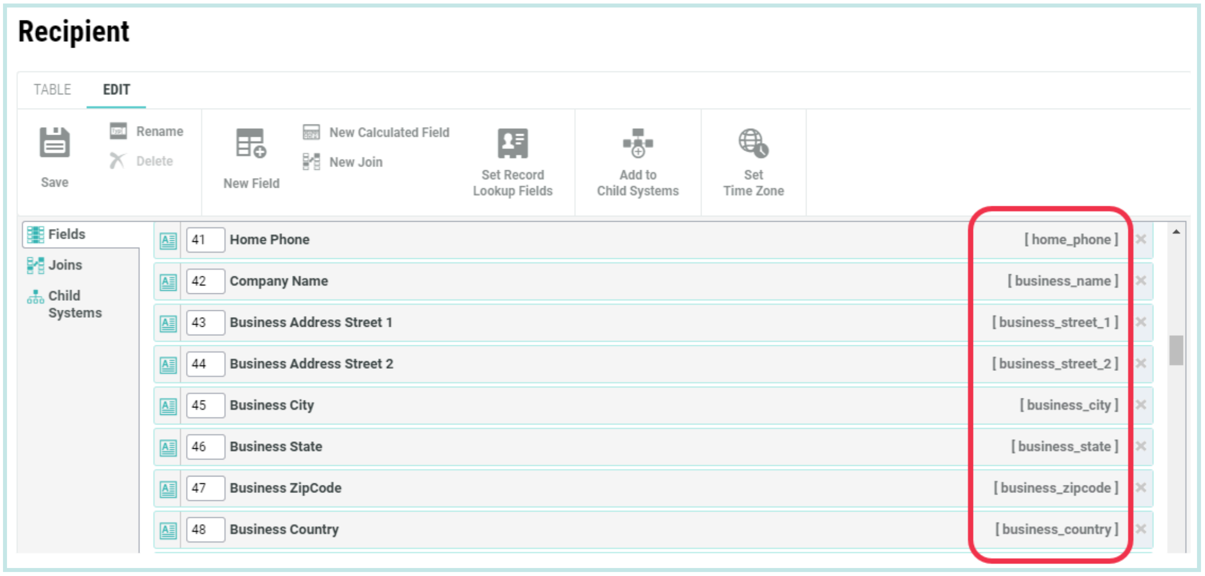

Recipient

TABLE EDIT

41	Home Phone	[home_phone]
42	Company Name	[business_name]
43	Business Address Street 1	[business_street_1]
44	Business Address Street 2	[business_street_2]
45	Business City	[business_city]
46	Business State	[business_state]
47	Business ZipCode	[business_zipcode]
48	Business Country	[business_country]

To retrieve the Column Name for a field using the TABLE endpoint:

1. Submit a GET request to the TABLE API endpoint. The simplest method is to use the version of the TABLE endpoint that allows you to retrieve table information based on the table's name. For example:

```
https://api.eccmp.com/services2/api/Table?tableName=recipient
```

2. Within the API response message, the system lists every field in this table. As part of that field definition, the response includes the Column Name (referred to as the **columnName**).

Sample Response:

```
{
 "viewId": 1002,
 "entityId": 100,
 "displayName": "First Name",
 "propId": 1030,
 "columnName": "name_first"
}
```