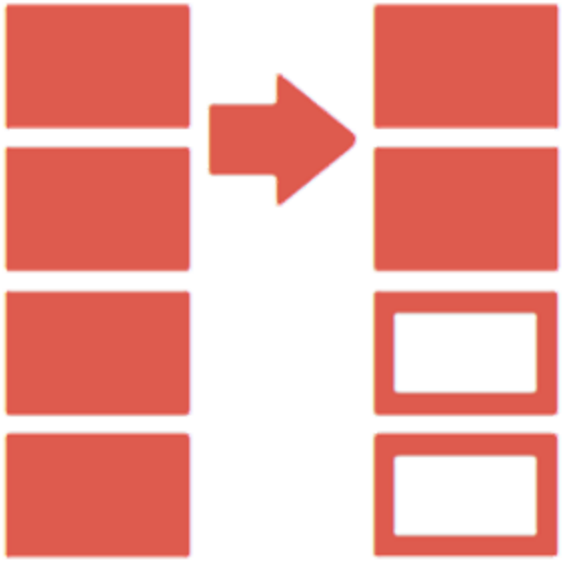
MARIGOLD™

API Technical Guide: Data Map

Cheetah Messaging

Table of Contents

1 Introduction



Purpose

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

Overview

The DATA MAP endpoint is used to create, view, edit, and delete Data Maps. Data Maps are reusable assets that control how inbound data is written to your Messaging database. The Data Map defines the sequence of the columns in the import, identifies the correct table and field into which each column should be written, and specifies any optional special processing or formatting to perform on the data.

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/DataMap
- **Europe:** https://api.ccmp.eu/services2/api/DataMap
- **Japan**: https://api.marketingsuite.jp/services2/api/DataMap

Methods

The DATA MAP endpoint supports the following HTTP methods:

- **POST**: Create a new Data Map.
- **GET**: Retrieve information about a specified Data Map.
- **PUT**: Submit modifications to an existing Data Map.
- **DELETE**: Delete a specified Data Map.

Authentication

Access to the DATA MAP 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 Create a Data Map

Overview

This section describes how to create a new Data Map via a POST request to the DATA MAP endpoint. This section is split into two sub-sections: "Common Parameters" and "Advanced Parameters."

The "Common Parameters" sub-section describes the basic, required parameters needed to create a simple Data Map suitable for most purposes.

The "Advanced Parameters" sub-section describes the additional, optional parameters needed for utilizing the platform's more complex import options.

Common Parameters

The options and parameters described in this section explain how to create a typical Data Map containing one or more fields on a single table.

entity_id

This integer parameter is required.

The **entity_id** parameter represents the Entity ID of the Data Map's source table. The source table is the main database table into which data will be loaded through this Data Map.

> **Note**
>
> You can also optionally load data into tables joined to the source table; see Data Maps with Joined Tables for more details.

Example:

```
"entity_id": 100
```

cust_id

This integer parameter is required.

The **cust_id** parameter represents the Customer ID of your Messaging account. The Customer ID is a unique, system-generated identifier for every Messaging client account. This value isn't displayed anywhere within the Messaging application, so you must retrieve it by means of an API request, or speak to your Client Services Representative, who can proivde you with this value.

Example:

```
"cust_id": 394
```

PropMapImportSetting

This object contains data handling instructions that tell the platform whether to use the data in the import to only create new records, to only update existing records, or to both create and update records.

Example:

```
"propMapImportSetting":
  {
  "suppress_new_flag": 0,
  "suppress_old_flag": 1
  }
```

The parameters in this object are described below in more detail.

suppress_new_flag / suppress_old_flag

These two integer parameters are optional.

The combination of these two flags is used to tell the system whether to use the data in the import to create new records only, update existing records only, or both create and update records.

Flag	Value	Flag	Value	Result
suppress_new_flag	0	suppress_old_flag	0	Create or Update Records
suppress_new_flag	1	suppress_old_flag	0	Update Only

suppress_new_flag	0	suppress_old_flag	1	Create Only

If not provided, the system will default to "0" for both parameters.

PropMapFlat

This object contains information about the structure of the import, such as the delimiter character, and whether the import contains a header row.

Example for a character-delimited import:

```
"propMapFlat":
  {
  "first_row": 1,
  "separator": "\\t",
  "text_qualifier": "'"
  }
```

Example for a fixed width import:

```
"propMapFlat":
  {
  "first_row": 1,
  "separator": "fixed",
  "fixed_width_flag": 1,
  "text_qualifier": "\""
  }
```

The parameters in this object are described below in more detail.

first_row

This integer parameter is optional. If not provided, the system will default to "1."

This parameter is used to indicate the first row in the import that contains data. If the import has a header row, you can use this parameter to "skip" the header row, and start looking for data in the second row.

Set this parameter to "2" if the import contains a header row; set the parameter to "1" if the import doesn't contain a header row.

separator

This string parameter is required.

This parameter is used to indicate whether the import uses fixed widths to determine the field lengths, or if the import fields are delimited (and if so, what character is used as the delimiter).

The possible values for this parameter are:

- "\\t" -- tab-delimited
- "," -- comma-delimited
- ";" -- semi colon-delimited
- "|" -- pipe-delimited
- "^" -- caret-delimited
- "fixed" -- fixed width

multivalue_separator

This parameter is not currently supported.

fixed_width_flag

This integer parameter is optional. If not provided, the system will default to "0" (meaning, the import is delimited, not fixed width).

This parameter is used to indicate that the import uses fixed widths to determine fields lengths, rather than delimiter characters. If your import is fixed width, set this parameter to "1."

text_qualifier

This integer parameter is optional. If not provided, the system will default to blank.

This parameter is used to indicate whether the import uses text qualifiers around text strings, and if so, what character is used to qualify the text strings.

The possible values for this parameter are:

- " ' " -- Single quote
- " \" " -- Double quote

PropMapProps

This object tells the platform where to store the inbound data in the import by "mapping" each desired column in the import to a field in a database table.

When creating a Data Map, you must repeat this object for each column in the import that you want to write to the database.

Example for a character-delimited import containing three columns:

```
"propMapProps":
  [
    {
    "prop_id": 14457,
    "seq": 1,
    "label": "email"
    },
    {
    "prop_id": 14459,
    "seq": 2,
    "label": "first_name"
    },
    {
    "prop_id": 14460,
    "seq": 3,
    "label": "last_name"
    }
  ]
```

Example for a fixed width import containing three columns:

```
"propMapProps":
  [
    {
    "prop_id": 14457,
    "seq": 1,
    "label": "email",
    "field_start_index": 1,
    "field_length": 30
    },
    {
    "prop_id": 14459,
    "seq": 2,
    "label": "firstname",
    "field_start_index": 32,
    "field_length": 50
    },
    {
    "prop_id": 14460,
    "seq": 3,
    "label": "lastname",
    "field_start_index": 83,
    "field_length": 50
```

```
        }
    ]
```

The parameters in this object are described below in more detail.

prop_id

This integer parameter is required.

The **prop_id** parameter represents the Field ID of a single database field, and tells the system where you want to store the data in a given import column.

seq

This integer parameter is optional. If you don't provide this parameter, the system will default the value to "-2147483648."

The value in this parameter is used to control the sequence of the columns in the import. In most Data Maps, you will simply increment this parameter by 1 for each subsequent column in your import.

If you need to ignore a column in the import, simply skip over that column number in the sequence. For example, if you want to ignore column #4 in your import file, the numbering sequence in your Data Map would go from "3" to "5," and leave out "4." The key thing to keep in mind is that the sequence you're defining with the **seq** parameter represents the column order within the import, and the **prop_id** is the location in the database where that data is to be written.

label

This string parameter is optional. If you don't provide this parameter, the system will default to using the Column Name for this field.

This parameter contains the Form Field Name, which is intended for use with Web Forms. When a consumer fills out and submits a Web Form, the Form sends that data to Messaging by means of an API request message. If you want to use a different name for a field within the API message, you must provide that name within this parameter, so the system knows where to store that data within the database.

field_start_index

This integer parameter is optional.

This parameter is used in a Fixed Width import to indicate the field's starting position within the import.

field_length

This integer parameter is optional.

This parameter is used in a Fixed Width import to indicate the field's length.

propMapPropRules

This object is used for including Calculated Fields within your Data Map. Calculated Fields contain values that are derived using logic. If you include a Calculated Field within your Data Map, when the system processes an import with this Data Map, it will derive and populate the specified Calculated Field for each record in the import. A Data Map can optionally contain multiple Calculated Fields.

> **Note**
>
> The **PropMapPropRules** object is required within the request message, even if you don't want to derive any Calculated Fields for this Data Map; in this case, simply include an empty **PropMapPropRules** object.

Example:

```
"propMapPropRules":
[
  {
  "prop_id": 15585,
  "seq": 1
  }
],
```

The parameters in this object are described below in more detail.

prop_id

This integer parameter is optional.

The **prop_id** parameter represents the Field ID of a Calculated Field.

seq

This integer parameter is optional.

This parameter is used to control the sequence of the Calculated Fields within the Data Map. Typically, you will simply increment this counter by one for each additional Calculated Field in your Data Map.

Obj

This object contains the name of the Data Map.

Example:

```
"obj":
  {
  "display_name": "Sample Data Map"
  }
```

The parameters in this object are described below in more detail.

display_name

This string parameter is required.

This parameter contains the name of the Data Map. This name must be unique within the client account.

Advanced Parameters

The options and parameters described in this section explain how to create a complex Data Map using the platform's advanced data integration features. Many of the objects and parameters needed for a complex Data Map are described in detail in the previous section, **Common Parameters**. This section focuses only on the advanced parameters not already described in the previous section.

PropMapProps

The **Common Parameters** section above describes the basic options for this object, which is used to define each column in the import. In addition to the parameters described above, this object supports a few additional, optional parameters that provide special formatting options for certain data types.

For example:

```
"propMapProps":
  [
    {
    "prop_id": 13049,
    "seq": 1,
    "label": "Price",
    "culture_name": "en-US"
    },
    "prop_id": 11360,
    "seq": 2,
    "label": "mobile_phone",
    "country_id": 140
    }
  ]
```

The advanced parameters in this object are described below in more detail.

culture_name

This string parameter is optional, and can be used for columns with a data type of "Money / Decimal," "Integer," "Big Integer," or "Preference."

This parameter lets you select a locale that controls the number formatting you want to use for this field. The valid values for this parameter are:

- "de-DE" -- German (Germany)
- "en-GB" -- English (United Kingdom)
- "en-US" -- English (United States)
- "es-ES" -- Spanish (Spain)
- "fr-FR" -- French (France)
- "ja-JP" -- Japanese (Japan)
- "pt-BR" -- Portuguese (Brazil)

country_id

This integer parameter is optional, and can be used for columns with a data type of "Phone."

This parameter is used to prepend a standard international calling code to the beginning of a phone number value. If using this parameter, you must provide the value for the desired country. The possible values for this parameter, and the corresponding calling code for each value, are listed in Additional Resources.

propMapPropImportSetting

This object is used to indicate whether a column in the import should use any of the platform's optional "Advanced Options" when importing data.

> **Note**
>
> The DATA MAP endpoint doesn't support the "Don't Re-Opt-In" Advanced Option. If you want to use this option, you'll have to set it within the Messaging application.

Example:

```
"propMapProps":
  [
    {
    "prop_id": 1150,
    "seq": 1,
    "label": "EmailAddr",
    "propMapPropImportSetting":
      {
        "blast_flag": 0,
        "keep_old_value_flag": 0,
        "treat_null_as_value_flag": 0,
        "case_sensitive_flag": 0,
        "multivalue_append_flag": 0,
        "skip_email_ban_flag": 1,
        "soft_match_seq": 1
      }
    }
  ]
```

The parameters in this object are described below in more detail.

blast_flag

This integer parameter is optional. If you don't provide this parameter, the system will default the value to "0."

If set to "1," the system will delete all existing data in the database for this column, and add any new data from this import.

keep_old_value_flag

This integer parameter is optional. If you don't provide this parameter, the system will default the value to "0."

If set to "1," the system will keep any data that already exists in this column, and will not overwrite this existing data with the data in the import.

treat_null_as_value_flag

This integer parameter is optional. If you don't provide this parameter, the system will default the value to "0."

If set to "1," the system will overwrite the data in the database as NULL if the column in the import is empty.

case_sensitive_flag

This integer parameter is optional. If you don't provide this parameter, the system will default the value to "0."

If set to "1," the system will use case-sensitive comparisons between any existing data in the database and the data being imported.

multivalue_append_flag

This integer parameter is optional. If you don't provide this parameter, the system will default the value to "0."

If set to "1," the system will append values from the import onto any existing values in the database, rather than overwriting them.

skip_email_ban_flag

This integer parameter is optional, and is used only for columns with a data type of "Email. " If you don't provide this parameter, the system will default the value to "0."

If set to "1," the system will not process this Email field through any Email Ban lists.

soft_match_seq

This integer parameter is optional, and is used only for columns with a data type of "Email," "Phone," "Twitter," "Facebook ID," or "Push Registration ID." If you don't provide this parameter, the system will default the value to "0."

This parameter is used to identify a field (or fields) on which to perform a Soft Match. If the import file doesn't contain the Unique Identifier field for the source table, then you must use the Soft Match feature in order to successfully load the data.

If the soft-matched value in the import file matches an existing value in the database, then the platform will make the update to that record in the database. If the soft-matched value does not already exist in the database, the platform creates a "temporary" record. This temporary record can later be merged with a full record that contains the Unique Identifier.

If your import contains multiple Soft Match fields, this parameter should increment sequentially in the order in which you want to perform the Soft Match ("1" for the first field, "2" for the second field, and so on).

propMapPropProcess

This object is used to define any special processing that you want to perform on a column during import.

Example:

```
"propMapProps":
  [
    {
    "prop_id": 1150,
    "seq": 1,
    "label": "EmailAddr",
    "propMapPropProcess":
      {
      "procedure_id": 100,
      "procedure_param": "1500"
      }
    }
  ]
```

The parameters in this object are described below in more detail.

procedure_id

This integer parameter is optional. If you don't provide this parameter, the system will default the value to "0."

This parameter identifies the special procedure that you want to perform on this column. The valid values for this parameter are:

- **"100" -- Set Default Value:** Allows you to set the value in this field as the default value.

- **"110" -- Resolve camp_id from msg_id:** Given a "msg_id" on the import file, the system will attempt to identify the associated Campaign ID (typically used for custom responses).
- **"120" -- Resolve pk_id fro msg_id:** Given a "msg_id" on the import file, the system will attempt to identify the associated Primary Key ID (typically used for custom responses).
- **"130" -- Set to mixed case:** Transforms a text string into mixed case (capitalize the first letter of each word).
- **"140" -- Get name prefix:** Identifies and stores only the Name Prefix value from a "Full Name" field.
- **"150" -- Get first name:** Identifies and stores only the First Name value from a "Full Name" field.
- **"160" -- Get last name:** Identifies and stores only the Last Name value from a "Full Name" field.
- **"170" -- Get name suffix:** Identifies and stores only the Name Suffix value from a "Full Name" field.
- **"200" -- Set to mixed case (specific acronyms excluded):** Transforms a text string into mixed case (capitalize the first letter of each word), except for a global list of acronyms; if the text string is one of these specific acronyms, the platform won't change the value to mixed case. Please see Additional Resources for the list of specific excluded acronyms.
- **"210" -- Make positive:** Changes all numeric values to positive values.
- **"220" -- Make negative:** Changes all numeric values to negative values.
- **"230" -- Set to lower case:** Transforms a text string into all lower case.
- **"240" -- Set to upper case:** Transforms a text string into all upper case.

procedure_param

This string parameter is optional.

If you selected a **procedure_id** of "100" (Set Default Value), this parameter allows you to provide the desired default value.

Data Maps with Joined Tables

The Messaging platform allows you to define Data Maps that write inbound data, not just to the Data Map's source table, but also to other tables that are joined to the source table.

A common example of a join is connecting a Customer table (which contains contact information about an individual) with an Order table (which contains purchase order information). This type of join is called a "one-to-many" join because one customer can make multiple purchases. You could further join the Order table to an Order Item table that contains the details of the items contained in each purchase.

Within Messaging, joins must be set up in the many-to-one direction. In the above example, you would select the Order table (the "many"), and create the join to the Customer table (the "one"), and not the other way around. Also, joins should be created only in one direction, not both.

When using the DATA MAP endpoint, the process of creating a Data Map to a joined table involves two separate steps, with two separate POST requests.

1. Create the Data Map for the "one" table. In the response message, make a note of the **prop_map_id** (that is, the system-generated Object Reference ID for this new Data Map).
2. Create the Data Map for the "many" table. In this request message, you must reference the Field ID for the join, and the Object Reference ID from the "one" table Data Map.

Essentially, you are creating two Data Maps, and "nesting" the first Data Map inside of the second Data Map.

Example:

```
"propMapProps":
  [
    {
    "prop_id": 18000,
    "fk_prop_map_id": 4584
    }
  ]
```

The parameters in this object are described below in more detail.

prop_id

This integer parameter is optional.

In the context of a Data Map with a join, the **prop_id** parameter represents the Field ID of the join between the two tables.

fk_prop_map_id

This integer parameter is optional.

The **fk_prop_map_id** parameter represents the Object Reference ID of the "nested" Data Map (that is, the Data Map for the joined "one" table).

3 Edit a Data Map

Overview

This section describes how to work with existing Data Maps via a GET, PUT, or DELETE request to the DATA MAP endpoint.

Retrieve a Data Map

The GET method is used to retrieve all of the information about a specified Data Map.

When submitting a GET request to the DATA MAP endpoint, the request message must include the Data Map's **Object Reference ID** as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/DataMap?id=3456
```

Delete a Data Map

The DELETE method is used to delete a specified Data Map.

When submitting a DELETE request to the DATA MAP endpoint, the request message must include the Data Map's **Object Reference ID** as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/DataMap?id=3456
```

Edit a Data Map

The PUT method allows you to submit modifications to an existing Data Map. Using this method, you can change the Data Map name, modify the import information, modify the data processing options, modify the Advanced Options, add new fields, and remove

existing fields. The parameters for the PUT method are the same as described in the **Create a Data Map** section.

To remove a field from an existing Data Map, simply omit it from the request message; any existing fields that aren't referenced in the request message will be removed from the Data Map.

When submitting a PUT request to the DATA MAP endpoint, the request message must include the Data Map's **Object Reference ID** as a query type parameter within the URL.

For example:

```
https://api.eccmp.com/services2/api/DataMap?id=3456
```

4 Response

This section describes the possible response messages sent back from the DATA MAP endpoint.

Success

A successful response to a POST message will generate a response code of "200," followed by the details of the new Data Map contained within the body of the response message.

A successful response to a GET message will generate a response code of "200," followed by the details of the specified Data Map contained within the body of the response message.

A successful response to a PUT message will generate a response code of "200," followed by the details of the modified Data Map contained within the body of the response message.

A successful response to a DELETE message will generate a response code of "204;" the body of the response message will be empty.

Errors

If Messaging encounters a problem with a DATA MAP 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
500	"File Delimiter is required"	Required parameter **separator** is missing

Response Code	Error message	Description
500	"An Obj with this name already exists"	Duplicate Data Map name; a Data Map must have a unique name within your account.
500	"Invalid procedure_id"	The **procedure_id** value is not valid (see procedure_id for a list of the valid values for this parameter).

5 Sample Messages

This section contains sample request messages for the DATA MAP endpoint.

Request Message #1

This sample POST request message creates a tab-delimited Data Map consisting of two fields.

JSON Payload

```
{
  "entity_id": 100,
  "cust_id": 394,
  "propMapImportSetting":
    {
    "suppress_new_flag": 1,
    "suppress_old_flag": 0
    },
  "propMapFlat":
   {
   "first_row": 2,
   "separator": "\\t"
   },
  "propMapProps":
  [
    {
    "prop_id": 1150,
    "seq": 1,
    "label": "EmailAddr"
    },
    {
    "prop_id": 1100,
    "seq": 2,
    "label": "FName"
    }
  ],
  "propMapPropRules": [],
  "obj": {
    "display_name": "Sample Data Map 1"
  }
}
```

Request Message #2

This sample POST request message includes a few more options than the one described above.

In this example, the user has included a Money / Decimal field, and defined the locale for this field. In addition, the user has set an Advanced Option, to not process the email address field against the Email Ban lists. Lastly, the user wants to derive and populate a calculated field, which is controlled by the **PropMapPropRules** object.

JSON Payload


```
{
  "entity_id": 100,
  "cust_id": 394,
  "propMapImportSetting":
    {
    "suppress_new_flag": 1,
    "suppress_old_flag": 0
    },
  "propMapFlat":
   {
   "first_row": 2,
   "separator": "\\t"
   },
  "propMapProps":
  [
    {
    "prop_id": 1150,
    "seq": 1,
    "label": "EmailAddr",
    "propMapPropImportSetting":
      {
      "blast_flag": 0,
      "keep_old_value_flag": 0,
      "treat_null_as_value_flag": 0,
      "case_sensitive_flag": 0,
      "multivalue_append_flag": 0,
      "skip_email_ban_flag": 1
      }
    },
    {
    "prop_id": 1100,
    "seq": 2,
    "label": "FName"
    },
    {
    "prop_id": 13049,
    "seq": 3,
    "label": "Price",
```

```
    "culture_name": "en-US"
    }
  ],
  "propMapPropRules":
  [
    {
     "prop_id": 15585,
     "seq": 1
    }
  ],
  "obj": {
    "display_name": "Sample Data Map 2"
  }
}
```

Request Message #3

These sample POST request messages result in a Data Map that includes a join to another table.

The first message creates the Data Map for the "one" table.

JSON Payload

```
{
  "entity_id": 474,
  "cust_id": 394,
  "propMapImportSetting":
    {
    "suppress_new_flag": 0,
    "suppress_old_flag": 0
    },
  "propMapFlat":
    {
    "first_row": 2,
    "separator": "\\t"
    },
  "propMapProps":
    [
      {
      "prop_id": 14457,
      "seq": 1,
      "label": "email"
      },
      {
      "prop_id": 14459,
      "seq": 2,
      "label": "FNAME"
      },
      {
      "prop_id": 14460,
```

```
      "seq": 3,
      "label": "LNAME"
      }
    ],
  "propMapPropRules": [],
  "obj":
    {
      "display_name": "ONE Data Map"
      }
  }
```

Then, the second message creates the Data Map for the "many" table. This message references the Object Reference ID for the Data Map created above.

JSON Payload

```
{
  "entity_id": 754,
  "cust_id": 394,
  "propMapImportSetting":
    {
    "suppress_new_flag": 0,
    "suppress_old_flag": 0
    },
  "propMapFlat":
    {
    "first_row": 2,
    "separator": "\\t"
    },
  "propMapProps":
    [
      {
      "prop_id": 18000,
      "fk_prop_map_id": 4584
      },
      {
      "prop_id": 17997,
      "seq": 1,
      "label": "ORDERID"
      },
      {
      "prop_id": 18024,
      "seq": 2,
      "label": "EMAIL"
      },
      {
      "prop_id": 17999,
      "seq": 3,
      "label": "TOTAL"
      }
    ],
  "propMapPropRules": [],
  "obj":
    {
```

```
        "display_name": "MANY Data Map"
        }
    }
```

6 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.

Entity ID

The Entity ID is a unique, system-generated identifier for every table in your database. This value is not displayed within the application user interface anywhere, so to get the Entity ID for a table, you must retrieve it by means of the TABLE API endpoint.

To retrieve the Entity ID for a table:

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 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 the field details, the response message provides the Entity ID for this table.

Sample Response:

```
{
  "viewId": 1002,
  "entityId": 100,
  "displayName": "create_date",
  "propId": 1030,
  "columnName": "create_date"
}
```

For more details on the TABLE endpoint, please see the Messaging Online Help system or the *Messaging -- Table API Technical Guide*.

Field ID

The Field ID (or "Property ID") is a unique, system-generated identifier for every field in a table. This value is not displayed within the application user interface anywhere, so to get the Field ID for a field, you must retrieve it by means of the TABLE API endpoint.

To retrieve the Field ID for a field:

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 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 Field ID (referred to as the **propId**).

Sample Response:

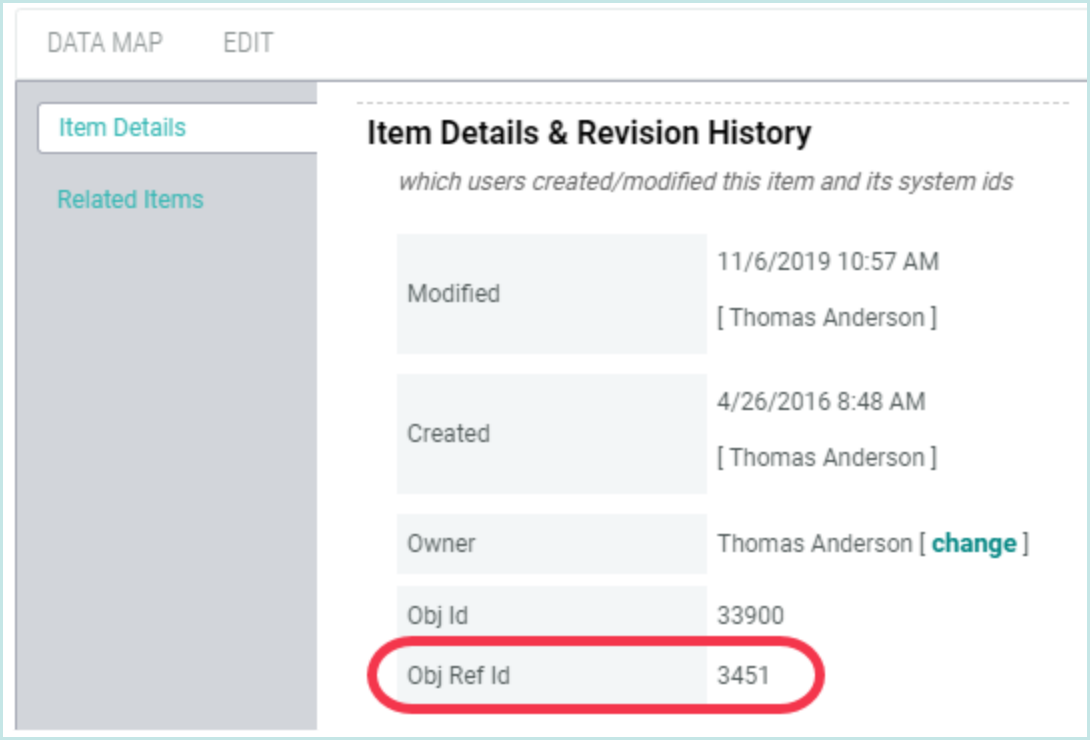
```
{
 "viewId": 1002,
 "entityId": 100,
 "displayName": "create_date",
 "propId": 1030,
 "columnName": "create_date"
}
```

Object Reference ID

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

For Data Maps, the value for this identifier can be found within the Messaging application:

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



Optionally, you can use the SEARCH endpoint, and search for the desired Data Map:

1. Submit a GET request to the SEARCH API endpoint. The simplest method is to use the versions of the SEARCH endpoint that allow you to retrieve information based on either the asset's name or its type. For example, to retrieve information about all of your Data Maps:

```
https://api.eccmp.com/services2/api/Object?type=PropertyMap
```

2. The response message provides a list of all the assets in your system that match the search criteria. Find the desired asset in the response message.

3. As part of the API response message, the system provides the Object Reference ID, which is referred to as the "**ref_id**." For example:

```
{
  "obj_id": 44737,
  "display_name": "Import Data Map",
  "type_id": "PropertyMap",
  "ref_id": 40329,
  "parent_obj_id": 43269,
  "eligibility_status_id": "READY"
}
```

7 Appendix B -- Additional Resources

This Appendix provides additional details on several parameters used in the DATA MAP endpoint.

Country Codes

The **country_id** parameter allows you to prepend the standard international dialing code to the beginning of a "Phone" field. If using this parameter, you must provide the value for the desired country, as listed below:

country_id	Code	Country	Dialing Code	Leading Zero
10	AC	Ascension Island	247	
20	AD	Andorra	376	
30	AE	United Arab Emirates	971	
40	AF	Afghanistan	93	
50	AG	Antigua and Barbuda	1	
60	AI	Anguilla	1	
70	AL	Albania	355	
80	AM	Armenia	374	
90	AN	Netherlands Antilles	599	
100	AO	Angola	244	
110	AR	Argentina	54	
120	AS	American Samoa	1	
130	AT	Austria	43	
140	AU	Australia	61	

country_id	Code	Country	Dialing Code	Leading Zero
150	AW	Aruba	297	
160	AX	Aland Islands	358	
170	AZ	Azerbaijan	994	
180	BA	Bosnia and Herzegovina	387	
190	BB	Barbados	1	
200	BD	Bangladesh	880	
210	BE	Belgium	32	
220	BF	Burkina Faso	226	
230	BG	Bulgaria	359	
240	BH	Bahrain	973	
250	BI	Burundi	257	
260	BJ	Benin	229	
270	BL	Saint Barthelemy	590	
280	BM	Bermuda	1	
290	BN	Brunei Darussalam	673	
300	BO	Bolivia	591	
310	BQ	Bonaire, Sint Eustatius And Saba	599	
320	BR	Brazil	55	
330	BS	Bahamas	1242	
340	BT	Bhutan	975	
350	BW	Botswana	267	
360	BY	Belarus	375	
370	BZ	Belize	501	Yes
380	CA	Canada	1	
390	CC	Cocos Islands	61	
400	CD	Congo, Democratic Republic of	243	
410	CF	Central African Republic	236	
420	CG	Congo	242	Yes
430	CH	Switzerland	41	

country_id	Code	Country	Dialing Code	Leading Zero
440	CI	Cote d Ivoire	225	Yes
450	CK	Cook Islands	682	
460	CL	Chile	56	
470	CM	Cameroon	237	
480	CN	China	86	
490	CO	Colombia	57	
500	CR	Costa Rica	506	
510	CU	Cuba	53	
520	CV	Cape Verde	238	
530	CW	Curacao	599	
540	CX	Christmas Island	61	
550	CY	Cyprus	357	
560	CZ	Czech Republic	420	
570	DE	Germany	49	
580	DJ	Djibouti	253	
590	DK	Denmark	45	
600	DM	Dominica, Commonwealth of	1	
610	DO	Dominican Republic	1	
620	DZ	Algeria	213	
630	EC	Ecuador	593	
640	EE	Estonia	372	
650	EG	Egypt	20	
660	ER	Eritrea	291	
670	ES	Spain	34	
680	ET	Ethiopia	251	
690	FI	Finland	358	
700	FJ	Fiji	679	Yes
710	FK	Falkland Islands	500	
720	FM	Micronesia, Federated States	691	

country_id	Code	Country	Dialing Code	Leading Zero
730	FO	Faroe Islands	298	
740	FR	France	33	
750	GA	Gabon	241	
760	GB	United Kingdom	44	
770	GD	Grenada	1	
780	GE	Georgia	995	
790	GF	French Guiana	594	
800	GG	Guernsey	44	
810	GH	Ghana	233	
820	GI	Gibraltar	350	
830	GL	Greenland	299	
840	GM	Gambia	220	
850	GN	Guinea	224	
860	GP	Guadeloupe	590	
870	GQ	Equatorial Guinea	240	
880	GR	Greece	30	
890	GT	Guatemala	502	
900	GU	Guam	1	
910	GW	Guinea-Bissau	245	
920	GY	Guyana	592	
930	HK	Hong Kong	852	
940	HN	Honduras	504	
950	HR	Croatia	385	
960	HT	Haiti	509	
970	HU	Hungary	36	
980	ID	Indonesia	62	
990	IE	Ireland	353	
1000	IL	Israel	972	
1010	IM	Isle of Man	44	

country_id	Code	Country	Dialing Code	Leading Zero
1020	IN	India	91	
1030	IO	British Indian Ocean Territory	246	
1040	IQ	Iraq	964	
1050	IR	Iran	98	
1060	IS	Iceland	354	
1070	IT	Italy	39	Yes
1080	JE	Jersey	44	
1090	JM	Jamaica	1	
1100	JO	Jordan	962	
1110	JP	Japan	81	
1120	KE	Kenya	254	
1130	KG	Kyrgyzstan	996	
1140	KH	Cambodia	855	
1150	KI	Kiribati	686	
1160	KM	Comoros	269	
1170	KN	Saint Kitts and Nevis	1	
1180	KP	Korea, Democratic People's Republic of	850	
1190	KR	Korea, Republic of	82	
1200	KW	Kuwait	965	
1210	KY	Cayman Islands	1	
1220	KZ	Kazakhstan	7	
1230	LA	Laos	856	
1240	LB	Lebanon	961	
1250	LC	Saint Lucia	1	
1260	LI	Liechtenstein	423	
1270	LK	Sri Lanka	94	
1280	LR	Liberia	231	
1290	LS	Lesotho	266	
1300	LT	Lithuania	370	

country_id	Code	Country	Dialing Code	Leading Zero
1310	LU	Luxembourg	352	
1320	LV	Latvia	371	
1330	LY	Libya	218	
1340	MA	Morocco	212	
1350	MC	Monaco	377	
1360	MD	Moldova, Republic of	373	
1370	ME	Montenegro	382	
1380	MF	Saint Martin	590	
1390	MG	Madagascar	261	
1400	MH	Marshall Islands	692	
1410	MK	Macedonia	389	
1420	ML	Mali	223	
1430	MM	Myanmar	95	
1440	MN	Mongolia	976	
1450	MO	Macau	853	
1460	MP	Northern Mariana Islands	1	
1470	MQ	Martinique	596	
1480	MR	Mauritania	222	
1490	MS	Montserrat	1	
1500	MT	Malta	356	
1510	MU	Mauritius	230	
1520	MV	Maldives	960	
1530	MW	Malawi	265	
1540	MX	Mexico	52	
1550	MY	Malaysia	60	
1560	MZ	Mozambique	258	
1570	NA	Namibia	264	
1580	NC	New Caledonia	687	
1590	NE	Niger	227	Yes

country_id	Code	Country	Dialing Code	Leading Zero
1600	NF	Norfolk Island	672	
1610	NG	Nigeria	234	
1620	NI	Nicaragua	505	
1630	NL	Netherlands	31	
1640	NO	Norway	47	Yes
1650	NP	Nepal	977	
1660	NR	Nauru	674	
1670	NU	Niue	683	
1680	NZ	New Zealand	64	
1690	OM	Oman	968	
1700	PA	Panama	507	
1710	PE	Peru	51	
1720	PF	French Polynesia	689	
1730	PG	Papua New Guinea	675	
1740	PH	Philippines	63	
1750	PK	Pakistan	92	
1760	PL	Poland	48	
1770	PM	Saint Pierre and Miquelon	508	
1780	PR	Puerto Rico	1	
1790	PS	Palestine	970	
1800	PT	Portugal	351	
1810	PW	Palau	680	
1820	PY	Paraguay	595	
1830	QA	Qatar	974	
1840	RE	Reunion	262	
1850	RO	Romania	40	
1860	RS	Serbia	381	
1870	RU	Russia	7	
1880	RW	Rwanda	250	Yes

country_id	Code	Country	Dialing Code	Leading Zero
1890	SA	Saudi Arabia	966	
1900	SB	Solomon Islands	677	
1910	SC	Seychelles	248	
1920	SD	Sudan	249	
1930	SE	Sweden	46	
1940	SG	Singapore	65	
1950	SH	Saint Helena, Ascension And Tristan Da Cunha	290	
1960	SI	Slovenia	386	
1970	SJ	Svalbard And Valbard And Jan Mayen	47	Yes
1980	SK	Slovakia	421	
1990	SL	Sierra Leone	232	
2000	SM	San Marino	378	Yes
2010	SN	Senegal	221	
2020	SO	Somalia	252	
2030	SR	Suriname	597	
2040	SS	South Sudan	211	
2050	ST	Sao Tome and Principe	239	
2060	SV	El Salvador	503	
2070	SX	Sint Maarten	1	
2080	SY	Syria	963	
2090	SZ	Swaziland	268	Yes
2100	TC	Turks and Caicos Islands	1	
2110	TD	Chad	235	
2120	TG	Togo	228	
2130	TH	Thailand	66	
2140	TJ	Tajikistan	992	
2150	TK	Tokelau	690	
2160	TL	Timor-Leste	670	

country_id	Code	Country	Dialing Code	Leading Zero
2170	TM	Turkmenistan	993	
2180	TN	Tunisia	216	
2190	TO	Tonga Islands	676	Yes
2200	TR	Turkey	90	
2210	TT	Trinidad and Tobago	1	
2220	TV	Tuvalu	688	
2230	TW	Taiwan	886	
2240	TZ	Tanzania	255	
2250	UA	Ukraine	380	
2260	UG	Uganda	256	
2270	US	United States	1	
2280	UY	Uruguay	598	
2290	UZ	Uzbekistan	998	
2300	VA	Vatican City State	379	Yes
2310	VC	Saint Vincent and The Grenadines	1	
2320	VE	Venezuela	58	
2330	VG	Virgin Islands, British	1	
2340	VI	Virgin Islands, U.S.	1	
2350	VN	Vietnam	84	
2360	VU	Vanuatu	678	
2370	WF	Wallis and Futuna	681	
2380	WS	Samoa (American)	685	
2390	YE	Yemen	967	
2400	YT	Mayotte Island	262	
2410	ZA	South Africa	27	
2420	ZM	Zambia	260	
2430	ZW	Zimbabwe	263	

Excluded Acronyms

If you use the "Set to mixed case (Specific acronyms excluded)" Advanced Field Option (see **procedure_id**), the system will not transform the following text strings to mixed case:

- AB
- AB)
- ABW
- ABW)
- BC)
- BCBS
- BEHP
- BRHC
- BRHC)
- c/o
- CDPHP
- CEO
- CHCS
- CIO
- CJ
- CPA
- CVS
- DBA
- FEP
- FSC
- FSC)

- II
- III
- IV
- LLC
- LNR
- MD
- MSBCBS
- NCC
- NRECA
- NW
- PHD
- PO
- RN
- SW
- UAFC/CHCS
- VI
- VII
- VIII
- VP