Marketing cloud has Inbuilt SOAP and REST API that allows user to interact with MC server ,with this API object we can manage the Platform data (API is one which talk to user on behalf of MC server -this APi objects understands user request and get the work done for us from Marketing cloud Programmatically ) .

.There are many predefined API object used for various purpose.few important one listed below and site

<https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/supported_operations_for_objects_and_methods.html>

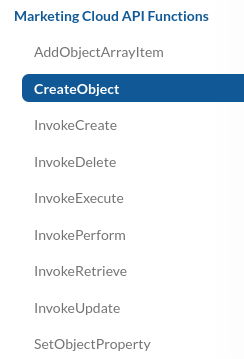
<https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/soap_web_service_objects.html>

=====================

In Salesforce Marketing Cloud (SFMC), there is a wide range of predefined API objects used to interact with different parts of the platform through the SOAP API. These objects allow you to perform tasks like creating, retrieving, updating, and deleting data. Some key objects include:

1. **DataExtension**: Manage data extensions (tables) and their records.
2. **Subscriber**: Manage subscribers, including their subscriptions to lists or data extensions.
3. **Send**: Used to send emails or messages and retrieve job data based on a JobID.
4. **TriggeredSend**: Automate the sending of emails based on specific events or triggers.
5. **Automation**: Create and manage automations for complex workflows.
6. **Campaign**: Track and manage marketing campaigns.

These objects follow predefined naming conventions and operations (e.g., create, retrieve, update,delete,execute,Perform ), meaning you cannot use random names when creating API objects. Instead, you need to stick to the predefined names like "RetrieveRequest" when using the CreateObject() function



In Salesforce Marketing Cloud (SFMC), API objects, like those created with CreateObject(), allow you to interact programmatically with the platform's various functionalities through its APIs, such as retrieving data, sending emails, or managing subscribers. Let's break it down:

### **Why Use API Objects in Marketing Cloud?**

API objects are used when you need to access, modify, or retrieve data in a programmatic way within SFMC, often bypassing the UI. They allow for automation and integration with external systems, providing powerful ways to manipulate data, trigger actions, or retrieve records dynamically.

### **Purpose of API Objects**

The main purposes of API objects are:

1. **Automation**: They enable tasks like sending emails or retrieving data to be automated.
2. **Integration**: API objects allow integration with external systems (such as CRMs or other marketing platforms) to exchange data.
3. **Customization**: Through API objects, developers can create custom solutions that fit their specific business logic.
4. **Advanced Data Management**: API objects can retrieve, update, or delete records from Data Extensions or Subscriber lists programmatically.

### **Difference Between Normal Objects and API Objects**

* **Normal Objects**: These are data structures and tools you work with through the UI, such as Data Extensions, Journey Builder, and emails. You interact with them using point-and-click tools.
* **API Objects**: API objects, like RetrieveRequest, interact with SFMC's APIs, meaning they are used in code to automate or customize processes. They provide programmatic control over SFMC features, and you manipulate them through languages like SSJS (Server-Side JavaScript) or AMPscript.

### **Naming of API Objects**

You **cannot** use any random name for API objects. The names must match the predefined API object names in the SFMC platform. For instance, RetrieveRequest is a specific object that instructs the API to retrieve data. If you were to use an arbitrary name, SFMC wouldn’t recognize it, and the call would fail.

For example, this code:

javascript

Copy code

set @rr = CreateObject("RetrieveRequest")

is creating an instance of the RetrieveRequest API object, which is used to retrieve data from the platform.

Salesforce Marketing Cloud provides a set of predefined API objects to enable interaction with various features and data. Here's a list of some commonly used **predefined API objects** and their **usages** in Marketing Cloud:

### **1. RetrieveRequest**

* **Usage**: Used to retrieve data from objects like Data Extensions, Subscriber lists, and other API objects.
* **Example**: Retrieve specific subscribers from a list or rows from a Data Extension.

**Code**:  
javascript  
Copy code  
set @rr = CreateObject("RetrieveRequest")

### **2. Subscriber**

* **Usage**: Represents a subscriber (recipient) in the Marketing Cloud email system. You can use this to add, update, or retrieve subscriber information.
* **Example**: Add a new subscriber or update their status.

**Code**:  
javascript  
Copy code  
set @sub = CreateObject("Subscriber")

My view-Create a Human subscriber object and give individual property for this and create in MC platform using InvokeCreate(Subobj).will see how to code this

NOTE: Instantiated objects do not persist after the first subsequent Invoke function call (for example, InvokeRetrieve(), InvokeUpdate(), etc.).

**1.Create the Subscriber Object**:

* Platform.Function.CreateObject("Subscriber") creates a new subscriber object.

2.Use Platform.Function.SetObjectProperty() to set properties like EmailAddress, SubscriberKey, and Status.

**3.Assign List Membership**:

* A SubscriberList object is created to assign the subscriber to a specific list. The list is identified by its **ID**.
* Use AddObjectArrayItem() to associate the subscriber with this list.

**4.Invoke Create/Update**:

* InvokeCreate(subscriber) performs the operation, creating or updating the subscriber.
* The result is checked to see if the operation was successful.

### **Usage Notes:**

* If the subscriber already exists, this operation will update the existing record.
* If the subscriber is new, it will be created with the properties defined.

<script runat="server">

// Create a Subscriber object

var subscriber = Platform.Function.CreateObject("Subscriber");

// 3 properties for this Subscriber ,Set the Email Address property

Platform.Function.SetObjectProperty(subscriber, "EmailAddress", "john.doe@example.com");

// Set the SubscriberKey (often this is the same as the email, but can be different)

Platform.Function.SetObjectProperty(subscriber, "SubscriberKey", "john.doe@example.com");

// Set the Status (Active, Unsubscribed, Held, etc.)

Platform.Function.SetObjectProperty(subscriber, "Status", "Active");

// Now create another Object to represent Suncriber List ,Add subscriber to a specific List

var subscriberList = Platform.Function.CreateObject("SubscriberList");

Platform.Function.SetObjectProperty(subscriberList, "ID", "12345"); // List ID goes here

Platform.Function.SetObjectProperty(subscriberList, "Status", "Active");//2 properties

// add the Subcriberobject to a List array -subcriber is Object ,Lists is property name and Sublist is property of List

**Platform.Function.AddObjectArrayItem(subscriber, "Lists", subscriberList);**

// Invoke the creation of the Subscriber

var result = Platform.Function.InvokeCreate(subscriber);

// Output result

if (result.Status == "OK") {

Write("Subscriber created/updated successfully.");

} else {

Write("Error occurred: " + result.Status);

}

</script>

Syntax:

AddObjectArrayItem(1,2,3)

Ordinal Type Required Description

1 API Object True The API Object that contains the array

2 String True The array property associated with the item to be added

3 String True The item to add to the array

==============================================

### **3. DataExtension**

* **Usage**: Represents a Data Extension (a table where you store custom data in Marketing Cloud). You can use this to manipulate Data Extension records.
* **Example**: Add, update, or retrieve data from a Data Extension.

**Code**:  
javascript  
Copy code  
set @de = CreateObject("DataExtension")

<script runat="server">

// Create a DataExtension object

var dataExtension = Platform.Function.CreateObject("DataExtension");

// Set the properties of the DataExtension

Platform.Function.SetObjectProperty(dataExtension, "CustomerKey", "Sample\_DE\_Key");

Platform.Function.SetObjectProperty(dataExtension, "Name", "Sample Data Extension");

Platform.Function.SetObjectProperty(dataExtension, "Description", "A sample data extension created using SSJS.");

Platform.Function.SetObjectProperty(dataExtension, "IsSendable", true); // Set to true if this Data Extension will be used for sending emails

Platform.Function.SetObjectProperty(dataExtension, "SendableDataExtensionField", "EmailAddress");

Platform.Function.SetObjectProperty(dataExtension, "SendableSubscriberField", "SubscriberKey");

// Define fields for the DataExtension

var fields = Platform.Function.CreateObject("Field");

Platform.Function.SetObjectProperty(fields, "Name", "EmailAddress");

Platform.Function.SetObjectProperty(fields, "FieldType", "EmailAddress");

Platform.Function.SetObjectProperty(fields, "IsPrimaryKey", true); // This field is the primary key

Platform.Function.SetObjectProperty(fields, "MaxLength", 254);

var fields2 = Platform.Function.CreateObject("Field");

Platform.Function.SetObjectProperty(fields2, "Name", "FirstName");

Platform.Function.SetObjectProperty(fields2, "FieldType", "Text");

Platform.Function.SetObjectProperty(fields2, "MaxLength", 50);

// Add the fields to the DataExtension

Platform.Function.AddObjectArrayItem(dataExtension, "Fields", fields);

Platform.Function.AddObjectArrayItem(dataExtension, "Fields", fields2);

// Invoke the create request to create the Data Extension

var result = Platform.Function.InvokeCreate(dataExtension);

// Output the result

if (result.Status == "OK") {

Write("Data Extension created successfully.");

} else {

Write("Error occurred: " + result.Status);

}

</script>

==================

### **4. Email**

* **Usage**: Represents an email message in Marketing Cloud. This can be used to retrieve information about emails or trigger specific emails.
* **Example**: Retrieve details about a specific email or trigger an email send.

**Code**:  
javascript  
Copy code  
set @email = CreateObject("Email")

Prerequisite:create a email in email studio and get the Id for the email .

<script runat="server">

Platform.Load("core", "1.1.5");

// Create the email object

var email = Platform.Function.CreateObject("Email");

// Set the email properties, such as ID of the email to send

Platform.Function.SetObjectProperty(email, "ID", "123456"); // Replace with your Email ID

// Create the subscriber object

var subscriber = Platform.Function.CreateObject("Subscriber");

Platform.Function.SetObjectProperty(subscriber, "EmailAddress", "example@example.com"); // Replace with recipient's email

Platform.Function.SetObjectProperty(subscriber, "SubscriberKey", "example@example.com"); // SubscriberKey, usually the email in many cases

// Create the TriggeredSendDefinition object

var tsd = Platform.Function.CreateObject("TriggeredSendDefinition");

Platform.Function.SetObjectProperty(tsd, "CustomerKey", "MyTriggeredSend"); // Replace with your TriggeredSend CustomerKey

// Create the TriggeredSend object

var ts = Platform.Function.CreateObject("TriggeredSend");

Platform.Function.SetObjectProperty(ts, "TriggeredSendDefinition", tsd);

Platform.Function.AddObjectArrayItem(ts, "Subscribers", subscriber);

// Invoke the Perform method to send the email

var status = Platform.Function.InvokePerform(ts, "Start");

// Log status to check success or failure

if (status[0].Status == "OK") {

Write("Email sent successfully.");

} else {

Write("Failed to send email. Error: " + status[0].ErrorCode);

}

</script>

### **5. TriggeredSend**

* **Usage**: Used to send triggered emails from Marketing Cloud. You can create, send, and manage emails based on specific triggers, like user actions.
* **Example**: Send a confirmation email when a purchase is made.

**Code**:  
javascript  
Copy code  
set @ts = CreateObject("TriggeredSend")

### **6. SendDefinition**

* **Usage**: Defines the details of an email send. This object is used when setting up email sends, including recipients and configurations.
* **Example**: Set up and execute an email send to a specific group.

**Code**:  
javascript  
Copy code  
set @sd = CreateObject("SendDefinition")

===========================

### **7. Interaction**

* **Usage**: Represents a Journey in Journey Builder. It is used to manage interactions (journeys) that customers go through.
* **Example**: Retrieve or update information about customer journeys.

**Code**:  
javascript  
Copy code  
set @interaction = CreateObject("Interaction")

### **8. List**

* **Usage**: Represents a subscriber list within Marketing Cloud. You can use this to manipulate lists, like adding subscribers or retrieving list data.
* **Example**: Add subscribers to a list or retrieve information about a list.

**Code**:  
javascript  
Copy code  
set @list = CreateObject("List")

### **9. Campaign**

* **Usage**: Represents a campaign object in Marketing Cloud. You can manage and track campaigns using this object.
* **Example**: Create a new campaign or retrieve information about existing campaigns.

**Code**:  
javascript  
Copy code  
set @campaign = CreateObject("Campaign")

### **10. TrackingEvent**

* **Usage**: Used to retrieve tracking data such as opens, clicks, bounces, and unsubscribes for emails.
* **Example**: Get information about email performance and engagement metrics.

**Code**:  
javascript  
Copy code  
set @tracking = CreateObject("TrackingEvent")

### **11. Job**

* **Usage**: Represents a send job (an instance of an email send). You can use this to manage email sends and their status.
* **Example**: Retrieve details about an email send job or monitor its status.

**Code**:  
javascript  
Copy code  
set @job = CreateObject("Job")

### **12. Folder**

* **Usage**: Represents a folder where assets like emails, data extensions, and lists are stored in Marketing Cloud.
* **Example**: Retrieve or create a folder to organize assets.

**Code**:  
javascript  
Copy code  
set @folder = CreateObject("Folder")

### **13. BounceEvent**

* **Usage**: Used to retrieve information on email bounce events.
* **Example**: Track and report bounced emails.

**Code**:  
javascript  
Copy code  
set @bounce = CreateObject("BounceEvent")

### **14. UnsubEvent**

* **Usage**: Used to retrieve information about unsubscribe events.
* **Example**: Track and report unsubscribe events for email sends.

**Code**:  
javascript  
Copy code  
set @unsub = CreateObject("UnsubEvent")

### **15. ClickEvent**

* **Usage**: Represents a click event from a recipient in an email, allowing you to track email click activity.
* **Example**: Retrieve which subscribers clicked on links in an email.

**Code**:  
javascript  
Copy code  
set @click = CreateObject("ClickEvent")

### **16. OpenEvent**

* **Usage**: Tracks and retrieves email open events from subscribers.
* **Example**: Monitor who opened the email and how many times.

**Code**:  
javascript  
Copy code  
set @open = CreateObject("OpenEvent")

### **17. ContentArea**

* **Usage**: Represents a specific content area within an email, which you can retrieve, update, or create.
* **Example**: Retrieve content for use in emails or updates to content blocks.

**Code**:  
javascript  
Copy code  
set @content = CreateObject("ContentArea")