Automating Salesforce Marketing Cloud

Reap all the benefits of the SFMC platform and increase your productivity with the help of real-world examples

Greg Gifford | Jason Hanshaw
Foreword by Guilda Hilaire, Director, Product Marketing, Salesforce, Inc.

Technical requirements

Code for the book is available at this link: https://github.com/PacktPublishing/Automating-Salesforce-Marketing-Cloud

Chapter 1

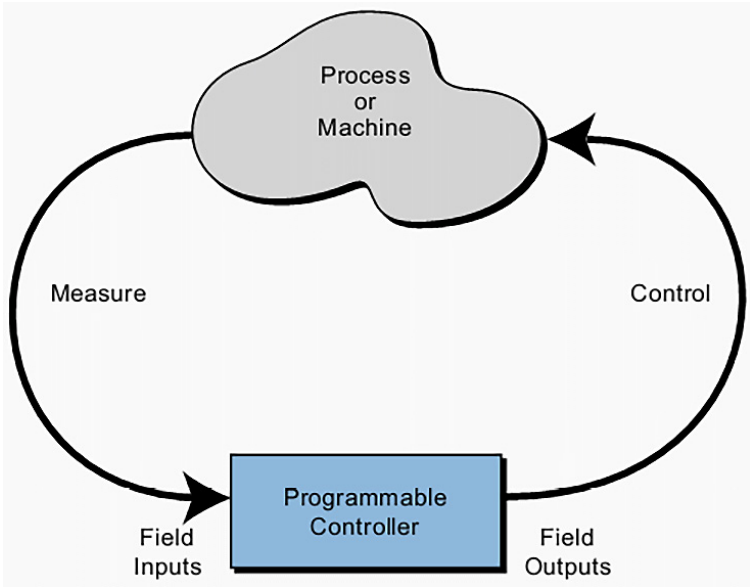

Figure 1.1 – Visual representation of automation theory

Chapter 2, Part 1

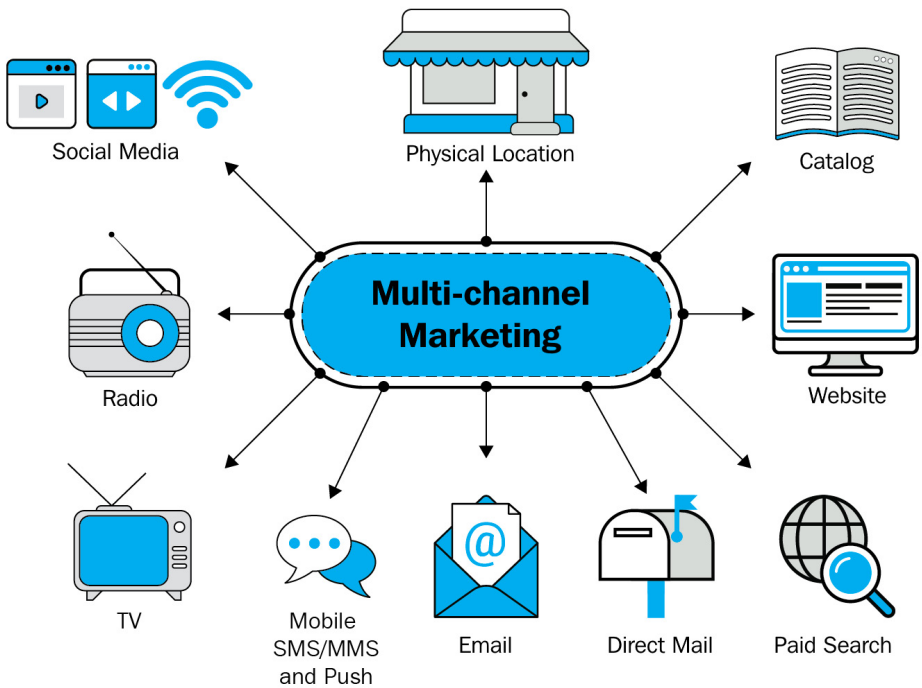

Figure 2.1 – Visualization of multi-channel marketing

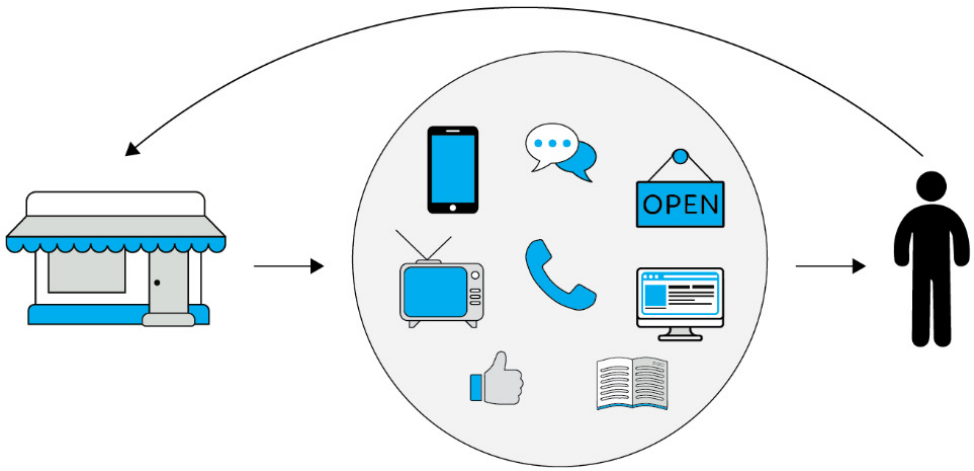

Figure 2.2 – Visualization of cross-channel marketing

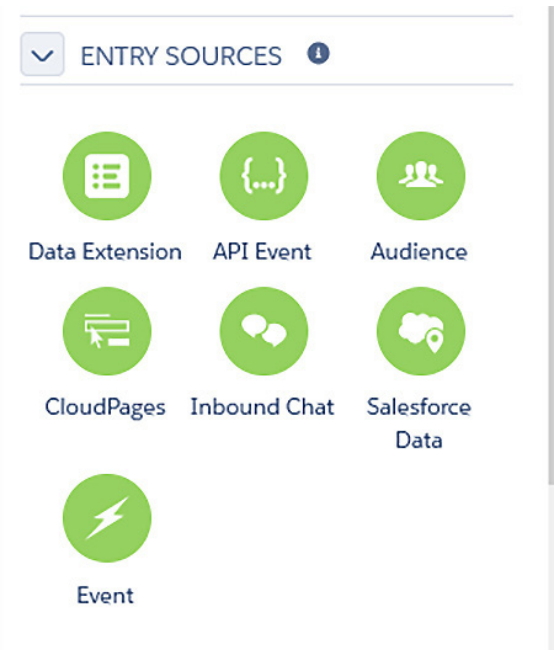

Figure 2.3: Examples of the entry sources available in Journey Builder

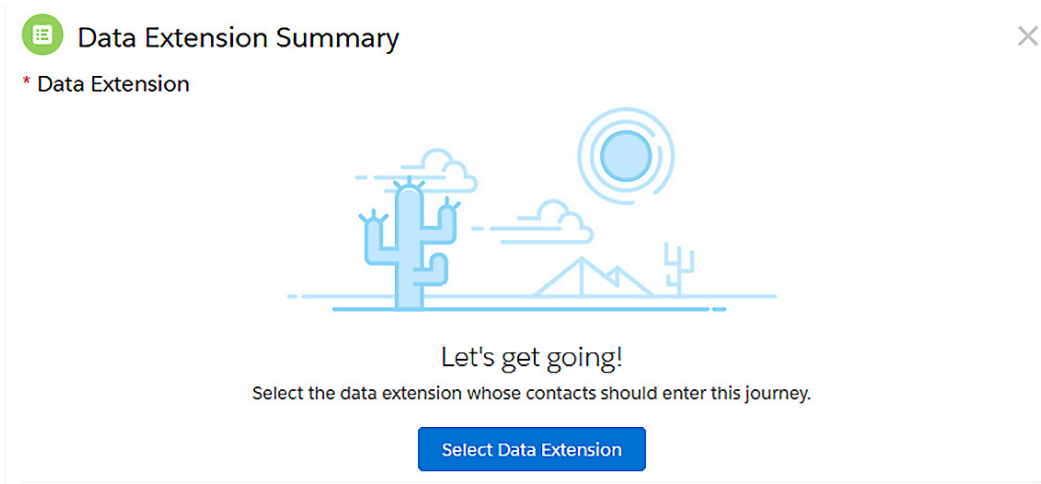

Figure 2.4: View of the data extension entry source summary

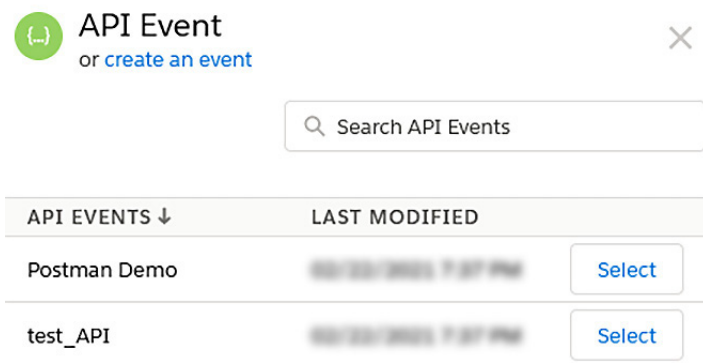

Figure 2.5: Example of the API event entry source

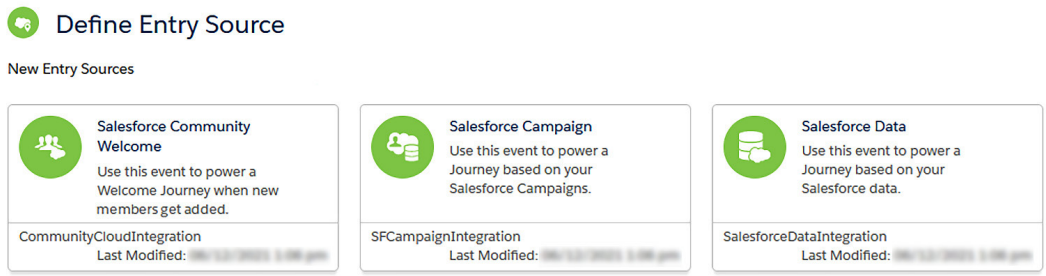

Figure 2.6: Example screen for setting up the Salesforce event entry source

Chapter 2, Part 2

Figures

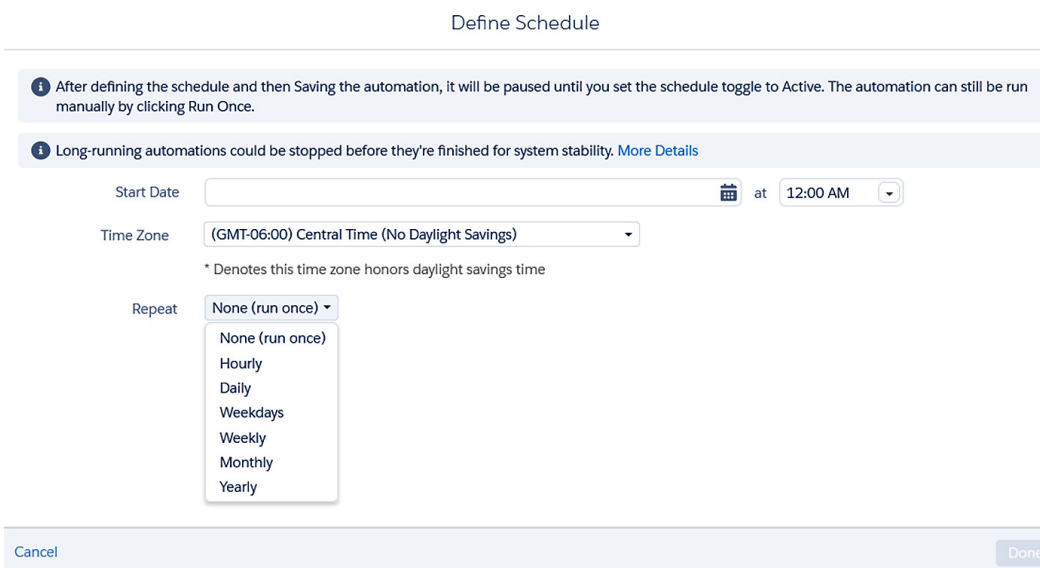

Figure 2.7: An example of the setup screen when creating a scheduled automation

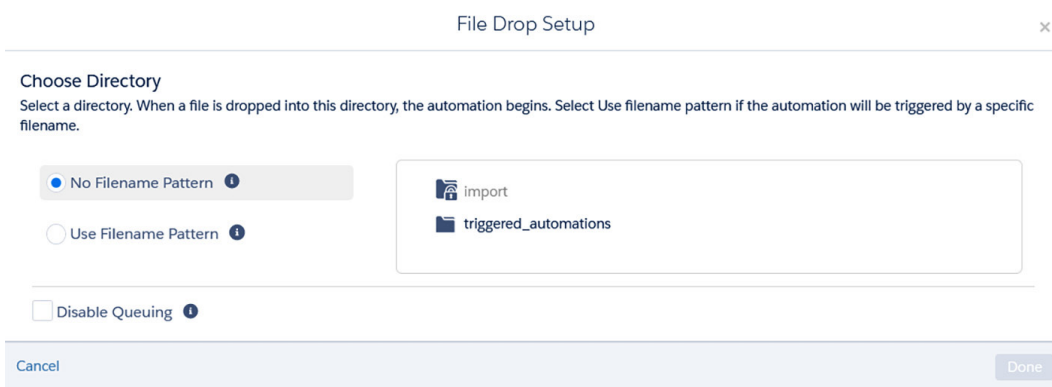

Figure 2.8: An example showing the setup window for a file drop automation

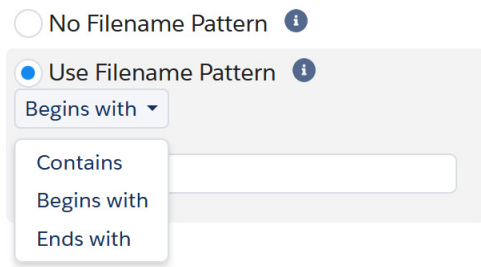

Figure 2.9: Filename pattern options

Chapter 3

Figures

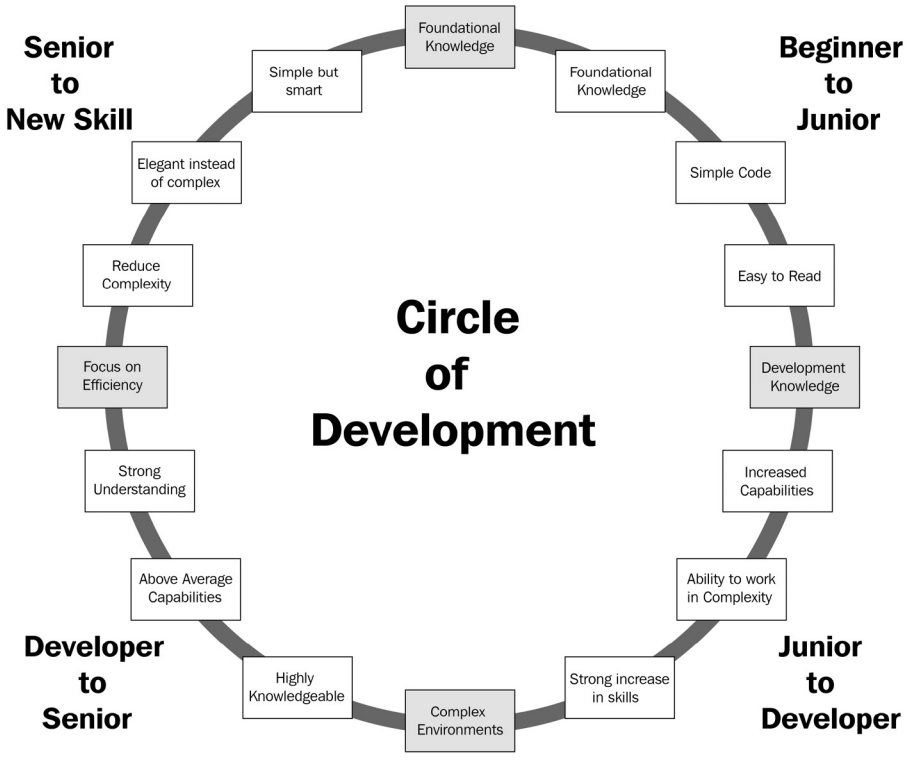

Figure 3.1: The circle of development

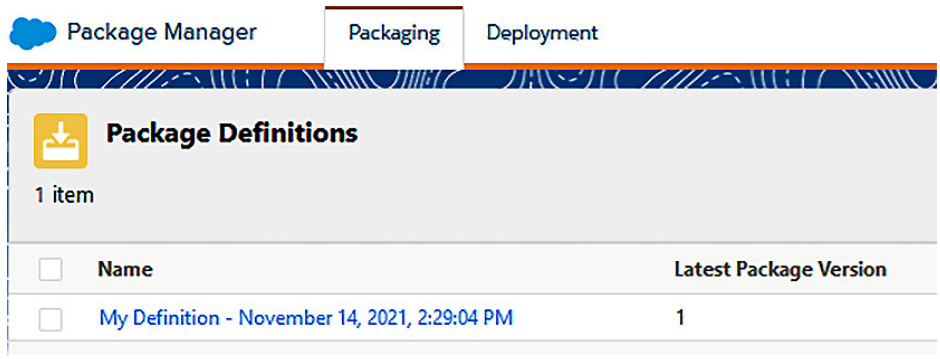

Figure 3.2: The Package Manager interface

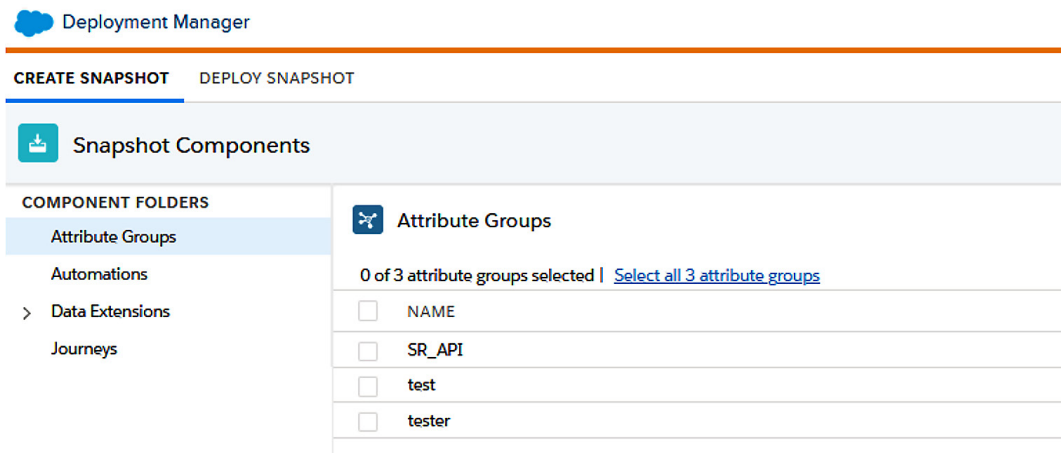

Figure 3.3: The Deployment Manager interface

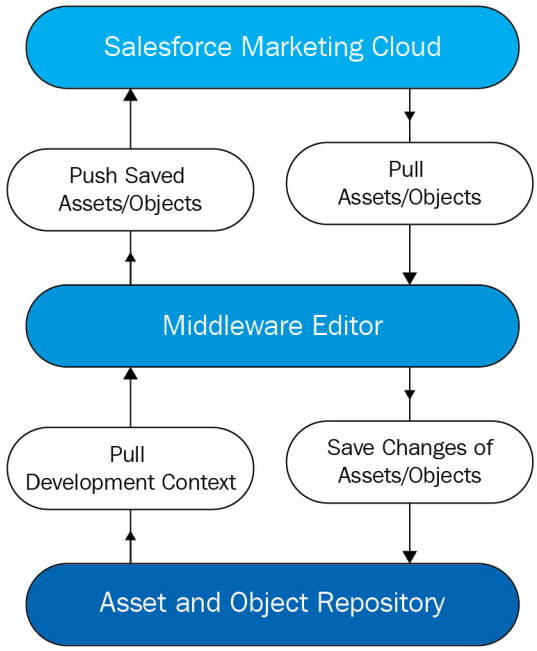

Figure 3.4: Example flow of external development and testing

Code blocks

Example of a complex solution – Utilizing something like the following is a possible solution:

```
var arr = []
arr[0] = "First";
arr[1] = "Second";
arr[2] = "Third";
arr[3] = "Fourth";
```

However, this is much more complex and explicit than just doing the following:

```
var arr = ["First","Second","Third","Fourth"]
```

An example of elegant processing

```
var arr = ["First","Second","Third","Fourth"]
arr.sort(function(x, y) {
  if (x < y) {
```

```
      return -1;
    }
    if (x > y) {
      return 1;
    }
    return 0;
  });
  //returns ["First","Fourth","Second","Third"]
```

Chapter 4

Links

https://dma.org.uk/uploads/misc/marketers-email-tracker-2019.pdf

Figures

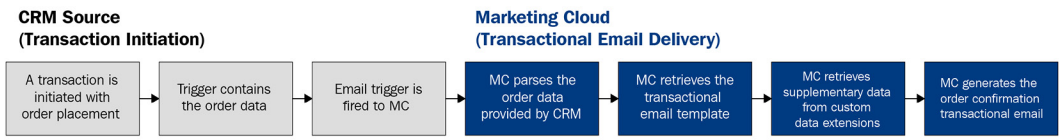

Figure 4.1: Example of workflow for the transactional email API

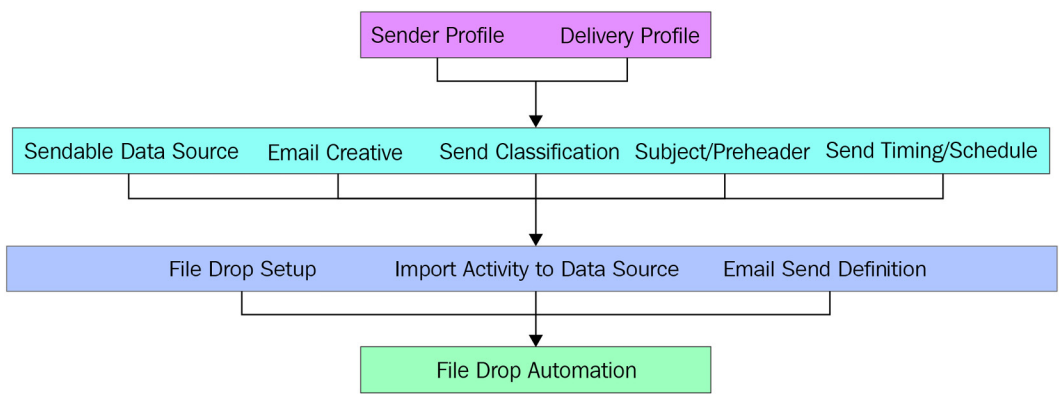

Figure 4.2: Diagram of each of the pieces of a File Drop Automation email send

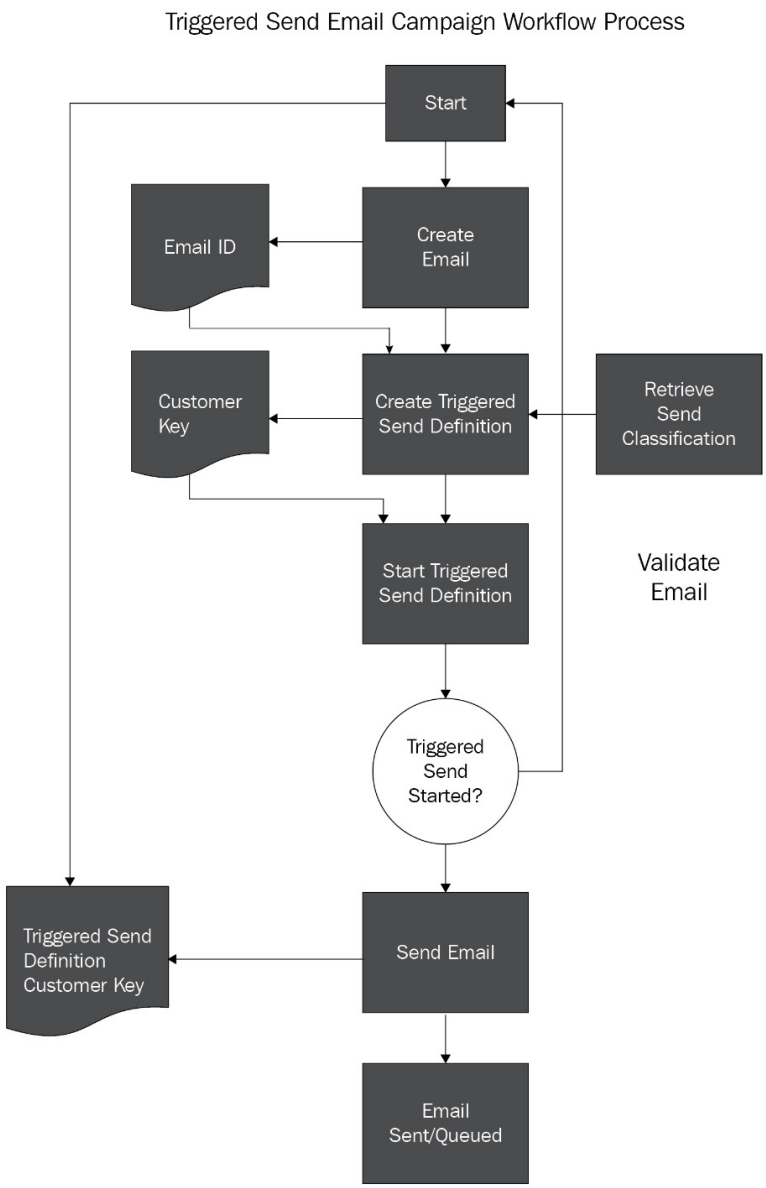

Figure 4.3: Representation of the triggered send workflow

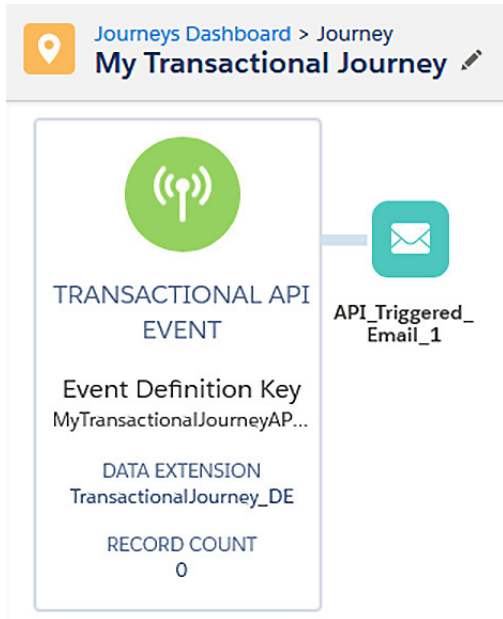

Figure 4.5: Sample of transactional journey in Journey Builder

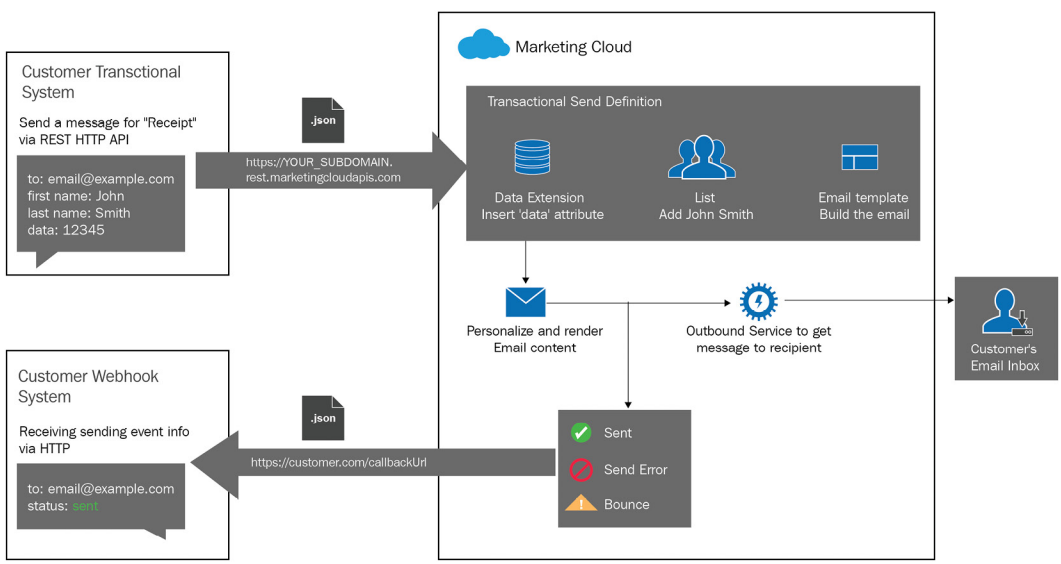

Figure 4.4: Representation of the Transactional API capabilities

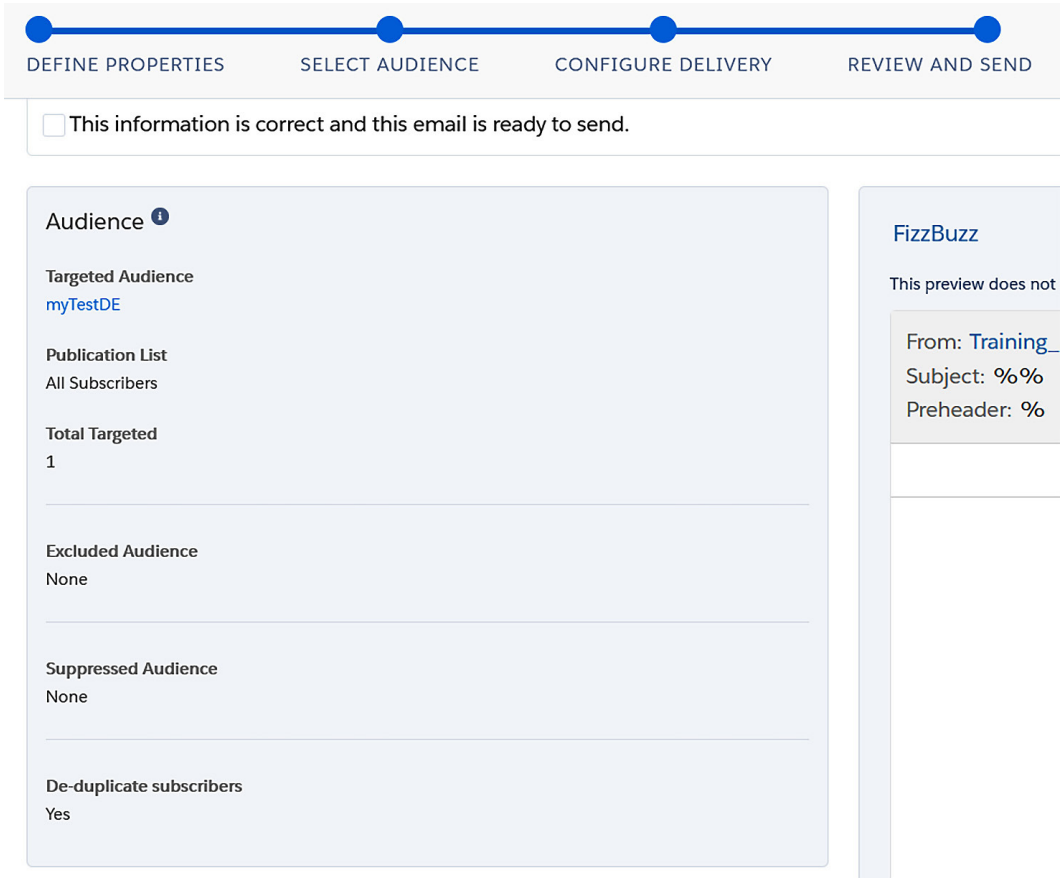

Figure 4.6: Example view of the Email Studio Send Wizard

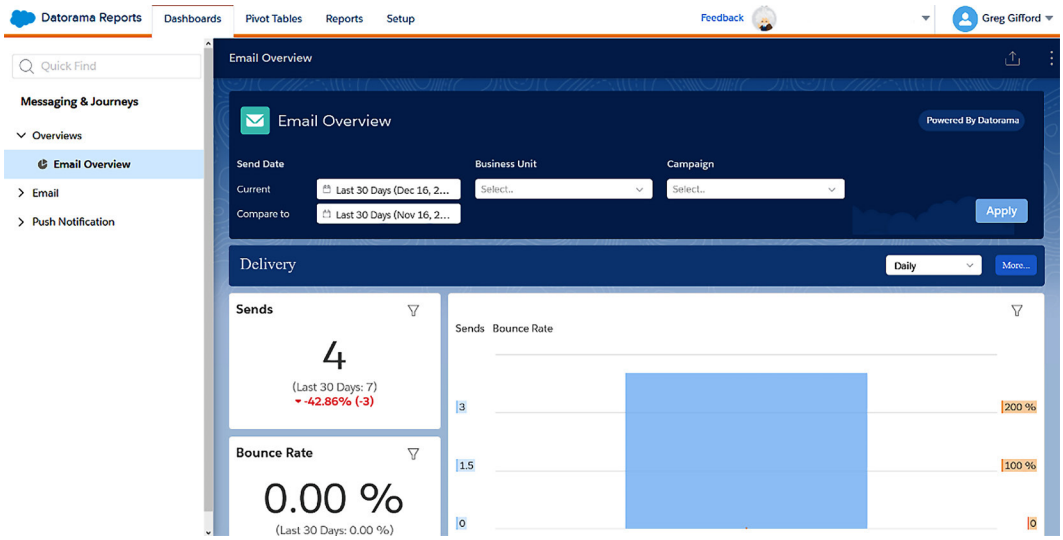

Figure 4.7: Example view of the Datorama Reports overview screen

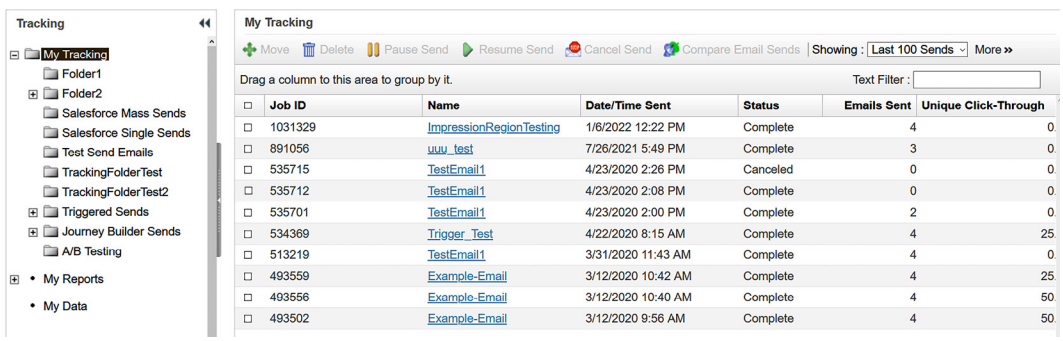

Figure 4.8: Example of the Tracking tab inside Salesforce Marketing Cloud

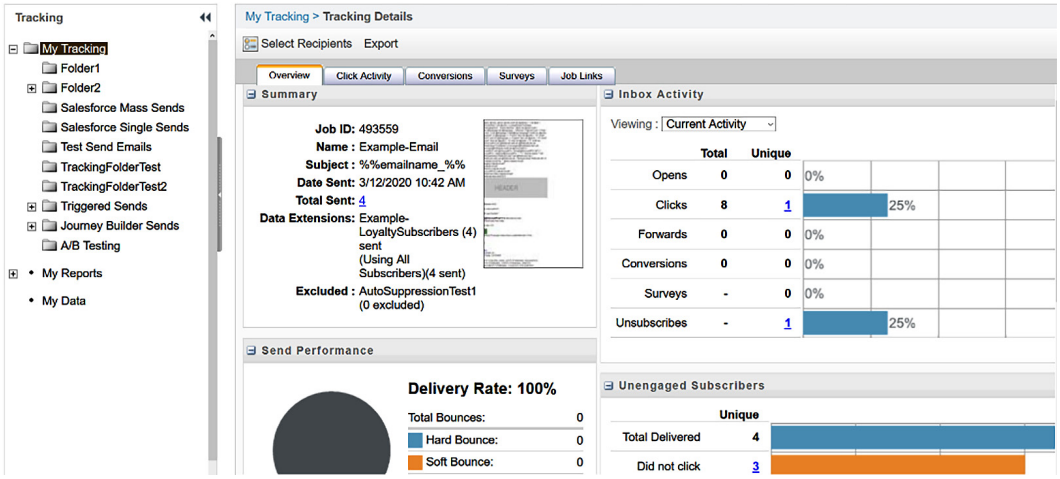

Figure 4.9: Tracking information for a send job inside the Tracking tab

Chapter 5

Links

- https://help.salesforce.com/s/articleView?id=mc_as_using_the_query_activity.htm&type=5&language=en_US
- https://www.w3schools.com/sql/default.asp

Figures

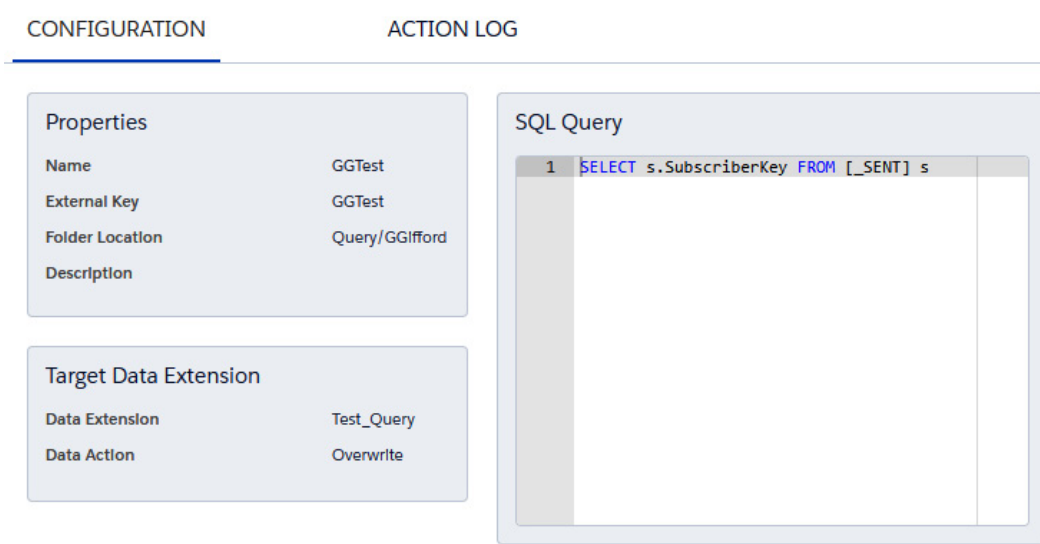

Figure 5.1: Example of the UI for a Query activity

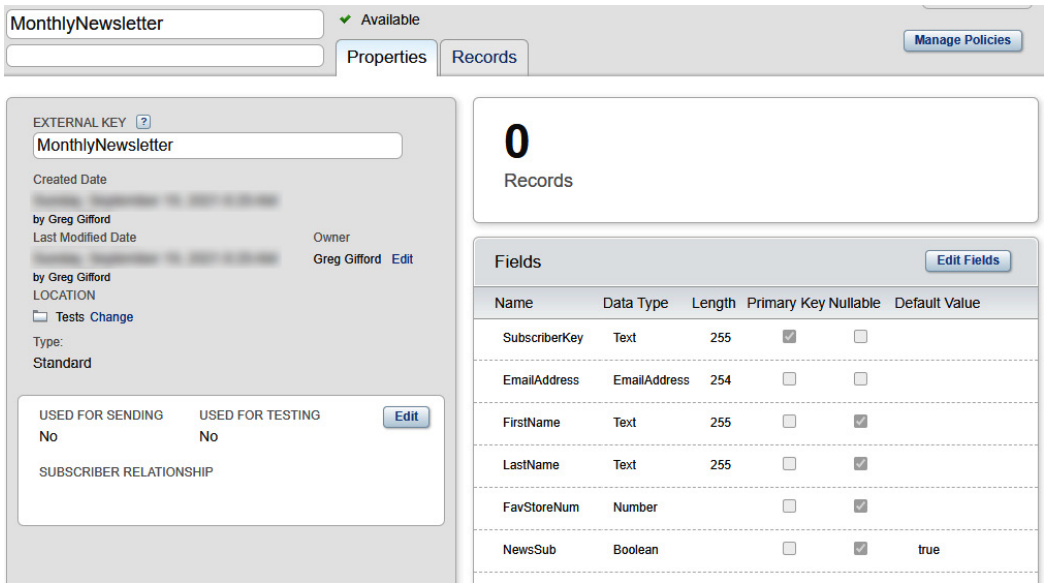

Figure 5.2: Screenshot of the MonthlyNewsletter data extension

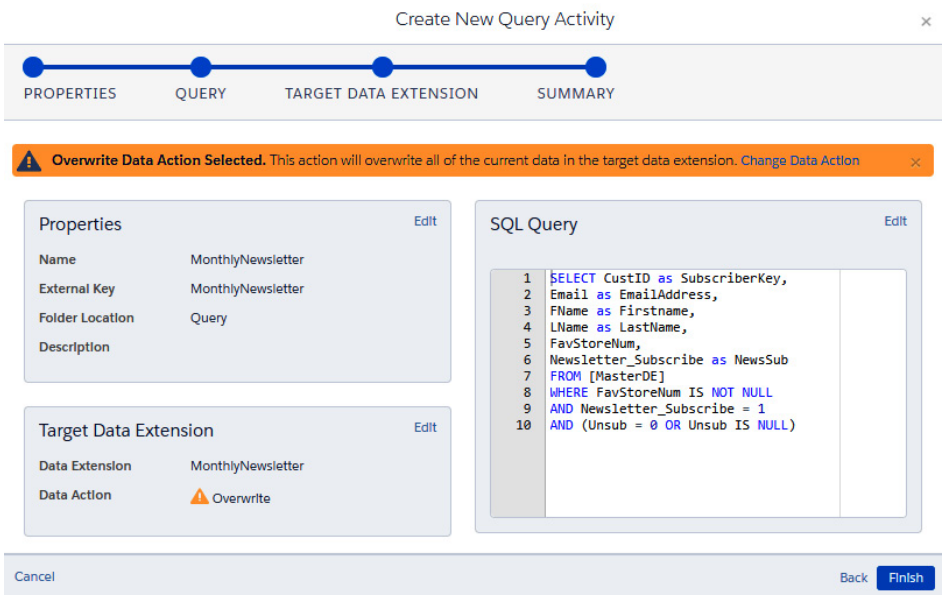

Figure 5.3: Screenshot of the MonthlyNewsletter Query Activity overview

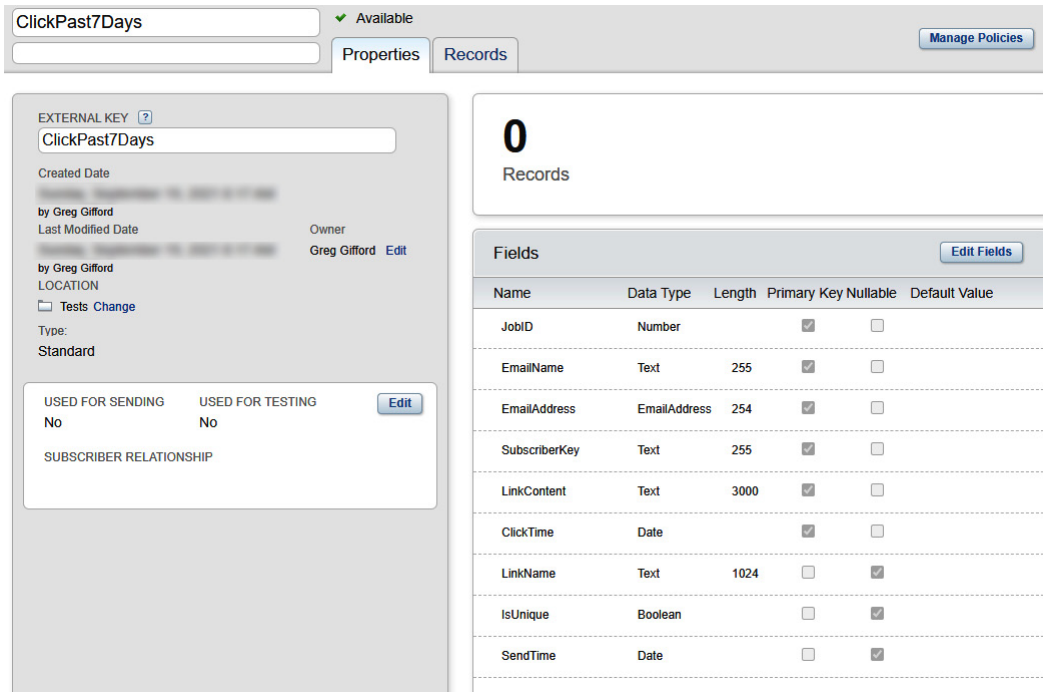

Figure 5.4: Screenshot of the ClickPast7Days data extension

Figure 5.5: Screenshot of the ClickPost7Days Query activity overview

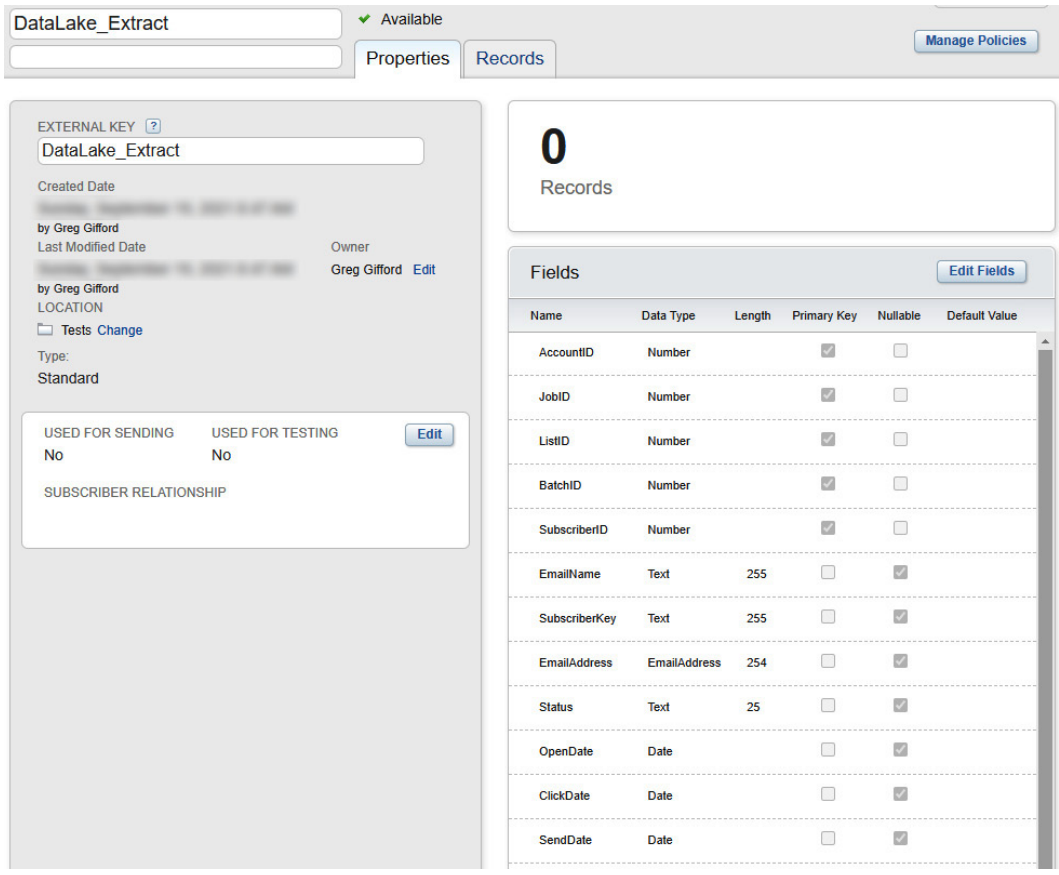

Figure 5.6: Screenshot of the DataLake_Extract data extension

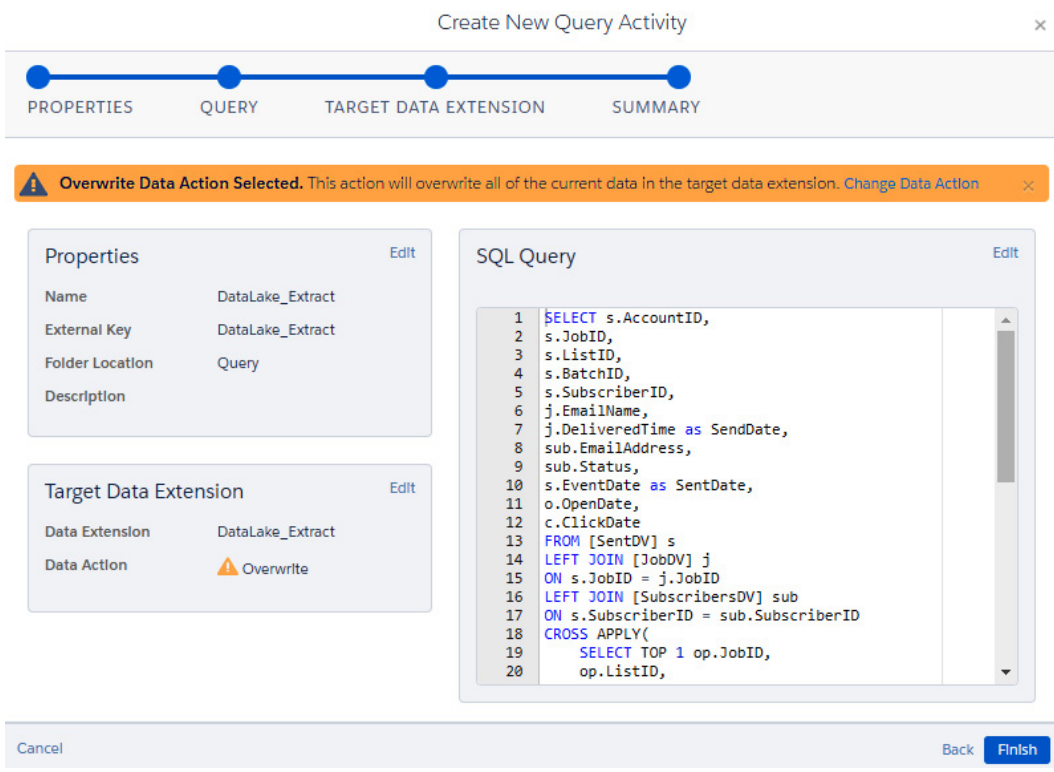

Figure 5.7: Screenshot of the DataLake_Extract Query activity overview

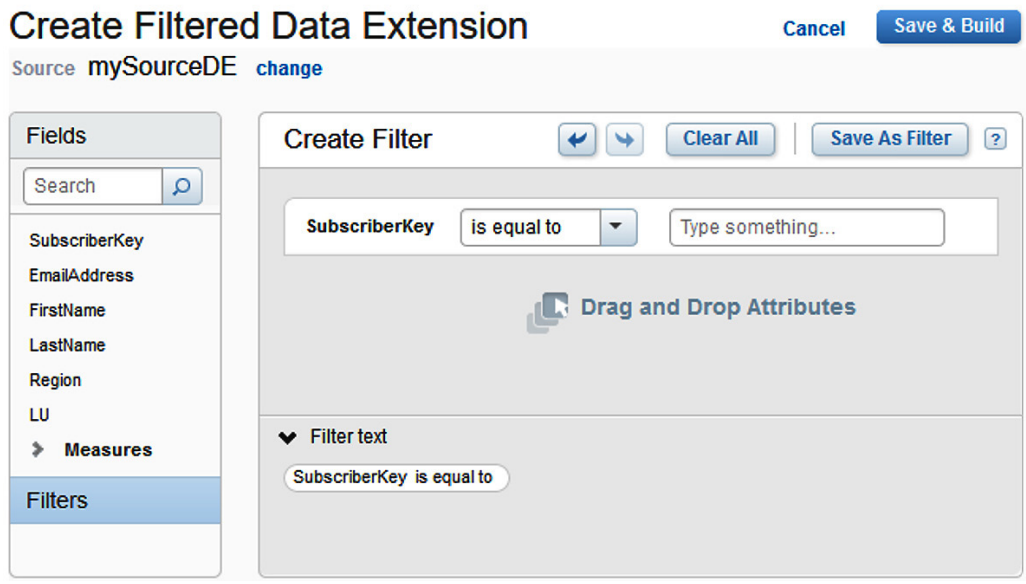

Figure 5.8: Drag and drop segmentation interface for filtered data extensions

Name	Type	Sendable	Record Count	Last Modified Date	Status	Actions
myFilteredDE	Filtered	No	74	Monday, September 06, 2021 7:47 AM	Available	
mySourceDE	Standard	No	152	Wednesday, December 08, 2021 10:43 PM	Available	

Figure 5.9: Example of a filtered data extension in U.I.

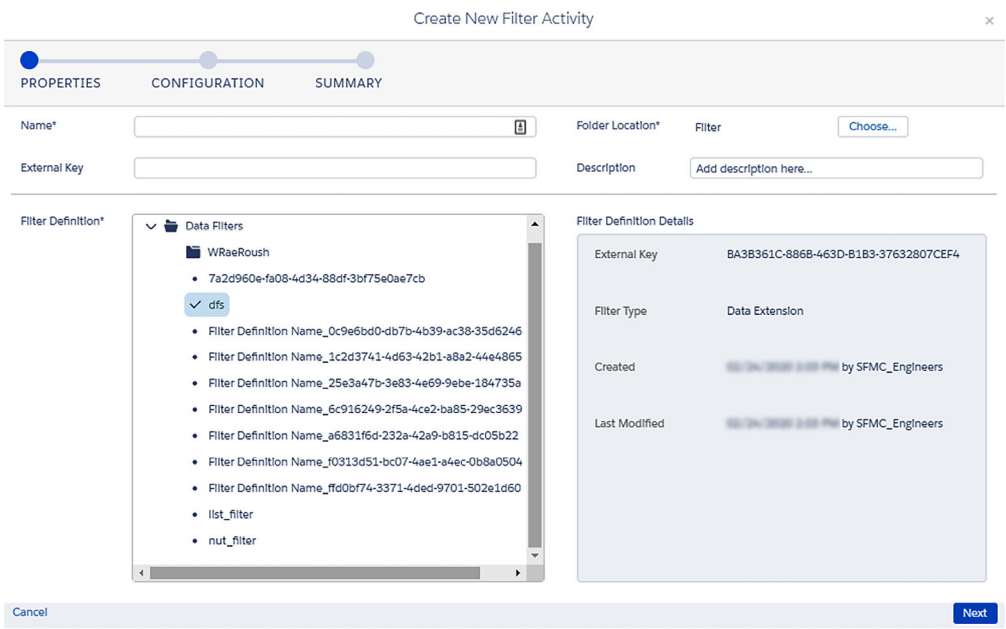

Figure 5.10: Example of the setup for creating a new filter activity

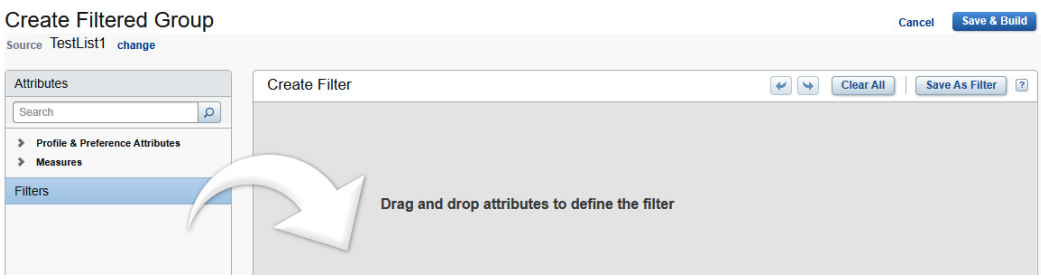

Figure 5.11: Example of the setup for creating a new filtered group

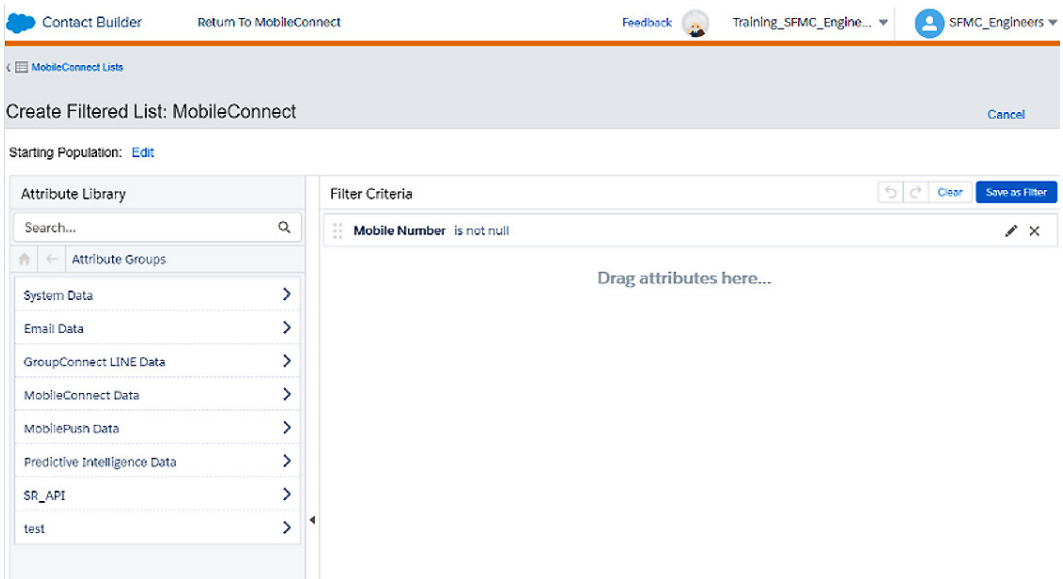

Figure 5.12: The U.I. for creating a filtered mobile list

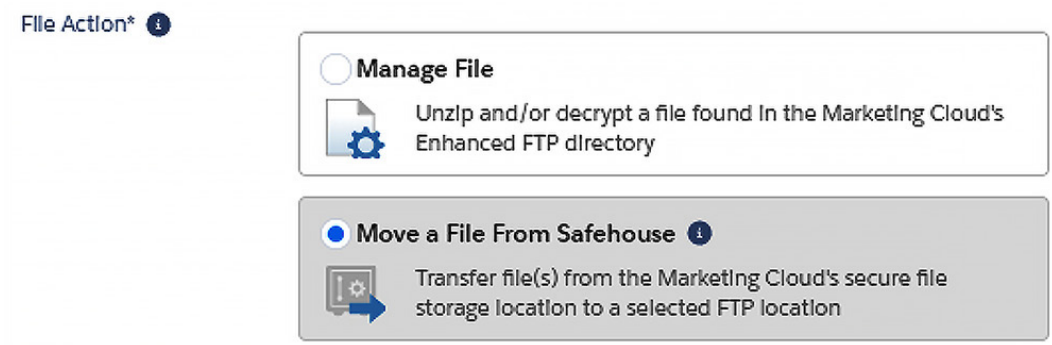

Figure 5.13: Two action options available for File Transfer

Tables

NAME	DATA TYPE	LENGTH	PRIMARY KEY	NULLABLE	DEFAULT
SubscriberKey	Text	255	Y		
EmailAddress	EmailAddress				
FirstName	Text	255		Y	
LastName	Text	255		Y	
FavStoreNum	Number			Y	
NewsSub	Boolean			Y	Y

Table 5.1: Fields and properties of the MonthlyNewsletter data extension

NAME	DATA TYPE	LENGTH	PRIMARY KEY	NULLABLE	DEFAULT
AccountID	Number		Y		
JobID	Number		Y		
ListID	Number		Y		
BatchID	Number		Y		
SubscriberID	Number		Y		
EmailName	Text	255		Y	
SubscriberKey	Text	255		Y	
EmailAddress	EmailAddress			Y	
Status	Text	25		Y	
OpenDate	Date			Y	
ClickDate	Date			Y	
SendDate	Date			Y	

Table 5.3: Fields and properties of the DataLake_Extract data extension

Code blocks

Code block 5.1: SQL query activity in automation studio

```
SELECT CustID as SubscriberKey,
Email as EmailAddress,
FName as Firstname,
LName as LastName,
FavStoreNum,
Newsletter_Subscribe as NewsSub
FROM [MyMasterDE]
WHERE FavStoreNum IS NOT NULL
AND Newsletter_Subscribe = 1
AND (Unsub = 0 OR Unsub IS NULL)
```

Code block 5.2:

```
SELECT j.JobID,
j.EmailName,
s.EmailAddress,
c.SubscriberKey,
c.LinkName,
c.LinkContent,
c.IsUnique,
j.DeliveredTime as SendTime,
c.EventDate as ClickTime
FROM [_Job] j
LEFT JOIN [_Click] c
ON j.JobID = c.JobID
LEFT JOIN [_Subscribers] s
ON c.SubscriberKey = s.SubscriberKey
WHERE c.EventDate > dateadd(d,-7,getdate())
```

Code block 5.3

```
SELECT s.AccountID,
s.JobID,
s.ListID,
s.BatchID,
s.SubscriberID,
j.EmailName,
j.DeliveredTime as SendDate,
sub.EmailAddress,
sub.Status,
s.EventDate as SentDate,
o.OpenDate,
c.ClickDate
FROM [SentDV] s
LEFT JOIN [JobDV] j
ON s.JobID = j.JobID
LEFT JOIN [SubscribersDV] sub
ON s.SubscriberID = sub.SubscriberID
CROSS APPLY(
    SELECT TOP 1 op.JobID,
    op.ListID,
    op.BatchID,
     op.SubscriberID,
    MAX(op.EventDate) as OpenDate
    FROM [OpenDV] op
    WHERE op.JobID = s.JobID
    AND op.ListID = s.ListID
    AND op.BatchID = s.BatchID
    AND op.SubscriberID = s.SubscriberID
    GROUP BY op.JobID, op.ListID, op.BatchID,
      op.SubscriberID
) o
CROSS APPLY(
    SELECT TOP 1 cl.JobID,
    cl.ListID,
```

```
    cl.BatchID,
    cl.SubscriberID,
    MAX(cl.EventDate) as ClickDate
    FROM [ClickDV] cl
    WHERE cl.JobID = s.JobID
    AND cl.ListID = s.ListID
    AND cl.BatchID = s.BatchID
    AND cl.SubscriberID = s.SubscriberID
    GROUP BY cl.JobID, cl.ListID, cl.BatchID,
      cl.SubscriberID
) c
WHERE s.EventDate > DATEADD(day,-7,s.EventDate)
```

Code block 5.4

```
Method: POST
Endpoint: /email/v1/filteredCustomObjects/{{filterDEid}}/
refresh
Host: {{tenantSubDomain}}.rest.marketingcloudapis.com
Authorization: Bearer {{auth_token}}
Content-Type: application/json
```

Chapter 6

Links

- Javascript Overview: https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript
- SSJS: https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/ssjs_serverSideJavaScript.html
- AMP Script: https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/ampscript.html
- First class function: https://developer.mozilla.org/en-US/docs/Glossary/First-class_Function

Table

NAME	DATA TYPE	LENGTH	PRIMARY KEY (PK)	NULLABLE	DEFAULT
MID	Number		Y		
TimeStamp	Date		Y		
QueueAlertArrayStr	Text	MAX		Y	

Table 6.1 – Fields and properties in myQueryQueue_LogDE DE

Code blocks

Code block 6.1

Script activity

```
<script runat="server">
  Platform.Function.ContentBlockByKey(
    "ampscriptContentBlock");
</script>
```

Code block 6.2

AMPscript content block

```
%%[
SET @sde = "mySourceDE"
SET @tde = "myTargetDE"
SET @rs = LookupRows(@sde, "HasBeenRun", "False")
SET @rc = RowCount(@rs)
FOR @i=1 TO @rc  DO
  SET @row = ROW(@rs,@i)
  SET @pkey = FIELD(@row,"pkey")
  SET @val2 = FIELD(@row,"val2")
  SET @val3 = FIELD(@row,"val3")
   UPSERTDATA(@tde, 1, "pkey",@pkey,"val2",@val2,"val3",@val3)
NEXT @i

]%%
```

Code block 6.3

SSJS inline AMPscript

```
<script runat=server>
Var newStr = Platform.Function.TreatAsContent('\%\%[SET @name
=ProperCase("greg Gifford") OUTPUT(CONCAT(@name)) ]\%\%')
</script>
```

Code block 6.4

The following code will create three different variable names:

```
<script runat=server>
  Platform.Load("Core","1.1.1");
  var arr = [0,1,2]
  var vName = "myVar"
  for (var i = 0; i < arr.length; i++) {
    Write(vName + arr[i] + '<br>')
  }
</script>
```

The preceding code would output the following result:

```
myVar1
myVar2
myVar3
```

Code block 6.5

```
var fruit = {
  fruitType: "Apple",
  subType: "Golden Delicious",
  fruitId: 123456,
  displayName : function() {
   return this.subType + " " + this.fruitType;
  }
};
//which would lead to being called like:
var display = fruit.displayName()
//Output: Golden Delicious Apple
//to get just the definition, you would remove the ()
var display = fruit.displayName() //Output: function() { return
this.subType + " " + this.fruitType;  }
```

Code block 6.6

```
//create empty object
var obj = {}
//create object with values
var obj = {name:"value"}
//set or update name:value pair to obj
obj.myName = "myValue"  // {name:"value",myName:"myValue"}
//set or update name:value pair to obj square brackets
obj["myName"] = "myValue"
// {name:"value",myName:"myValue"}
```

Code block 6.7

```
var obj = {
  vehicle: "car",
  make: "Ford",
  model: "Fusion",
  year: 1998,
  color: "grey",
  lastOilChange: "09/21/20XX",
  milage: 139990
}
```

Code block 6.8

See the following `for...in` loop example:

```
<script runat=server>
  var obj = {name1:"val1", name2:"val2"}
  for (var x in obj) {
    Write(x + '<br>');  //writes the string value of each
                        //property
  }
</script>
```

Code block 6.9

Array of objects

```
var arr = [
  {
    id: 1234,
    email: "myName@example.com",
    name: "John Doe"
  },
  {
    id: 2345,
    email: "myName2@example.com",
    name: "Jane Doe"
  },
  {
```

```
    id: 3456,
    email: "myName3@example.com",
    name: "Malcolm Doe"
  }
]
```

Code block 6.10

Object with an array

```
var obj = {
  id: 1234,
  name: "John Doe",
  email: "myName@example.com",
  relatives: [
    "Jane Doe",
    "Malcolm Doe",
    "Mindy Jo Doe"
  ]
}
```

Code block 6.11

Examples of funciton in SSJS

- Here is an example of a function's returned values being *named as variables*:

```
Var foo = function() { //my function script }
console.log(foo); //runs function()
```

- And here's an example of values being *passed as arguments to procedures*:

```
function callback (foo) {
    Foo();
}
Callback(function(){console.log('Successfully passed')})
```

- Here's an example of values being *returned as the result of procedures*:

```
function(foo) {
   if(foo == 'pass') {
      return function(){ return 'It worked!'};
```

```
    } else {
       return function() { return 'It failed!'};
    }
  }
```

- And here's an example of values being *stored in a variable for reference*:

```
  var sum = function (a,b) { return (a+b)}
  sum(4,4); //returns 8
```

Code block 6.12

```
<script runat=server>
  Platform.Load("Core","1.1.1");
  var ret;
  try {
    var req = new
      Script.Util.HttpGet("https://www.example.com");
    var resp = req.send();
  } catch(e) {
    Write(Stringify(e));
  }
</script>
```

Code block 6.13

```
function getTSDKeys(mid) {
  /* Set ClientID */
  if (mid) {
    prox.setClientId({ "I.D.": mid }); //Impersonates the B.U.
  }
  var cols = ["CustomerKey", "TriggeredSendStatus"];
  var res = prox.retrieve("TriggeredSendDefinition",
    cols, filter);
  return res;
}
```

Code block 6.14

```
function getTSDQueue(customerKey) {
  /* Set ClientID */
  if (mid) {
    prox.setClientId({ "ID": mid }); //Impersonates the BU
  }
  var cols = ["CustomerKey","Queued"];
  var filter = {
      Property: "CustomerKey",
      SimpleOperator: "Equals",
      Value: customerKey
  };
  var res = prox.retrieve("TriggeredSendSummary",
    cols, filter);
  var queue = res.Results[0].Queued
  return queue;
}
```

Code block 6.15

```
// Global variables
var prox = new Script.Util.WSProxy();
var mid = 123456;
var failures = 0;
var allTriggers = 1; // All triggers in BU, or false (0) if
                     // want specific filters only
var alertArray = []
```

Code block 6.16

```
if (allTriggers) {
  var tsdArray = getTSDKeys(mid);
  var length = tsdArray.Results.length;
} else {
  var tsdArray =
["TriggerA","TriggerB","TriggerC","TriggerD","TriggerE"]
// External Keys of the Triggers you want to check.
  var length = tsdArray.length;
```

```
  var maxQueueArray = [500,500,500,500,500];
// Enter max queue here. Can make an array as well, if
// different maxes per TSD
}
if (!maxQueueArray || maxQueueArray.length == 0) {
  var maxQueueDefault = 500; // Default for if not using
                             // maxQueueArray
}
```

Code block 6.17

```
for (i=0; i < length; i++) {
    if(allTriggers) {
      var customerKey = tsdArray.Results[i].CustomerKey
    } else {
      var customerKey = tsdArray[i]
    }
    var queued = getTSDQueue(customerKey);
    var queueArrLength = maxQueueArray.length;
// changes maxQueue to array value if exist and equal to i
    if (maxQueueArray.length > 0 &&
      maxQueueArray.length <= i) {
      var maxQueue = maxQueueArray[i];
    } else {
      var maxQueue = maxQueueDefault;
    }
    if (queued > maxQueue) {
      //creates the failure object
      var obj = {}
      obj.customerkey = customerKey;
      obj.queue = queued;
      //pushes the failure obj to array
      alertArray.push(obj)
      failures += 1 //increases failure count
    }
}
```

Code block 6.18

```
if (failures > 0) {
  // Upserts into log DE
  var rows =
    Platform.Function.UpsertData("myQueryQueue_LogDE",
    ["MID","TimeStamp"], [mid,getDate()],
    ["QueueAlertArrayStr"], [Stringify(alertArray)])
}
```

Code block 6.19

```
%%[
SET @emailaddr = "sample@example.com"
SET @subkey = 'Sub123456'
SET @tsObj = CreateObject("TriggeredSend")
SET @tsDefObj = CreateObject("TriggeredSendDefinition")
SET @tsSubObj = CreateObject("Subscriber")
SetObjectProperty(@tsDefObj, "CustomerKey", "MyTriggeredSend")
SetObjectProperty(@tsObj, "TriggeredSendDefinition", @tsDefObj)
SetObjectProperty(@tsSubObj, "EmailAddress", @emailaddr)
SetObjectProperty(@tsSubObj, "SubscriberKey", @subkey)
AddObjectArrayItem(@tsObj, "Subscribers", @tsSubObj)
SET @tsCode = InvokeCreate(@tsObj, @tsMsg, @error)
]%%
```

Bullet list

Natively, you can do the following actions in relation to arrays:

- Gather the length of the array (that is, the total number of indexed values), as exemplified here:

```
fruit.length //returns 3
```

- Turn an array into a delimited string, as exemplified here:

```
fruit.join('|') //returns Apples|Oranges|Bananas
```

- Remove the last element of an array, as exemplified here:

```
fruit.pop() //returns ["Apples","Oranges"]
```

- Create your own prototypes to use on the `Array()` method, as follows:

```
Array.prototype.myUcase = function() {
  for (i = 0; i < this.length; i++) {
    this[i] = this[i].toUpperCase();
  }
};
var fruit = ["Banana", "Orange", "Apple", "Mango"];
fruit.myUcase();
//Output: ["BANANA", "ORANGE", "APPLE", "MANGO"]
```

- Sort array values based on a passed function, as follows:

```
var array = [40, 100, 1, 5, 25, 10];
array.sort(function(x, y) {
  if (x < y) {
    return -1;
  }
  if (x > y) {
    return 1;
  }
  return 0;
});
//Output: [1,5,10,25,40,100]
```

- Slice out selected elements in an array as a new array, as follows:

```
var fruits = ["Banana", "Orange", "Lemon", "Apple",
"Mango"];
var citrus = fruits.slice(1, 3);
//Output: ["Orange","Lemon"]
```

Chapter 7

Code blocks

Code block 7.1

```
%%[
set @payload = '{
"ID":111213,
"FirstName":"Gor",
"LastName":"Tonington",
"TotalAmt":125
}'
set @postrequest = HTTPPost2("https://myAPIURL.
com","application/json", @payload, true)
]%%
```

Code block 7.2

```
//Creates new object(s) inside of the identified SOAP Object
function createGeneric(soapObjName, contentJSON, mid) {
    //example soapObjName: "DataExtension"
    //example contentJSON Object: { "CustomerKey": custkey,
    //"Name": name, "Fields": fields };
    //example contentJSON Array: [{ "CustomerKey": custkey,
    //"Name": name, "Fields": fields },{ "CustomerKey":
    //name, "Name": name, "Fields": fields }]
    //set default date (in a Data Extension) to
    //'Current Date': { FieldType: "Date", Name: "Field2",
    //DefaultValue: "getdate()" }
    if(mid) {
      prox.resetClientIds(); //reset previous settings
```

```
        // Set ClientID
        prox.setClientId({ "ID": mid });
      }
      var batch = isArray(contentJSON);
      if(batch) {
        var res = prox.createBatch(sopObjName,contentJSON);
      } else {
        var res = prox.createItem(sopObjName,contentJSON);
      }
      function isArray(arg) {
        return Object.prototype.toString.call(arg) ===
          '[object Array]';
      };
      return res;
    }
```

Code block 7.3

```
<script runat="server">
Platform.Load("Core","1.1.1")
var url = "https://{{et_subdomain}}.rest.marketingcloudapis.
com/email/v1/rest"
var req = new Script.Util.HttpGet(url);
var resp = req.send();</script>
```

Code block 7.4

```
<script runat=server>
Platform.Load("core", "1.1.1");
var accessToken = {{yourToken}};
var url = 'https:// {{et_subdomain}}.rest.marketingcloudapis.
com/asset/v1/content/assets/{{ContentID}}'
var payload = '{{yourPayload}}';
var auth = 'Bearer ' + accessToken;
var req = new Script.Util.HttpRequest(url);
req.emptyContentHandling = 0;
req.retries = 2;
req.timeout = 30
```

```
req.continueOnError = true;
req.contentType = "application/json"
req.setHeader("Authorization", auth);
req.method = "PUT"; /*** You can change the method here ***/
req.postData = payload;
var resp = req.send();
</script>
```

Code block 7.5

```
function createQuery(tenant,authToken,payload) {
    vvar url =  'https://' + tenant   +  '.rest.
marketingcloudapis.com/automation/v1/queries/';
    var req = new Script.Util.HttpRequest(url);
    req.emptyContentHandling = 0;
    req.retries = 2;
    req.continueOnError = true;
    req.contentType = "application/json"
    req.setHeader("Authorization", authToken);
    req.method = "POST";
    req.postData = Stringify(payload);
    var resp = req.send();
    var resultStr = String(resp.content);
    var resultJSON =
      Platform.Function.ParseJSON(String(resp.content));
    Write(resultStr);
    return resultJSON;
}
```

Code block 7.6

Example payload

```
{
    "name": "myQuery",
    "key": "myQuery",
    "description": "",
    "queryText": "select *\n FROM [myDE]",
    "targetName": "myQueryDE",
```

```
    "targetKey": "myQueryDE",
    "targetDescription": "",
    "targetUpdateTypeId": 0,
    "targetUpdateTypeName": "Overwrite",
    "categoryId": 123456,
    "isFrozen": false
}
```

Code block 7.7

```
{
    "name": "myQuery",
    "queryText": "select *\n FROM [myDE]",
    "targetKey": "myQueryDE",
    "targetUpdateTypeId": 0,
    "categoryId": 123456
}
```

Code block 7.8

```
function updateQuery(tenant,authToken,queryID,payload) {
    var url =  'https://' + tenant
  +  '.rest.marketingcloudapis.com/automation/v1/queries/'
  + queryID;
    var req = new Script.Util.HttpRequest(url);
    req.emptyContentHandling = 0;
    req.retries = 2;
    req.continueOnError = true;
    req.contentType = "application/json"
    req.setHeader("Authorization", authToken);
    req.method = "PATCH";
    req.postData = Stringify(payload);
    var resp = req.send();
    var resultStr = String(resp.content);
    var resultJSON =
      Platform.Function.ParseJSON(String(resp.content));
    return resultJSON;
}
```

Code block 7.9

```
<script runat="server">
    var prox = new Script.Util.WSProxy();
    /* Set ClientID */
    prox.setClientId({ "ID": mid}); //Impersonates the BU
    var props = [
       { Name: "SubscriberKey", Value: "sample@sample.com" },
      { Name: "EmailAddress", Value: "sample@sample.com" },
      { Name: "JobID", Value: 18099 },
      { Name: "ListID", Value: 8675309 },
      { Name: "BatchID", Value: 0 }
    ];

  var data = prox.execute(props, "LogUnsubEvent");
</script>
```

Code block 7.10

```
function clearDE(custKey) {
  var prox = new Script.Util.WSProxy();
  var action = "ClearData";
  var props = {
          CustomerKey: custKey
  };
  var data = prox.performItem("DataExtension", props,
    action);
  return data;
}
```

Code block 7.11

```
function getUser(host,token,userID) {
    var url = host + '/api/users/' + userID;
    var req = new Script.Util.HttpRequest(url);
    req.emptyContentHandling = 0;
    req.retries = 2;
    req.continueOnError = true;
```

```
    req.contentType = "application/json"
    req.method = "GET";
    req.setHeader("Authorization", token);
    var resp = req.send();
    var resultStr = String(resp.content);
    var resultJSON
      Platform.Function.ParseJSON(String(resp.content));
    return resultJSON;
}
```

Chapter 8

Figures

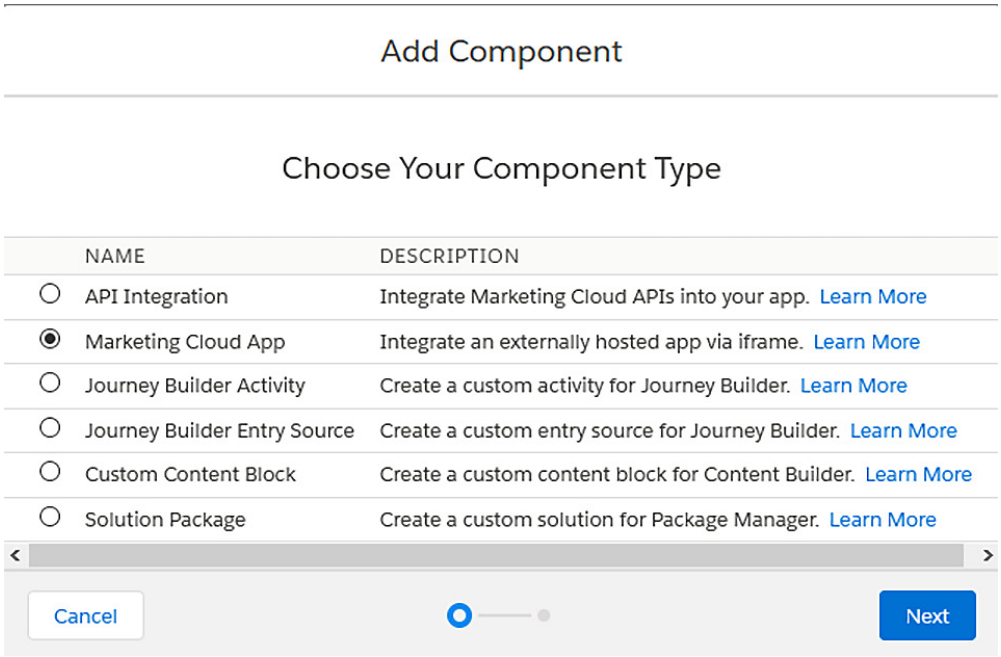

Figure 8.1: Example of Marketing Cloud App component selection

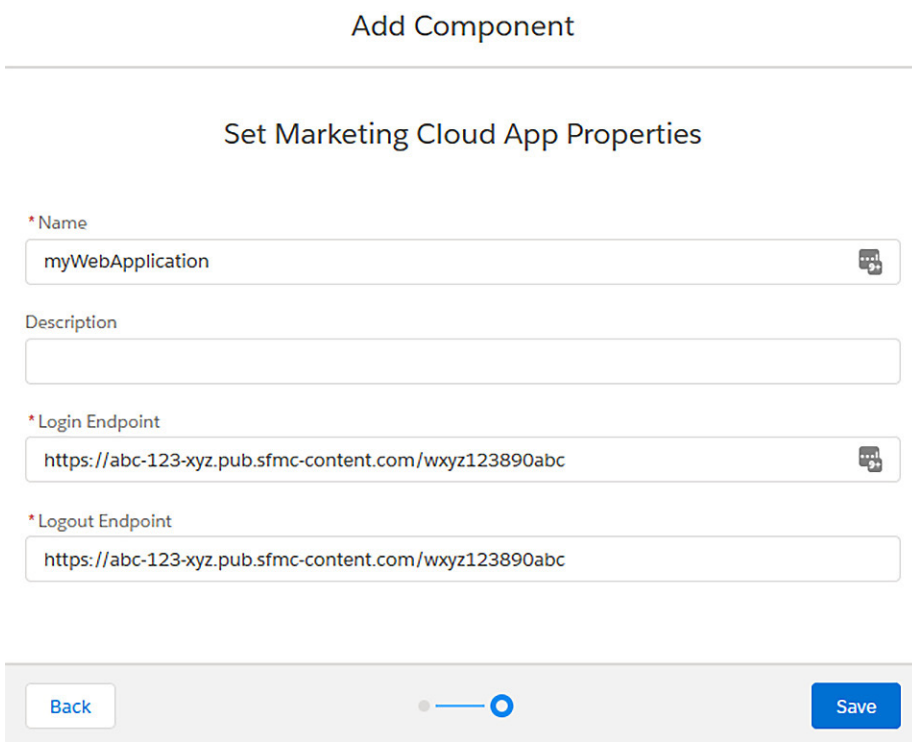

Figure 8.2, Example of filled out Marketing Cloud app properties

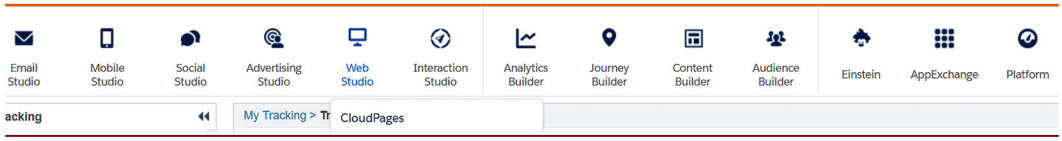

Figure 8.3, Example showing how to get to the Cloud Pages application

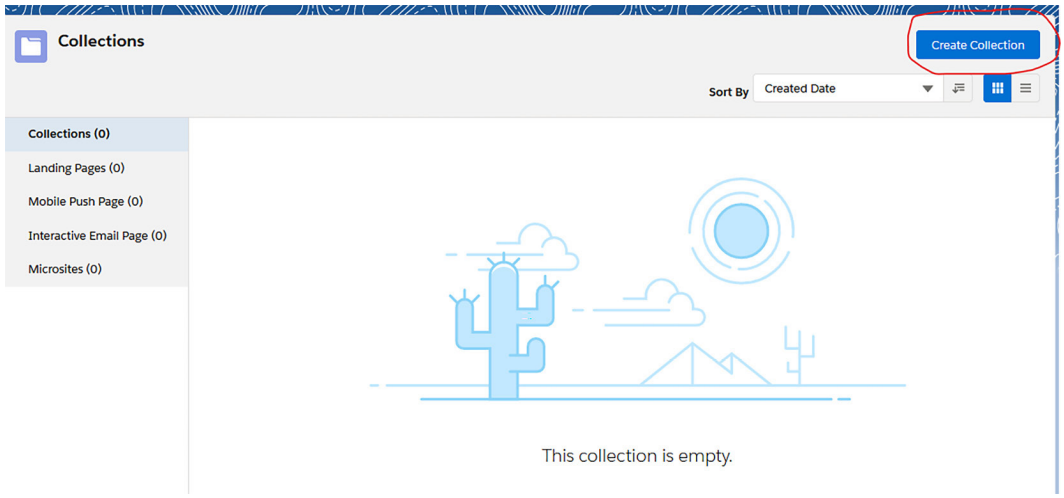

Figure 8.4, Example of creating a new collection inside Cloud Pages

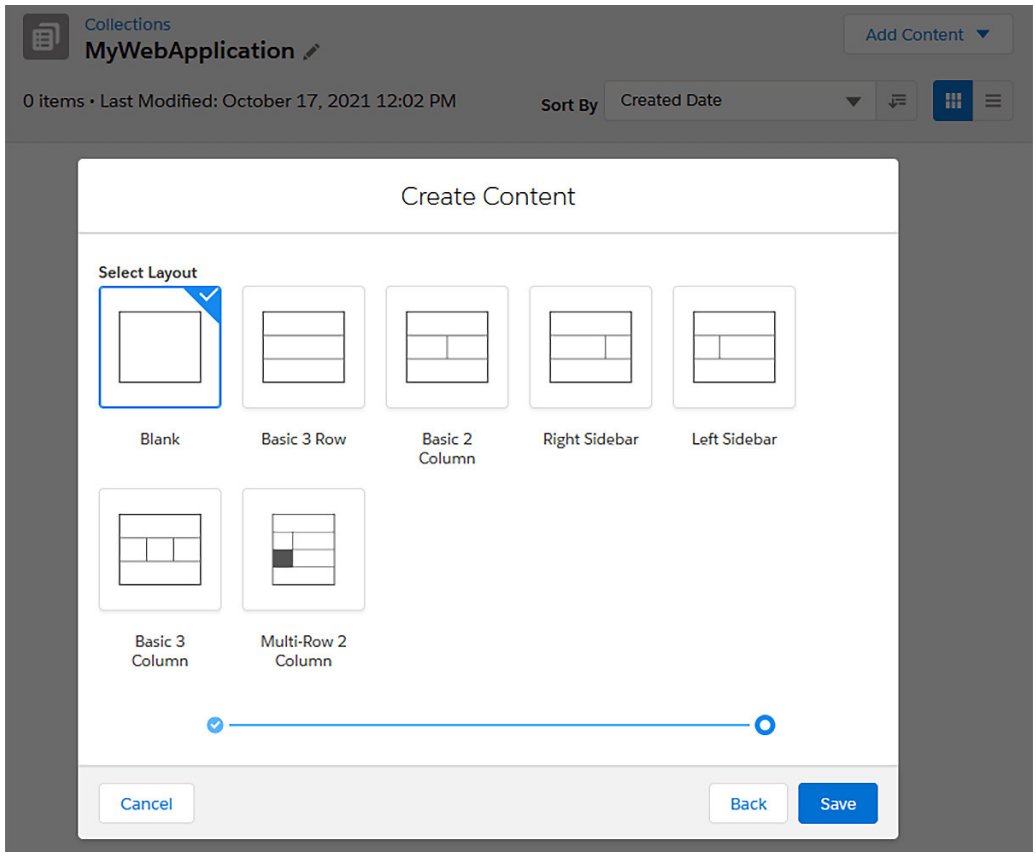

Figure 8.5: Example of creating a landing page inside of a collection in Marketing Cloud

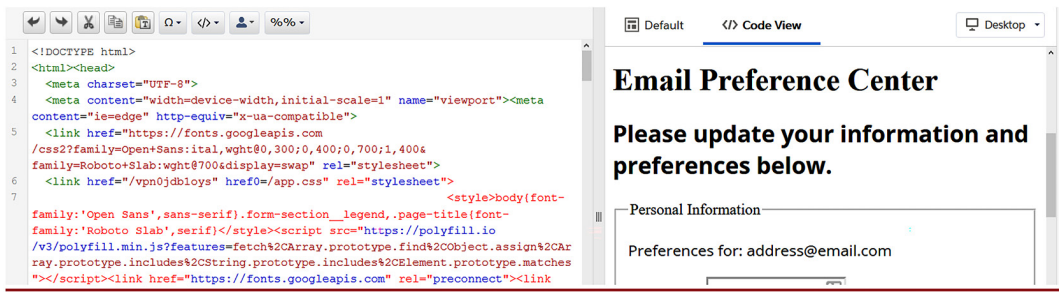

Figure 8.6: Example of options on editing a landing page inside of Cloud Pages

Export Data

Show 20 entries Search:

name	status	ModifiedDate	lastRunDate	NextRun	stepcount	type	lastrunstatus	duration
Geolocation_Claires_NA_Store_Axis_Test	PausedSchedule	12/08/2021 12:35 AM	12/08/2021 12:34 AM		1	scheduled	Complete	1 mins 33 secs
SearchParty	Ready	12/10/2021 12:09 AM	12/10/2021 12:06 AM		1	scheduled	Complete	3 mins 23 secs
CCPA_Delete2	Ready	10/21/2021 05:08 PM	10/21/2021 04:58 PM		7	scheduled	Complete	9 mins 53 secs
gg_DELETEME6	PausedSchedule	11/14/2021 01:30 PM			2	scheduled		
AutoSendReport_FULL_2021	Ready	11/15/2021 10:36 AM	11/15/2021 10:33 AM		2	scheduled	Complete	3 mins 4 secs
DE_Inventory	Ready	11/14/2021 01:30 PM			1	scheduled		
File-Drop Auto	AwaitingTrigger	10/19/2021 08:42 AM			2	triggered		
ImpressionRegions	Ready	01/06/2022 02:25 PM	01/06/2022 02:25 PM		2	scheduled	Complete	21 secs
test_Aero	Ready	11/27/2021 11:21 PM	11/27/2021 11:21 PM		3	scheduled	Complete	29 secs
CCPA_Itemization	Ready	10/21/2021 04:29 PM	10/21/2021 04:21 PM		6	scheduled	Complete	8 mins 0 secs
Combined_NA_Test	Ready	12/08/2021 12:49 AM	12/08/2021 12:49 AM		1	scheduled	Complete	28 secs

Figure 8.7: Example view of the Automation Dashboard web application

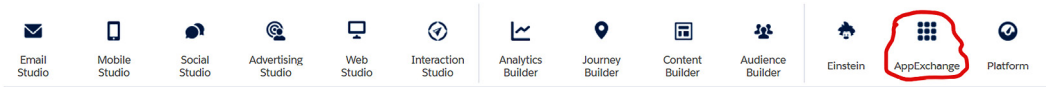

Figure 8.8: Example of where to find the AppExchange option

Code blocks

Code block 8.1

See the following code snippet from the full code hosted on GitHub:

```
    //Security headers to secure the page
    HTTPHeader.SetValue("Access-Control-Allow-Methods","post,
      GET");
    HTTPHeader.SetValue("Access-Control-Allow-Origin","*");
    Platform.Response.SetResponseHeader("Strict-Transport-
      Security","max-age=200");
    Platform.Response.SetResponseHeader("X-XSS-
      Protection","1; mode=block");
    Platform.Response.SetResponseHeader("X-Content-Type-
      Options","nosniff");
    Platform.Response.SetResponseHeader("Referrer-
      Policy","strict-origin-when-cross-origin");
```

```
  //Gather the referrer url
  var referrer = Platform.Request.ReferrerURL;
  //Verifies that this site is being called from within
  //SFMC
  //Pay attention to this as if they change the domain,
  //this will break and toss an error
  if (referrer.indexOf('exacttarget.com') < 0) {
    throw error;
  }
```

Code block 8.2

```
//App information, including REST API info
  var appURL = '{{urlOfCloudPage}}';
  var clientID = '{{clientID}}';
  var clientSecret = '{{clientSecret}}';
  var tenantID = '{{tenantSubDomain}}';
  //Gather state parameter from URL
  var state =
    Platform.Request.GetQueryStringParameter('state');

  //Validate if authorized
  if (state) {
    var code =
      Platform.Request.GetQueryStringParameter('code');
    var payloadObj = {
        grant_type: 'authorization_code',
        code: code,
        client_id: clientID,
        client_secret: clientSecret,
        redirect_uri: appURL
    };
    //Gets Token if already Authorized
    var res = HTTP.Post('https://' + tenantID +
      '.auth.marketingcloudapis.com/v2/token',
      'application/json', Stringify(payloadObj));
    var resJSON =
```

```
      Platform.Function.ParseJSON(res.Response[0]);
    var accessToken = resJSON.access_token;
    var authToken = 'Bearer ' + accessToken;
  } else {
    state = GUID();
    Platform.Response.Redirect('https://' + tenantID +
      '.auth.marketingcloudapis.com/v2/authorize
      ?response_type=code&client_id=' + clientID +
      '&redirect_uri=' + appURL + '&state=' + state);
    //Authorizes request if not already authorized
  }
```

Code block 8.3

```
  //Sets as global var for final JSON variable
  var data = [];
  //set Expire value for auto lookback
  var expire = new Date()
  expire.setMonth(expire.getMonth() - 5)
  //5 months subtraction because month is 0 index, meaning
  //0 is January in getMonth, but setMonth will see
  //Jan as 1.
  var autos = getAutomations(tenantID,authToken);
  var items = autos.entry;
```

Code block 8.4

Following is a snippet, removing the setting of variables and iteration parts, that shows the loop and the appropriate loop settings and usage:

```
  var c = 0;
  while(c < items.length) {
    var a = items[c]
     c++;
  }
  //Stringify for use in client-side JS
  var json = Stringify(data);
```

Code block 8.5

See the following snippet of the code for reference:

```
//Auto Object for pushing into data array - resets to
//null on each loop
autoObj = {};
var start = new Date(a.startTime)
var completed = new Date(a.completedTime)
var delta = Math.abs(completed.getTime() –
  start.getTime()) / 1000
// calculate (and subtract) whole days
var days = Math.floor(delta / 86400);
delta -= days * 86400;
// calculate (and subtract) whole hours
var hours = Math.floor(delta / 3600) % 24;
delta -= hours * 3600;
// calculate (and subtract) whole minutes
var minutes = Math.floor(delta / 60) % 60;
delta -= minutes * 60;
// what's left is seconds
var seconds = Math.floor(delta % 60);
var diff = (hours > 0 ? hours + 'hrs ': '') + (minutes
  > 0 ? minutes + ' mins ' : '') + seconds + ' secs';
```

Code block 8.6

The following snippet shows how we will do that:

```
//Validates if the automation was run in past 6 months
//Or the automation was modified in past 6 months and
//was not run but has schedule type defined.
if ((new Date(a.lastRunTime) > expire) || (new
  Date(a.modifiedDate) > expire && a.lastRunDate ===
  undefined && a.automationType != 'unspecified')) {
 //set each date into a date data type
 var modDate = new Date(a.modifiedDate)
 var lastRunDate = new Date(a.lastRunTime)
 var nextRunDate = new Date(a.scheduledTime)
```

```
      //add each key/value pair into autoObj
      autoObj.AutoID = a.id;
      autoObj.ClientID = a.clientId;
      autoObj.Name = a.name;
      autoObj.Status = a.status;
      autoObj.modifiedDate = formatDate(modDate);
      autoObj.LastRunDate = a.lastRunStatus === undefined?
        '' : formatDate(lastRunDate);
      autoObj.StepCount = a.processes ? a.processes.length :
        0;
      autoObj.Type = a.automationType;
      autoObj.LastRunStatus = a.lastRunStatus;
      autoObj.lastRunDuration = diff != '0 secs' ? diff :
        '';
      autoObj.NextRun = formatDate(nextRunDate);
      //push autoObj into the data array
      data.push(autoObj);
     }
```

Code block 8.7

```
AutoObj = {
  AutoId: a.id
}
```

Code block 8.8

```
GET /legacy/v1/beta/bulk/automations/automation/definition/
Host: {{tenant_subDomain}}.rest.marketingcloudapis.com
Authorization: Bearer {{authToken}}
Content-Type: application/json
```

Code block 8.9

```
  //gets a JSON of all automations inside of Business Unit
  function getAutomations(tenantID,authToken) {
    var url = 'https://' + tenantID +
      '.rest.marketingcloudapis.com/legacy/v1/beta/
      bulk/automations/automation/definition/';
```

```
    var req = new Script.Util.H.T.T.P. request(url);
    req.emptyContentHandling = 0;
    req.retries = 2;
    req.continueOnError = true;
    req.contentType = "application/json"
    req.setHeader("Authorization", authToken);
    req.method = "GET";
    var resp = req.send();
    var resultStr = String(resp.content);
    var resultJSON =
      Platform.Function.ParseJSON(String(resp.content));
    return resultJSON;
  }
```

Code block 8.10

```
  //prepares the date for display in dashboard
  function formatDate(myDate) {
    if(typeof(myDATE) !== null) {
    var month = (myDate.getMonth() + 1),
    day = myDate.getDate(),
    year = myDate.getFullYear(),
    hour = myDate.getHours(),
    mins = myDate.getMinutes(),
    meridiem = '';
    if (hour > 12) {
      hour = Number(hour) - 12;
      meridiem = 'PM'
    } else {
      meridiem = 'AM';
    }
    return ('0' + month).slice(-2)  + '/' + ('0' +
      day).slice(-2)  + '/' + year  + ' ' + ('0'
      hour).slice(-2)   + ':' + ('0' + mins).slice(-2) + '
      ' + meridiem;
    } else { return '';}
  }
```

Code block 8.11

```
<!DOCTYPE html>
<html lang="en">
  <head>
<!-- Required meta tags and external CSS Style Sheets -->
  </head>
  <body>
    <div style="width:90%; padding:20px 0 20px 0; margin:0
      auto;">
      <button id="exportBtn" onclick="download()">
        Export Data</button>
      <table id="autoTable" class="table table-striped
        table-bordered table-hover">
        <thead class="thead-dark">
          <tr>
            <th style="display:none;">AutoID</th>
            <th style="display:none;">Client</th>
            <th scope="col">name</th>
            <th scope="col">status</th>
            <th scope="col">ModifiedDate</th>
            <th scope="col">lastRunDate</th>
            <th scope="col">NextRun</th>
            <th scope="col">stepcount</th>
            <th scope="col">type</th>
            <th scope="col">lastrunstatus</th>
            <th scope="col">duration</th>
          </tr>
        </thead>
      </table>
    </div>
    <-- Script/Library blocks and external calls -->
  </body>
</html>
```

Code block 8.12

```
let currentUrl = document.referrer;
      let autoURL = currentUrl + 'cloud/#app/Automation
        Studio/AutomationStudioFuel3/%23Instance/';
      let myJSON = <ctrl:var name=json />;
```

Code block 8.13

```
      function buildAutoTable() {
        let autoTable = $('#autoTable').DataTable( {
            "data": myJSON,
            "paging": true,
            "pageLength": 20,
            "lengthMenu": [ [10, 20, 50, -1], [10, 20, 50,
              "All"] ],
            "search": { "caseInsensitive": false },
            "columns": [
              {"data": "AutoID",
               "visible":false
              },
              {"data": "ClientID",
               "visible":false
              },
              {"data": "Name",
                "render": function(data, type, row, meta){
                   if(type === 'display'){
                       data = '<a href="' + autoURL +
                         row["AutoID"] + '"
                         target="_blank" >' + data +
                         '</button>';
                   }
                   return data;
                }
              },
              {"data": "Status"},
              {"data": "modifiedDate", "type":
                'date-mm-dd-yyyy'},
              {"data": "LastRunDate"},
```

```
                {"data": "NextRun",
                  "render": function(data, type, row, meta){
                     if(type === 'display'){
                         data = data == '12/31/1969 06:00
                           PM' ? '' : data;
                     }
                     return data;
                  }
                },
                {"data": "StepCount"},
                {"data": "Type"},
                {"data": "LastRunStatus"},
                {"data": "lastRunDuration"}
             ]
    } );
    return autoTable;
}
$(document).ready(function() {
    let autoTable = buildAutoTable();
});
```

Code block 8.14

Take a look at the following snippet to see the function we are referencing:

```
//Export CSV JS
function download() {
    let csv = '';
    let items = myJSON;
    // Loop the array of objects
    for(let i = 0; i < items.length; i++){
        let numKeys = Object.keys(items[i]).length
        let counter = 0
        // If this is the first row, generate the
        // headings
        if(i === 0){
           // Loop each property of the object
           for(let key in items[i]){
```

```
          // This is to not add a comma at the last cell
          // The '\r\n' adds a new line
                    csv += key + (counter+1 < numKeys ? ','
                      : '\r\n' )
                    counter++
                }
             }else{
                for(let key in items[i]){
                    let data = items[i][key];
                    data = data == '12/31/1969 06:00 PM' ?
                      '' : data;
                    csv += data + (counter+1 < numKeys ? ','
                      : '\r\n' )
                    counter++
                }
             }
        }
        // Once we are done looping, download the .csv by
        // creating a link
        let link = document.createElement('a')
        link.id = 'download-csv'
        link.setAttribute('href',
          'data:text/csv;charset=utf-8,' +
           encodeURIComponent(csv));
        link.setAttribute('download',
          'AutoDashExport.csv');
        document.body.appendChild(link)
        document.querySelector('#download-csv').click()
      }
```

Chapter 9

Figures

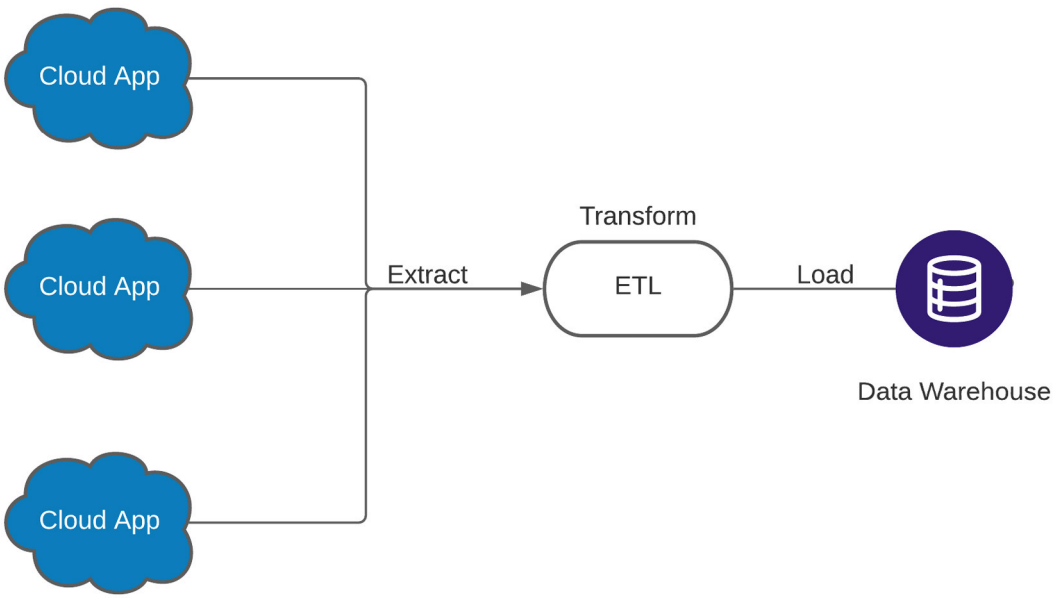

Figure 9.1: The ETL process

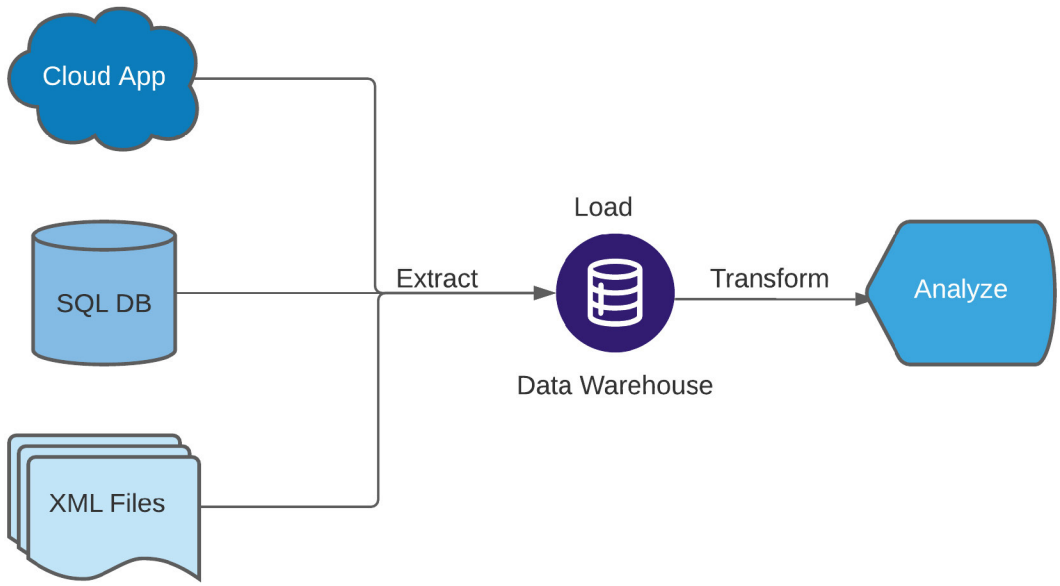

Figure 9.2: The ELT process

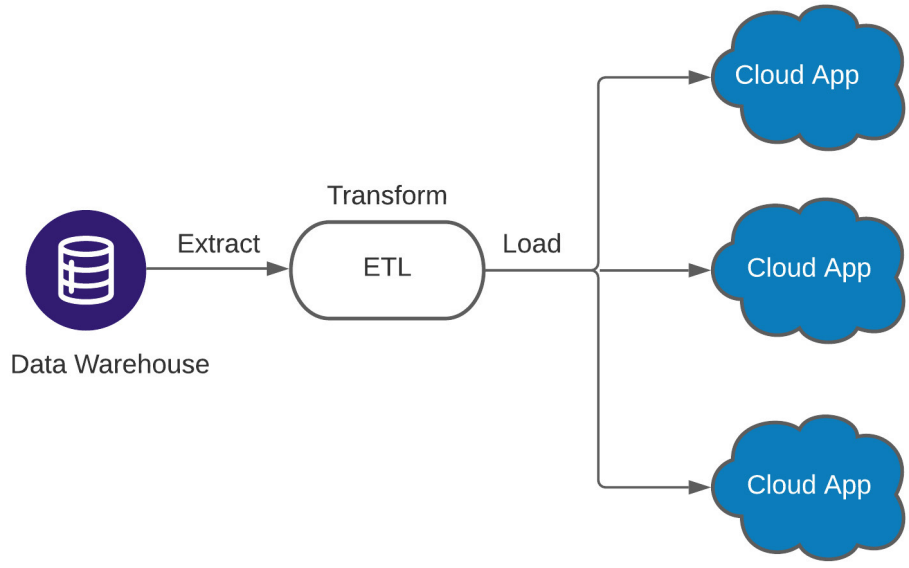

Figure 9.3: The Reverse E.T.L. process

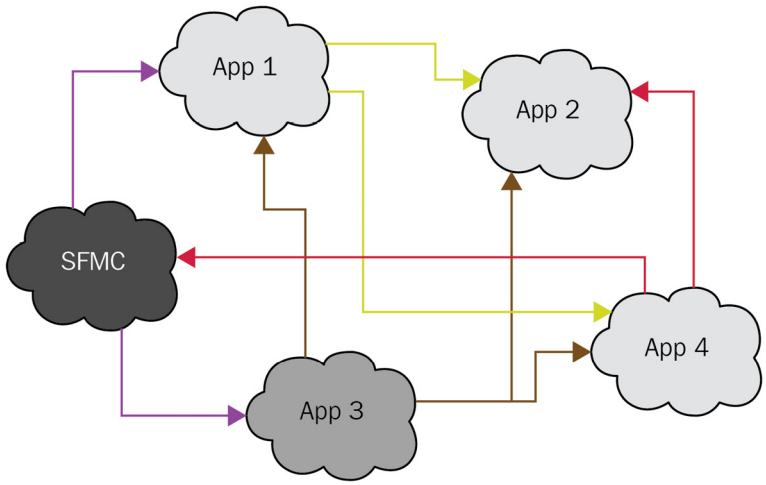

Figure 9.4: A point-to-point integration

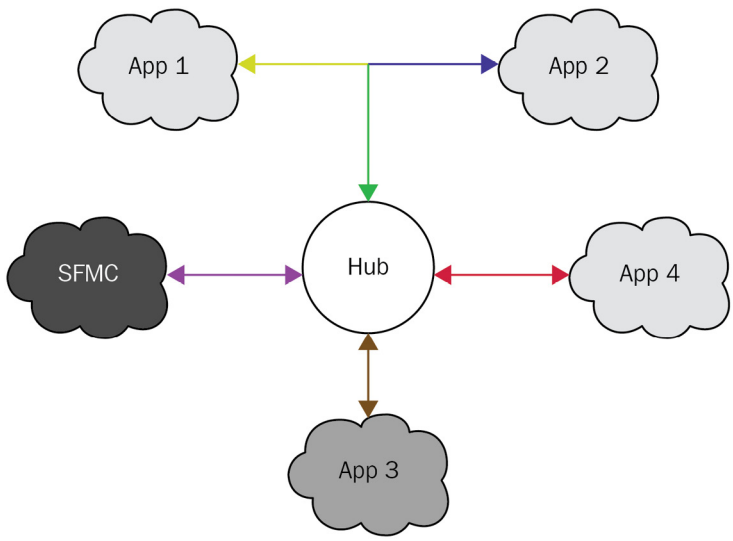

Figure 9.5: The hub-and-spoke methodology

Chapter 10

Tables

PersonContactID	ActivityType	ActivityDate	ActivityResult

Table 10.1: Structure of the data extension

Code blocks

Code block 10.1

```
private | public | global
[virtual | abstract | with sharing | without sharing]
class ClassName [implements InterfaceNameList] [extends
ClassName]
{
// The body of the class
}
```

Code block 10.2

```
    @AuraEnabled
    public static String getMarketingCloudToken(){
        String responseBody = makeJSONPostCall(
            MARKETING_CLOUD_AUTH_URL,
            JSON.serialize( new Map<String, String>{
                ‹grant_type’: ‘client_credentials’
                    ‹clientId’ => ClientID,
                        ‹clientSecret’ => ClientSecret
                            ‹account_id’: YOURMID
                    } ), NULL
        );
```

```
        return ((Map<String, String>) JSON.deserialize(
          responseBody, Map<String, String>.class))
          .get( ACCESS_TOKEN );
    }
    @AuraEnabled
    public static String searchDataExtension(String Email) {
            String authToken = getMarketingCloudToken();
            Http h = new Http();
            HttpRequest webReq = new HttpRequest();
            webReq.setMethod('GET');
            webReq.setHeader('Authorization',
              'Bearer ' + authToken);
            webReq.setEndpoint(searchDEURL);
            HttpResponse res = h.send(webReq);
            String response = res.getbody();
            return response;
    }
```

Code block 10.3

Let's look at some sample component markup for our solution here:

```
<aura:component implements="flexipage:availableForAllPageTypes"
access="global">
  <aura:attribute name="searchId" type="String"
    default="" />
  <aura:attribute name="searchResults" type="String[]" />
```

Code block 10.4

```
<div class="table-search">
  <lightning:input
    type="text"
    name="email-search"
    placeholder="Enter A PersonContactId To Search
      Data Extension"
    value="{!v.searchId}"
    />
  <div class="search-icon" onclick="{!c.searchSubmit}">
```

```
    <lightning:icon
      iconName="utility:search"
      alternativeText="Search"
      size=»small»
      />
  </div>
  <aura:if isTrue="{!not(empty(v.searchResults))}">
    <table>
      <thead>
        <tr>
          <th>PersonContactId</th>
          <th>Activity Type</th>
          <th>ActivityDate</th>
          <th>ActivityResult</th>
        </tr>
      </thead>
      <tbody>
        <aura:iteration items="{!v.searchResults}"
          var=»row»>
          <tr>
            <td>{!row.personcontactid}</td>
            <td>{!row.activitytype}</td>
            <td>{!row.activitydate}</td>
           <td>{!row.activityresult}</td>
         </tr>
        </aura:iteration>
      </tbody>
    </table>
  </aura:if>
</aura:component>
```

Code block 10.5

```
({
    searchSubmit: function (component, event, helper) {
    helper.getSearchResults(component, event, helper);
  }
```

```
});
```

Code block 10.6

```
({
    getSearchResults: function (component, event, helper) {
    var action = component.get("c.searchDataExtension");
    var searchId = component.get("v.searchId");
    action.setParams({ SearchParam: searchId });
    action.setCallback(this, function(response) {
        var temp = response.getReturnValue();
        var json = JSON.parse(temp.toString());
        var primNode = json.items;
        var resultsArr = [];
        for(var i in primNode) {
            var arrVals = primNode[i].values;
            resultsArr.push(arrVals);
        }
        component.set("v.searchResults",resultsArr);
    });
    $A.enqueueAction(action);
  }
});
```

Figures

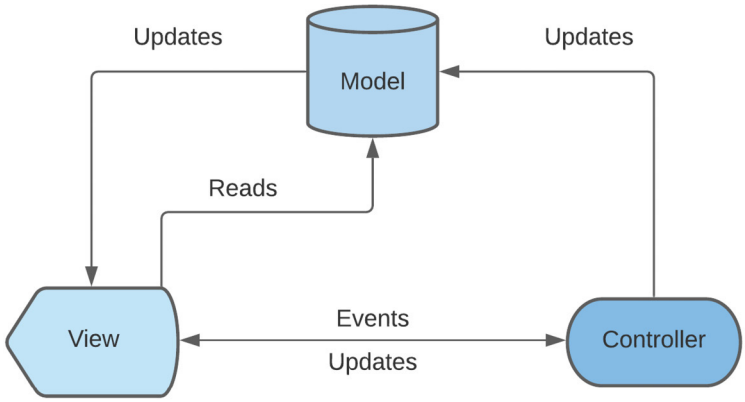

Figure 10.1, M.V.C. architecture

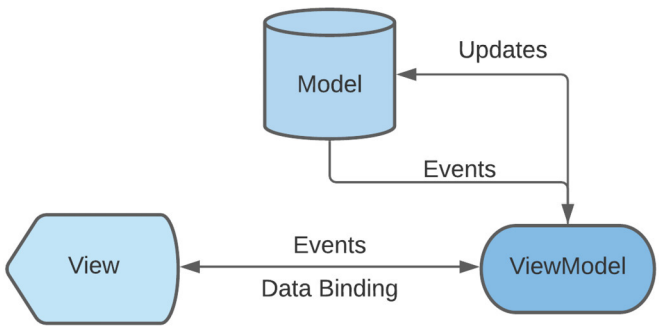

Figure 10.2, M.V.V.M. architecture

Links

- https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/install-packages.html

Chapter 11

Figures

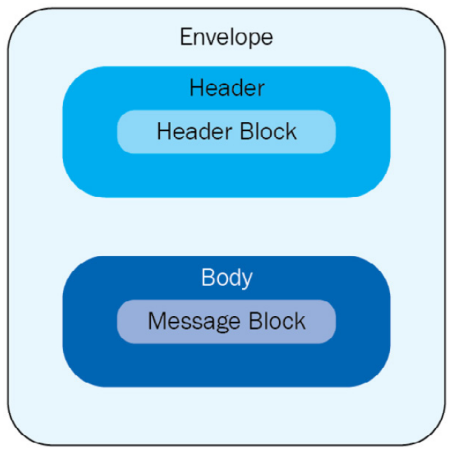

Figure 11.1: Visualization of a soap API envelope

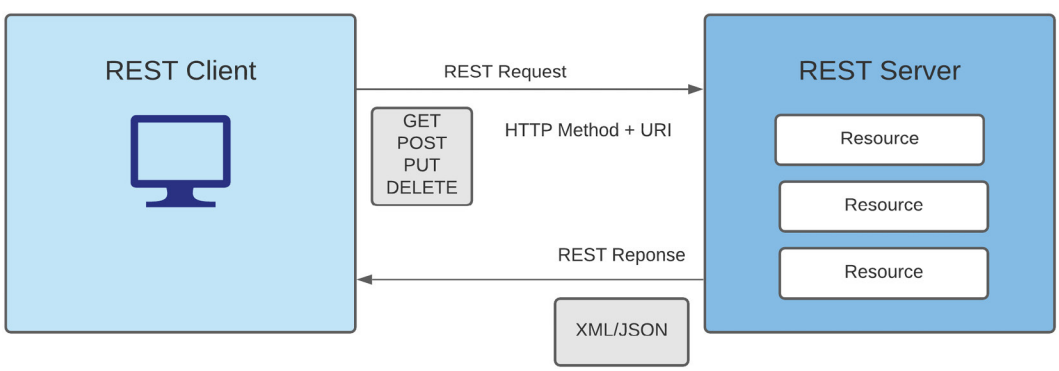

Figure 11.2: Visualization of the request structure of a rest request

Tables

Category	API	SDK
Purpose	Connects and integrates software and services.	Contains a variety of development tools that are used to streamline development.
Characteristics	Lightweight, efficient, and specialized for the purpose at hand.	Includes many utilities and tools that provide more comprehensive functionality beyond singular use cases.
Use Case	Adding specific functionality to an application or service.	Used to create new applications or services that have a multitude of functionalities or requirements.

Table 11.1: API and SDK comparison

Code blocks

Code block 11.1

```
{"compositeRequest" : [{
   "method" : "post",
   "url" : "/services/data/v53.0/sobjects/Account",
   "referenceId" : "myReferencedAccountID",
   "body" : { "Name" : "My cool Account" }
   },{
   "method" : "post",
   "url" : "/services/data/v53.0/sobjects/Contact",
   "referenceId" : "mySuperCoolContactID",
   "body" : {
     "LastName" : "Soup-herman",
     "AccountId" : "@{myReferencedAccountID.id}"
     }
   }]
}
```

Code block 11.2

```
<?xml version = "1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://www.w3.org/2001/12/
soap-envelope" >
   <SOAP-ENV:Header>
      {{your header content here}}
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      {{your body content here}}
      <SOAP-ENV:Fault>
         {{your fault content here}}
      </SOAP-ENV:Fault>   </SOAP-ENV:Body>
</SOAP_ENV:Envelope>
```

Code block 11.3

```
<?xml version="1.0" encoding="UTF-8"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
</s:Envelope>
```

Code block 11.4

```
<s:Envelope xmlns:account="https://example.com/xml/account"
xmlns:lead="https://example.com/xml/lead">
<account:id>Account1</account:id>
<lead:id>Lead1</lead:id>
</s:Envelope>
```

Code block 11.5

```
<s:Header>
<a:Action s:mustUnderstand="1">Create</a:Action>
<a:To s:mustUnderstand="1">{{myTenantSubDomain}}.soap.
marketingcloudapis.com/Service.asmx</a:To>
<fueloauth>{{myAuthToken}}</fueloauth>
</s:Header>
```

Code block 11.6

```
<s:Body
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <CreateRequest
        xmlns="http://exacttarget.com/wsdl/partnerAPI">
        <Options>
            <Client>
                <ClientID>{{MID}}</ClientID>
            </Client>
        </Options>
        <Objects xsi:type="TriggeredSend">
            <Client>
                <ClientID>{{MID}}</ClientID>
            </Client>
            <PartnerKey xsi:nil="true" />
            <ObjectID xsi:nil="true" />
            <TriggeredSendDefinition>
                <PartnerKey xsi:nil="true" />
                <ObjectID xsi:nil="true" />
                <CustomerKey>{{MyTrigger_ExternalKey}}
                  </CustomerKey>
            </TriggeredSendDefinition>
            <Subscribers><SubscriberKey>test@example.com
              </SubscriberKey>
            <EmailAddress>test@example.com
               </EmailAddress>
                <Attributes>
                    <Name>FirstName</Name>
                    <Value>Wiley</Value>
                </Attributes>
            </Subscribers>
        </Objects>
    </CreateRequest>
</s:Body>
```

Code block 11.7

```
        <soap:Fault>
           <faultcode>soap:VersionMismatch</faultcode>
           <faultstring, xml:lang='en">
              Message was not SOAP 1.1 compliant
           </faultstring>
           <faultactor>
              http://sample.org.ocm/jws/authnticator
           </faultactor>
        </soap:Fault>
```

Code block 11.8

```
{
  "prop1": "someVal",
  "prop2": "anotherVal"
}
```

Code block 11.9

```
[
  { "op": "replace", "path": "/prop1",
    "value": "newProp1Val" },
  { "op": "add", "path": "/newProp",
    "value": ["hello world"] },
  { "op": "remove", "path": "/prop2" }
]
```

Code block 11.10

```
{
  "prop1": "newProp1Val",
  "newProp": ["hello world"]
}
```

Code block 11.11

```
POST /asset/v1/content/assets/query HTTP/1.1
Host: YOURENDPOINT}}.rest.marketingcloudapis.com
Authorization: Bearer TOKEN
{
    "page":{
        "page":1,
        "pageSize":50
    },
    "query":{
        "leftOperand":{
            "property":"createdDate",
            "simpleOperator":"greaterThan",
            "value":"2021-07-04"
        },
        "logicalOperator":"AND",
        "rightOperand":{
            "property":"assetType.name",
            "simpleOperator":"equal",
            "value":"png"
        }
    }
}
```

Code block 11.12

```
// GET /person/1
{
    "name":"John Doe",
    "lastlogindate": "07/04/2021"
}
```

Code block 11.13

```
type Person {
  id: ID
  name: String
  lastlogindate: Date
  employer: Employer
}
type Employer {
  id: ID
  name: String
  role: String
  manager: String
  person: Person
}
```

Code block 11.14

```
type Query {
  person(id: ID!): Person
  employer(id: ID!): Employer
}
```

Code block 11.15

```
// GET /graphql?query={ person(id: "1") { name, employer {
manager } } }
{
   "name":"John Doe",
   "employer":{
      "manager":"Jane Doe"
   }
}
```

Code block 11.16

```
def getSDKToken()
  authClient = MarketingCloudSDK::Client.new({'client' =>
    {'id' => CLIENTID,'secret' => CLIENTSECRET}})
  return authClient
end
```

Code block 11.17

```
def getRestToken()
  uri =
    URI('https://auth.exacttargetapis.com/v1/requestToken')
  Net::HTTP.start(uri.host, uri.port, :use_ssl =>
    uri.scheme == 'https') do |http|
  req = Net::HTTP::Post.new(uri)
  req['Content-Type'] = 'application/json'
  req.set_form_data('clientId' => CLIENTID, 'clientSecret'
    => CLIENTSECRET)
  response = http.request req # Net::HTTPResponse object
  responseObj = JSON.parse(response.body)
  token = responseObj["accessToken"]
  return token
  end
end
```

Code block 11.18

```
et_client = Savon.client(
    wsdl: wsdl,
    endpoint: endpoint,
    wsse_auth: [username, password],
    raise_errors: false,
    log: false,
    open_timeout:180,
    read_timeout: 180
)
```

Code block 11.19

```
rqst = {}
rqst['ObjectType'] = 'Send'
rqst['Properties'] = ['ID',
'EmailName','SendDate','NumberSent',
'UniqueClicks','UniqueOpens','Unsubscribes','HardBounces',
'Subject']
filter = {'@xsi:type' => 'tns:SimpleFilterPart'}
filter['Property'] = 'SendDate'
filter['SimpleOperator'] = 'greaterThan'
filter['Value'] = '2020-12-17'
rqst['Filter'] = filter
```

Code block 11.20

```
rqstmsg = {'RetrieveRequest' => rqst}
response = et_client.call(:retrieve, :message => rqstmsg)
if !response.nil? then
    envelope = response.hash[:envelope]
    retrieveresponse =
      envelope[:body][:retrieve_response_msg]
        if retrieveresponse[:overall_status] == "OK"
        p 'Success'
        results = retrieveresponse[:results]
        if !results.kind_of?(Array)
            results = [results]
        end
        @send_results = results.to_json
        #results.each {|list| p "ListID: #{list[:id]}
        #{list[:email_name]} #{list[:send_date]}
        #{list[:number_sent]} #{list[:unique_opens]}
        #{list[:unique_clicks]}"}
    end
```

Chapter 12

Links

- Documentation outlining the possible routes and functionality within Content Builder: https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/content-api.html.
- For a complete list of asset type IDs, please refer to the Content Builder Asset Type documentation: https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/base-asset-types.html

Figures

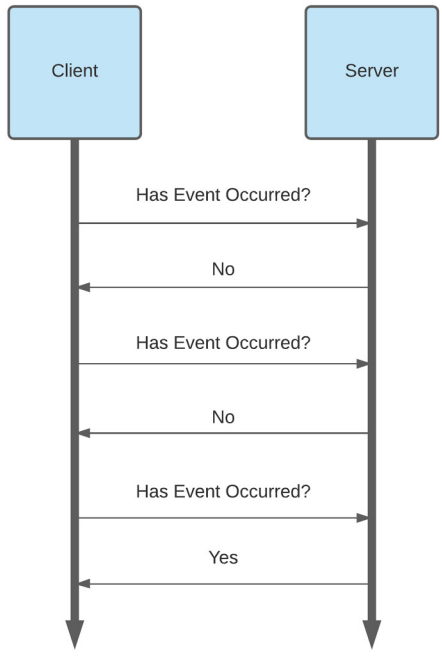

Figure 12.1: Continuous polling visualization

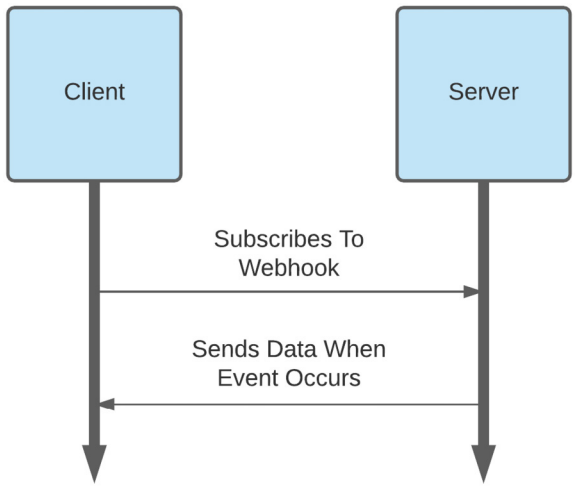

Figure 12.2: Webhook visualization

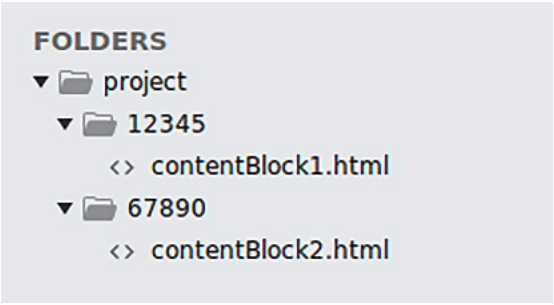

Figure 12.3: Webhook example project structure

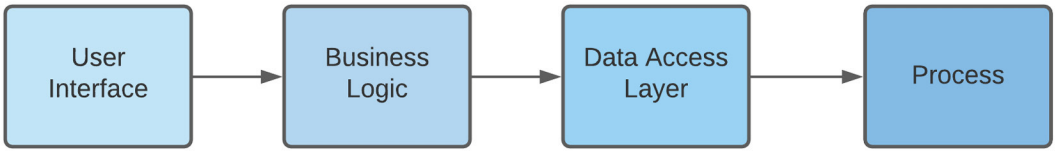

Figure 12.4: Monolithic architecture diagram

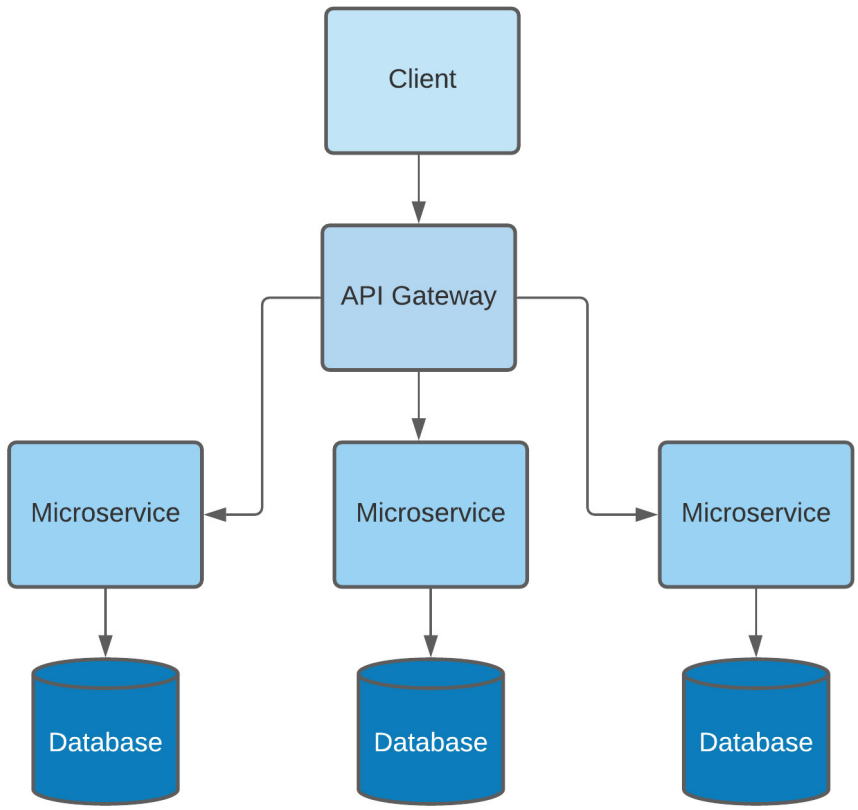

Figure 12.5: Microservice architecture diagram

Code blocks

Code block 12.1

```
{
    "repository":{
       "contents_url":"https://api.github.com/repositorys/
         {username}/{repositoryname}/contents/{+path}"
    },
    "commits":[
       {
          "added":[
             "12345/contentBlock1.html",
             "67890/contentBlock2.html"
```

```
          ],
        }
    ]
}
```

Code block 12.2

```
<script runat=server>
Platform.Load("core", "1.1.1");
var postData = Platform.Request.GetPostData();
var json = Platform.Function.ParseJSON(postData);
</script>
```

Code block 12.3

```
var baseContentsURL = json.repository.contents_url;
baseContentsURL = baseContentsURL.slice(0, baseContentsURL.
lastIndexOf('/') + 1);
var addedFilesInCommit = json.commits[0].added;
```

Code block 12.4

Let's take a look at what that script looks like and then break down its components a little further:

```
function getRawGithubData(assetPath, contentURL) {
    var accessToke = "YOUR GITHUB ACCESS TOKEN";
    var auth = 'token ' + accessToken;
    var url = contentURL + assetPath;
    var req = new Script.Util.HttpRequest(url);
    req.emptyContentHandling = 0;
    req.retries = 2;
    req.continueOnError = true;
    req.contentType = "application/json"
    req.setHeader("Authorization", auth);
    req.setHeader("user-agent", "marketing-cloud");
    req.setHeader("Accept",
      "application/vnd.github.VERSION.raw");
    req.method = "get";
    var resp = req.send();
```

```
    var resultString = String(resp.content);
    return resultString;
}
```

Code block 12.5

```
function createAsset(assetName, assetContent, assetId,
assetCategoryId) {
    var asset = Platform.Function.CreateObject("Asset");
    var nameIdReference =
      Platform.Function.CreateObject("nameIdReference");
    Platform.Function.SetObjectProperty(nameIdReference,
      "Id", assetId);
    Platform.Function.SetObjectProperty(asset, "AssetType",
      nameIdReference);
    var categoryNameIdReference = Platform.Function
      .CreateObject("categoryNameIdReference");
    Platform.Function.SetObjectProperty(
      categoryNameIdReference, "Id", assetCategoryId);
    Platform.Function.SetObjectProperty(asset, "Category",
      categoryNameIdReference);
    Platform.Function.SetObjectProperty(asset, "Name",
      assetName);
    Platform.Function.SetObjectProperty(asset, "Content",
      assetContent);
    Platform.Function.SetObjectProperty(asset,
      "ContentType", "application/json");
    var statusAndRequest = [0, 0];
    var response = Platform.Function.InvokeCreate(asset,
      statusAndRequest, null);
    return response;
}
```

Code block 12.6

```
for (var i in addedFilesInCommit) {
    var assetPath = addedFilesInCommit[i];
    var categoryId = assetPath.substring(0,
```

```
        assetPath.indexOf("/"));
    var contentName =
        assetPath.split("/").pop().replace(".html", "");
    var contentData = getRawGithubData(assetPath,
        baseContentsURL);
    createAsset(contentName, contentData, 220, categoryId);
}
```

Chapter 13

Links

- https://developer.salesforce.com/docs/marketing/marketing-cloud/guide/how-data-binding-works.html

Figures

Figure 13.1: The custom activity flow

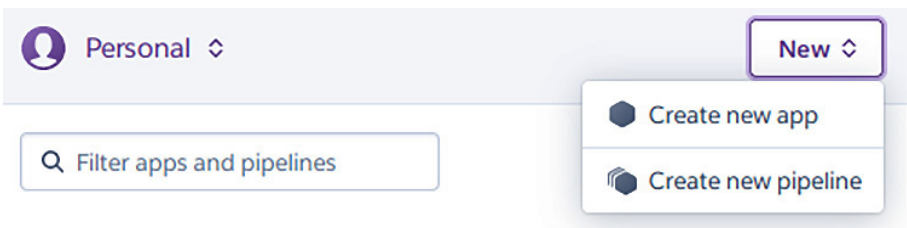

Figure 13.2: The Heroku app creation menu

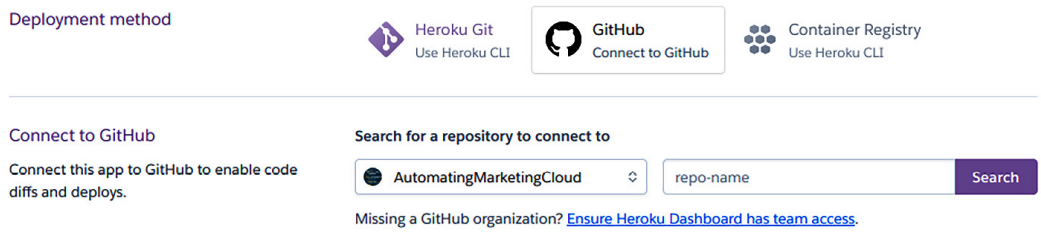

Figure 13.3: The GitHub connection menu

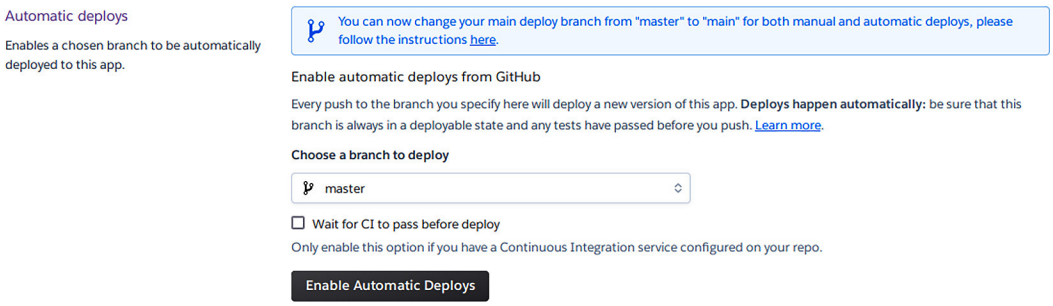

Figure 13.4: The automated deployment menu

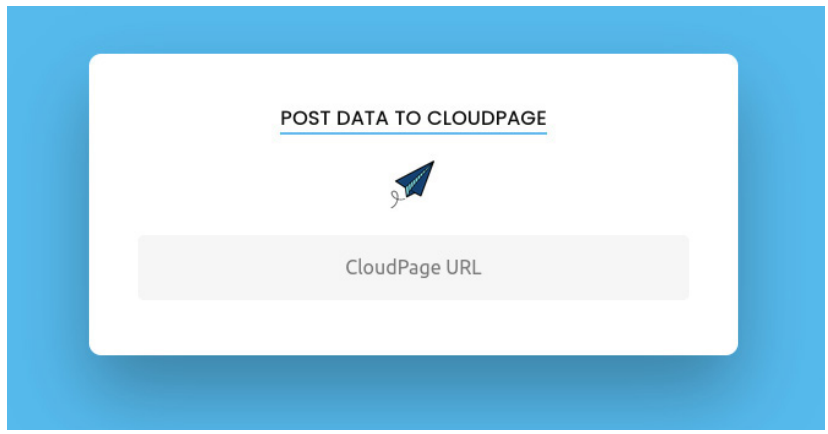

Figure 13.5: The application UI

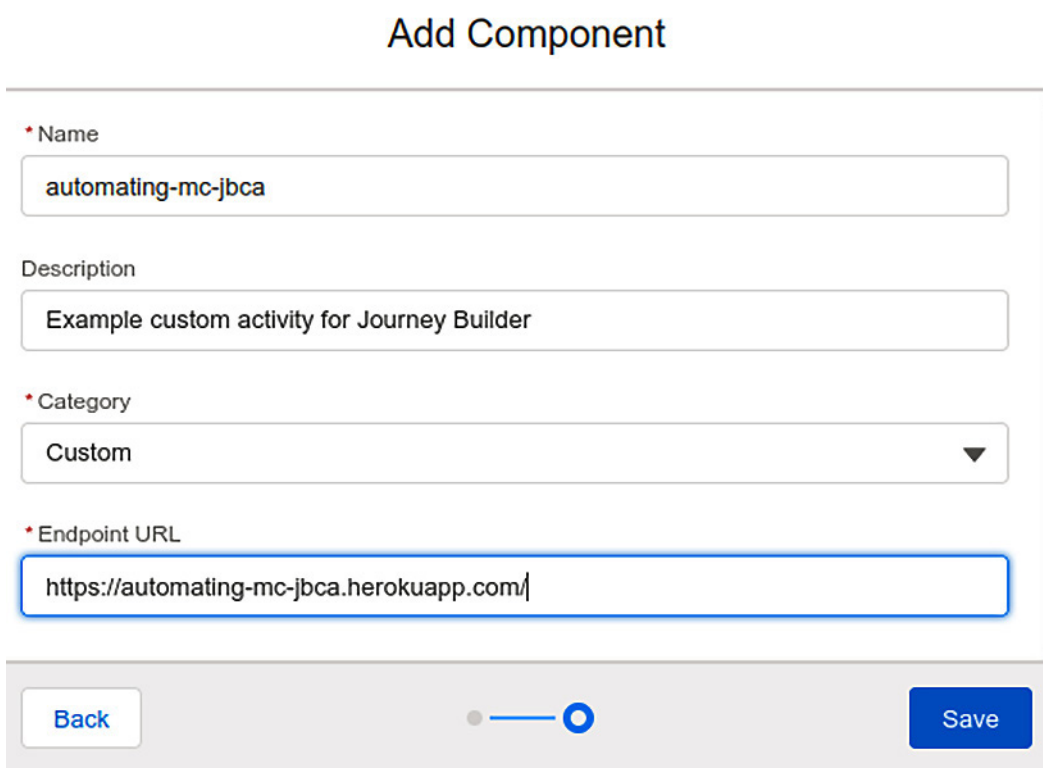

Figure 13.6: The custom activity configuration menu

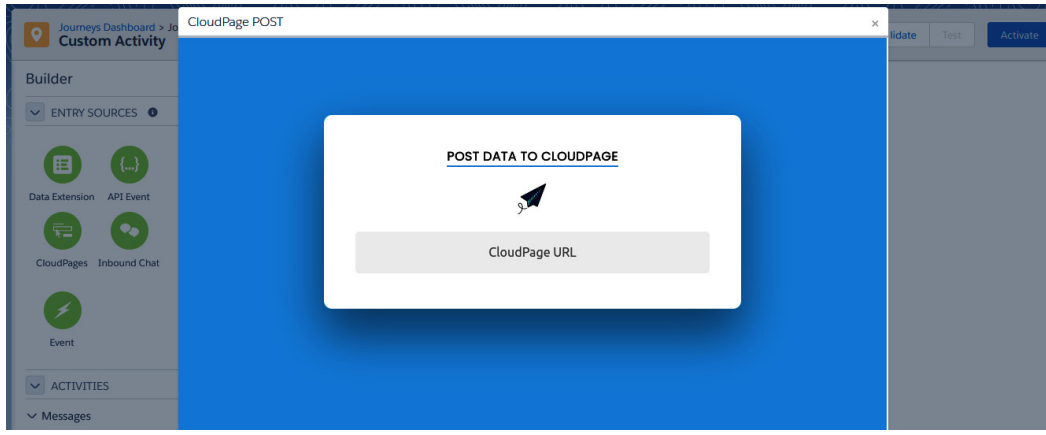

Figure 13.7: A custom activity in Journey Builder

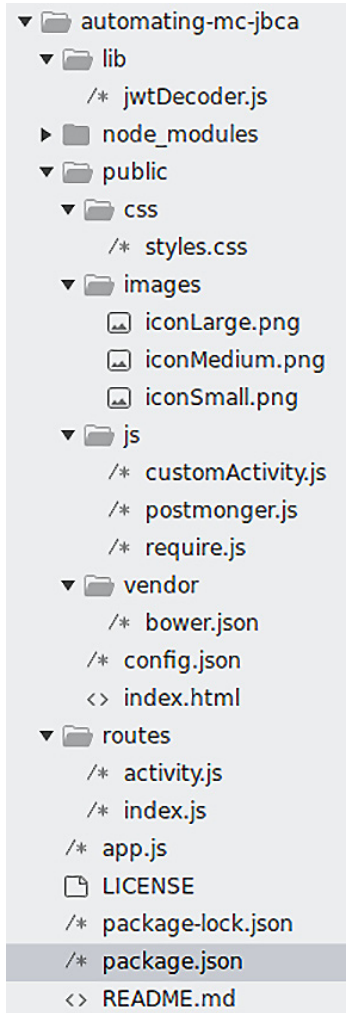

Figure 13.8: The custom activity project structure

Figure 13.9: Heroku Config Vars

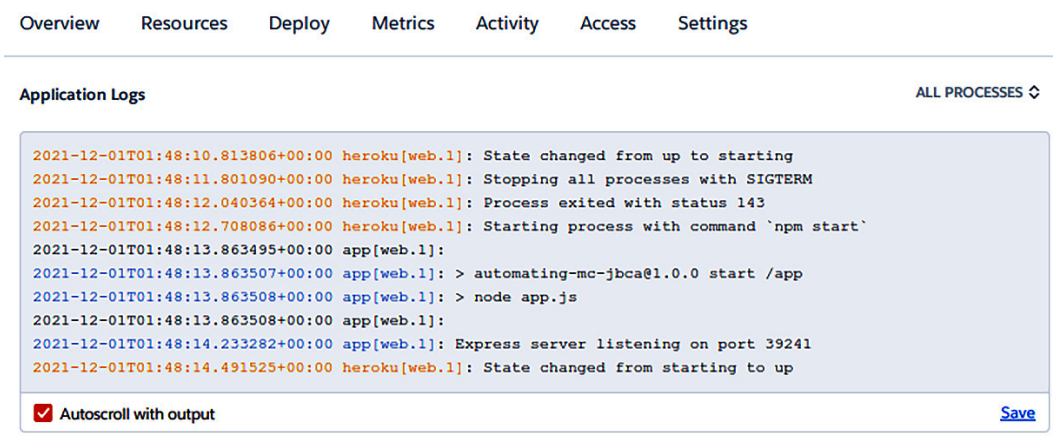

Figure 13.10: The Heroku logs

Code blocks

Code block 13.1

```
<script runat=server>
Platform.Load("Core","1");
var postData = Platform.Request.GetPostData();
Platform.Function.
InsertData("CustomActivityTest",["activityData"],[postData]);
</script>
```

Code block 13.2

```
    "workflowApiVersion": "1.1",
    "metaData": {
        "icon": "images/iconMedium.png",
        "iconSmall": "images/iconSmall.png",
        "category": "custom"
    },
    "type": "REST",
    "lang": {
        "en-US": {
         "name": "CloudPage POST",
```

```
            "description":
              "Sample Journey Builder Custom Activity",
            "step1Label": "Configure Activity"
          }
      },
```

Code block 13.3

```
      "arguments": {
          "execute": {
             "inArguments":[
                  {"subscriberKey":"{{Contact.Key}}"}
              ],
            "outArguments": [],
            "url": "https://automating-mc-jbca.herokuapp.com/
              journeybuilder/execute",
             "verb": "POST",
              "body": "",
              "header": "",
              "format": "json",
              "useJwt": true,
              "timeout": 10000,
              "retryCount": 5,
              "retryDelay": 100
          }
      },
```

Code block 13.4

```
      "wizardSteps": [
          { "label": "Configure Activity", "key": "step1" }
      ],
      "userInterfaces": {
          "configModal": {
              "height": 640,
              "width": 900,
            "fullscreen": false
          }
```

```
    },
```

Code block 13.5

```
<script type="text/javascript"
      src="js/require.js"></script>
    <script type="text/javascript">
        (function() {
            var config = {
                baseUrl: 'js'
            };
            var dependencies = [
                'customActivity'
            ];
            require(config, dependencies);
        })();
    </script>
```

Code block 13.6:

```
<input type="text" id="cpURL" class="fadeIn second"
name="cpURL" placeholder="CloudPage URL">
```

Code block 13.7

```
    var connection = new Postmonger.Session();
    var payload = {};
    $(window).ready(onRender);
    connection.on('initActivity', initialize);
    connection.on('clickedNext', save);
```

Code block 13.8

```
unction initialize(data) {
        if (data) {
            payload = data;
            var setcpURL = payload['arguments']
              .execute.inArguments[0].cloudpageURL;
            $('#cpURL').val(setcpURL);
        }
```

```
    }
```

Code block 13.9

```
    function save() {
        var cpURL = $('#cpURL').val();
        payload['arguments'].execute.inArguments = [{
            "subscriberKey": "{{Contact.Key}}",
            "cloudpageURL": cpURL
        }];
        payload['metaData'].isConfigured = true;
        connection.trigger('updateActivity', payload);
    }
```

Code block 13.10

```
const JWT = require(Path.join(__dirname, '..', 'lib',
'jwtDecoder.js'));
```

Code block 13.11

```
exports.route = function(req, res) {
  logData(req);
  res.send(200, 'someRoute');
};
```

Code block 13.12

```
exports.execute = function(req, res) {
  J.W.T.(req.body, process.env.jwtSecret, (err, decoded)
    => {
    if (err) {
      return res.status(401).end();
    } else {
```

Code block 13.4

```
    if (decoded && decoded.inArguments &&
      decoded.inArguments.length > 0) {
      var decodedArgs = decoded.inArguments[0];
      var cpURL = decodedArgs.cloudpageURL;
```

```
        var subKey = decodedArgs.subscriberKey;
    else {
        console.error('inArguments invalid.');
        return res.status(400).end();
     }
```

Code block 13.5

```
        var cpPostBody = JSON.stringify({
            "activityData": {
              "SubscriberKey": subKey,
              "CloudPageURL": cpURL
            }
        });
        request.post({
          headers: {
            "Content-Type": "application/json"
          },
          url: cpURL,
          body: cpPostBody
        }, function(error, response, body) {
            if (error) {
              console.log(error);
          };
        });
        logData(req);
        res.send(200, 'Execute');
```

Chapter 14

Links

- `https://gortonington.com/alert-for-your-triggered-send-queue-limits-in-sfmc/`

Figures

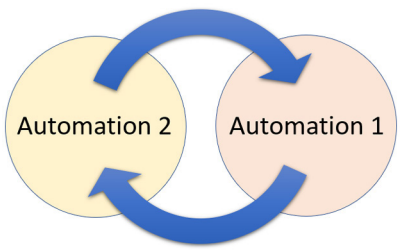

Figure 14.1: Visualization of a continuous automation

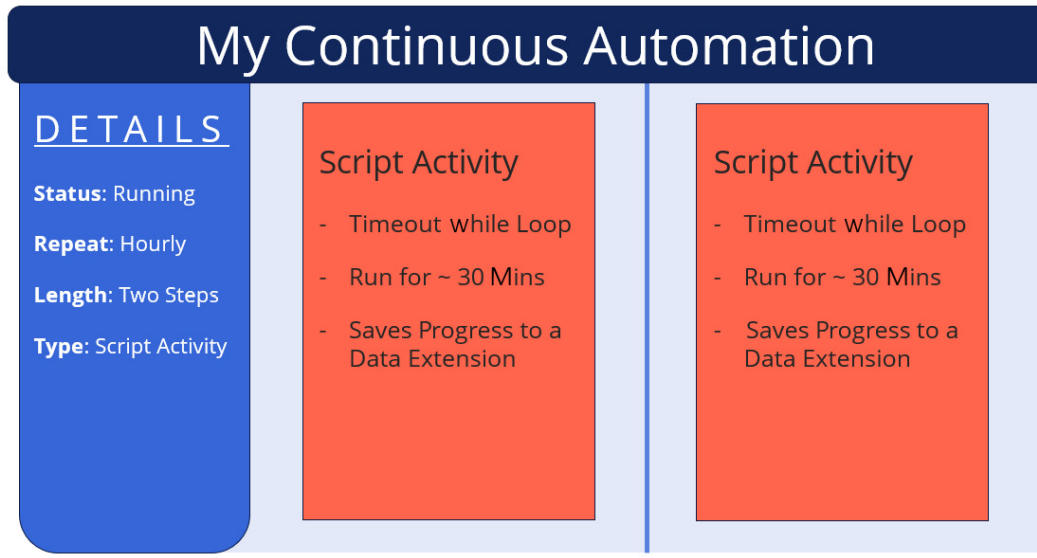

Figure 14.2: Visualization of continuous automation with Script activities

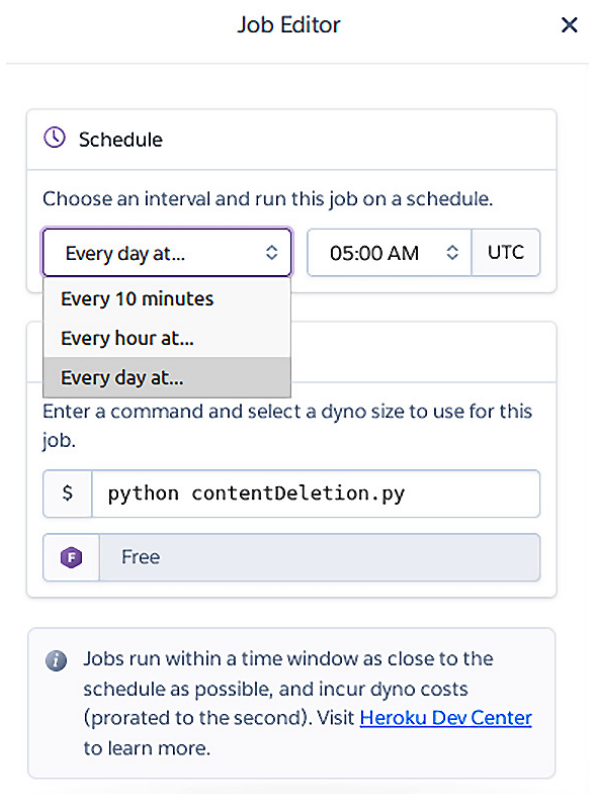

Figure 14.3, Heroku Scheduler configuration modal

Code blocks

Code block 14.1

Here is a quick example script using WSProxy to start another automation:

```
<script runat=server>
    Platform.Load("Core","1.1.1");
    var prox= new Script.Util.WSProxy();
    var objId = {{autoObjectID}}
    var props = {
            ObjectID: objId
     };
    var opts = {};
    var request = prox.performItem("Automation", props,
```

```
      'start', opts);
</script>
```

Code block 14.2

Here's the next sample call to get an automation object ID:

```
<script runat=server>
    Platform.Load("Core","1.1.1");
    var prox = new Script.Util.WSProxy();
    var name = "My Automation";
    var request = prox.retrieve("Automation",
      ["ProgramID"], {
        Property: "Name",
        SimpleOperator: "equals",
        Value: name
    });
    var objId = request.Results[0].ObjectID;
</script>
```

Code block 14.3

Triggering send queue checkup:

```
var date = new Date(),
startTime = now.getTime(),
timeOut = 1620000; //27 minutes
//60000 milliseconds in a minute
do {
  //your JS code
} while((new Date().getTime() - startTime) < timeOut)
```