Understanding outbound web services in ServiceNow

Posted on 2 August 2016

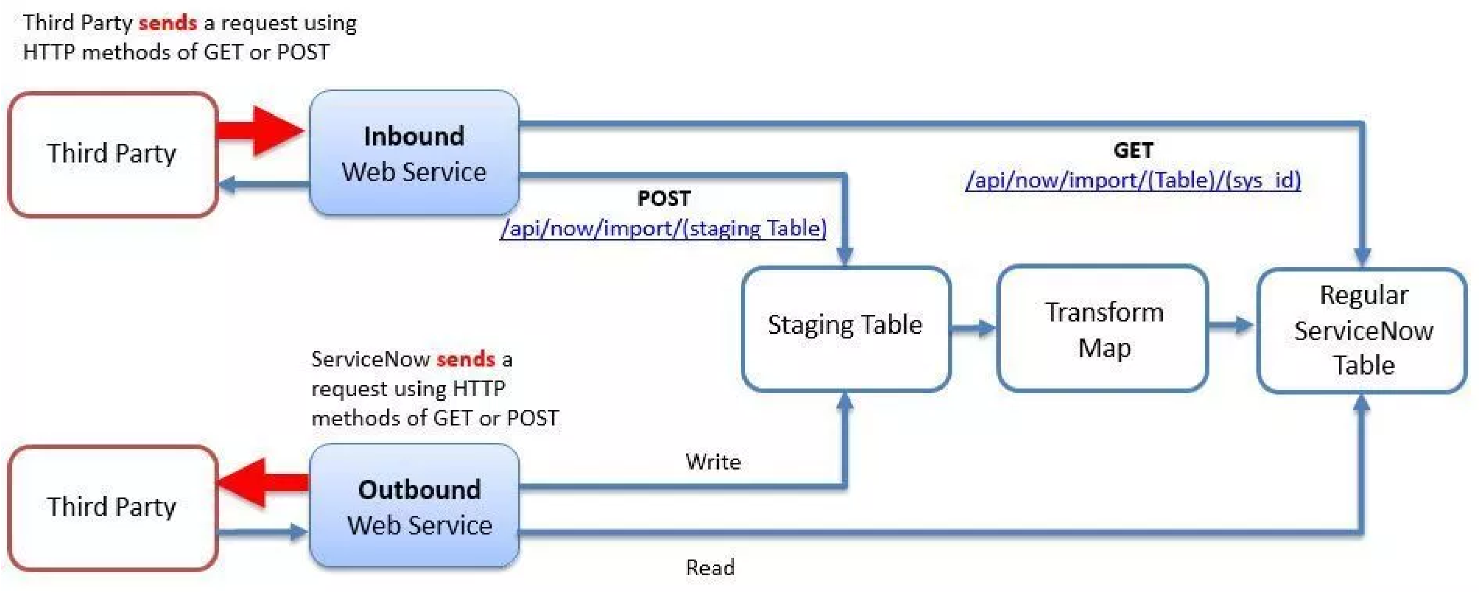
This blog was updated on 20 March 2018

ServiceNow has the ability to manage both inbound and outbound web services, but as they're handled slightly differently, they can be confusing.

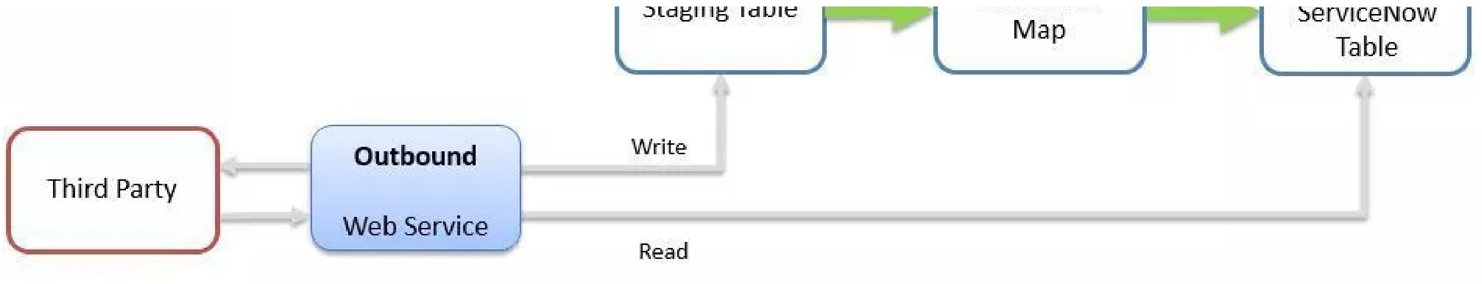
The term "inbound" and "outbound" does not describe the type of HTTP method being used but rather the point of origin for the web service. For example, both inbound and outbound services can use GET and POST.

Inbound web services are designed to provide third parties with the ability to retrieve (GET) or update (POST) data in ServiceNow, while outbound web services allow ServiceNow to initiate a transaction with a third party (also using either GET or POST, etc.)

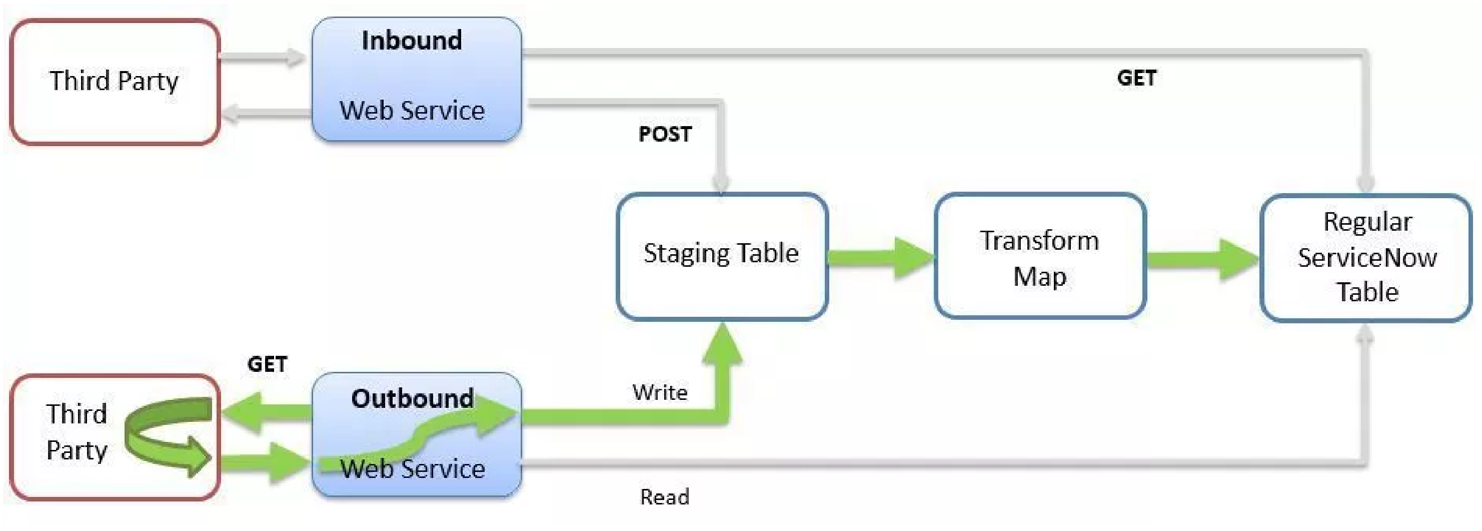
Although they are closely related, inbound and outbound web services are handled differently through the ServiceNow GUI even though they're essentially the same under the covers. Most developers are comfortable with inbound web services because ServiceNow provides a walk-through wizard for them, but they struggle with outbound services, even though these can be handled in the same way.


Third Party sends a request using HTTP methods of GET or POST
Third Party
Inbound Web Service
GET
/api/now/import/(Table)/(sys_id)
POST
/api/now/import/(staging Table)
Staging Table
Transform Map
Regular ServiceNow Table
ServiceNow sends a request using HTTP methods of GET or POST
Third Party
Outbound Web Service
Write
Read

If a third-party wants to send information to ServiceNow using web services, then an inbound web service will allow them to POST that information.



If ServiceNow wants to retrieve information from a third-party using web services, then an outbound web service will allow ServiceNow to GET that information.



In both scenarios, the end result is the same—ServiceNow is populated with information from a third-party.

This tutorial assumes you have a basic knowledge of how to create applications in ServiceNow as we're going to make an outbound web service to retrieve a weather report for use in ServiceNow.

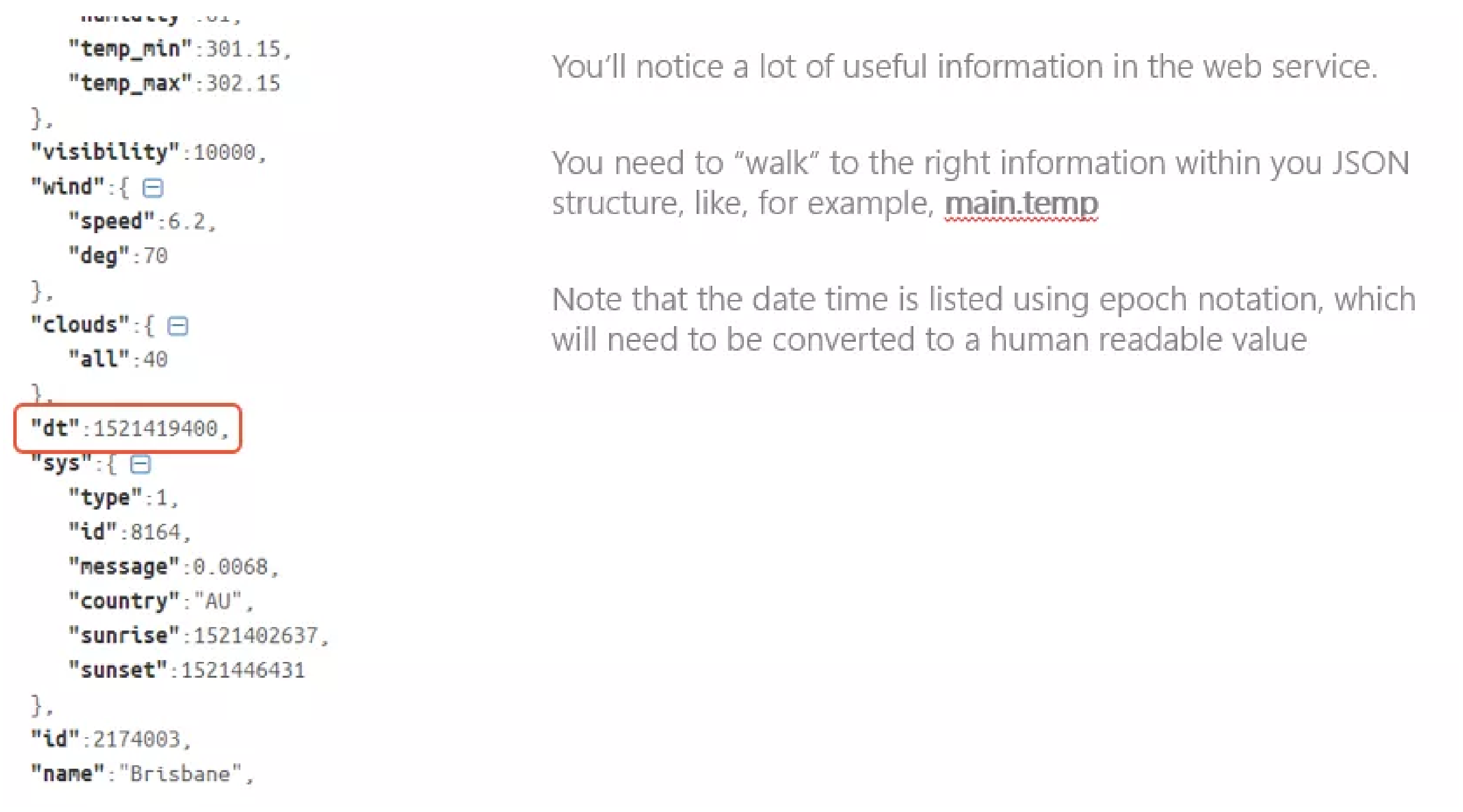
To complete this tutorial, you will need to sign up for a free account with http://openweathermap.org/api and subscribe to the **Current weather data**.

Step One: Create an outbound web service

The external web service made available by Open Weather is:

```
http://api.openweathermap.org/data/2.5/weather?q=Brisbane,au&appid={yourAppID}
```

If you sign up for an account with Open Weather and view the results in a browser, you'll get a wall of JSON data in response, but if you copy and paste this into Notepad++ using the JSON formatter plugin, you'll see there's structure to



```
    "temp_min":301.15,
    "temp_max":302.15
},
"visibility":10000,
"wind":{
    "speed":6.2,
    "deg":70
},
"clouds":{
    "all":40
},
"dt":1521419400,
"sys":{
    "type":1,
    "id":8164,
    "message":0.0068,
    "country":"AU",
    "sunrise":1521402637,
    "sunset":1521446431
},
"id":2174003,
"name":"Brisbane",
```

You'll notice a lot of useful information in the web service.

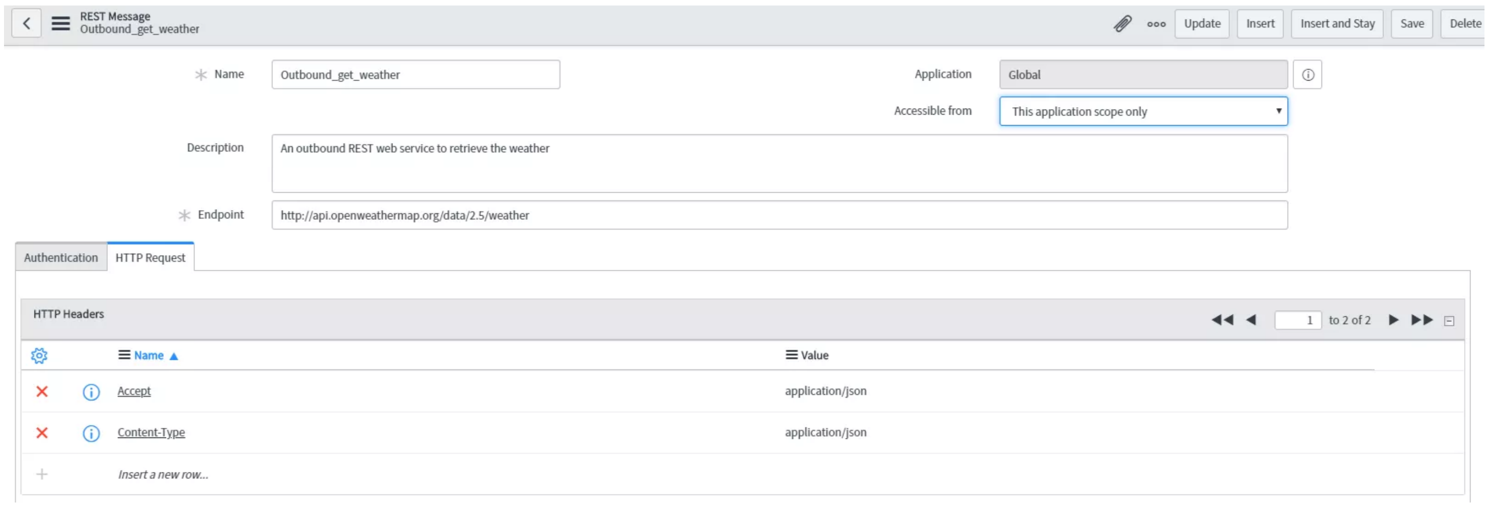
You need to "walk" to the right information within you JSON structure, like, for example, main.temp

Note that the date time is listed using epoch notation, which will need to be converted to a human readable value

As you can see, the temperature is in Kelvin rather than either Celsius or Fahrenheit, as 301K relates to a balmy 28C or 82F.

Also, the date time (dt) is using a numeric format known as Epoch time, so we're going to need to transform the results from our outbound web service before they're going to be useful.

Creating an outbound web service is easy enough, simply provide the web service with a name like **outbound_get_weather** and the end point URL, and tell ServiceNow what to accept and what the content type will be.



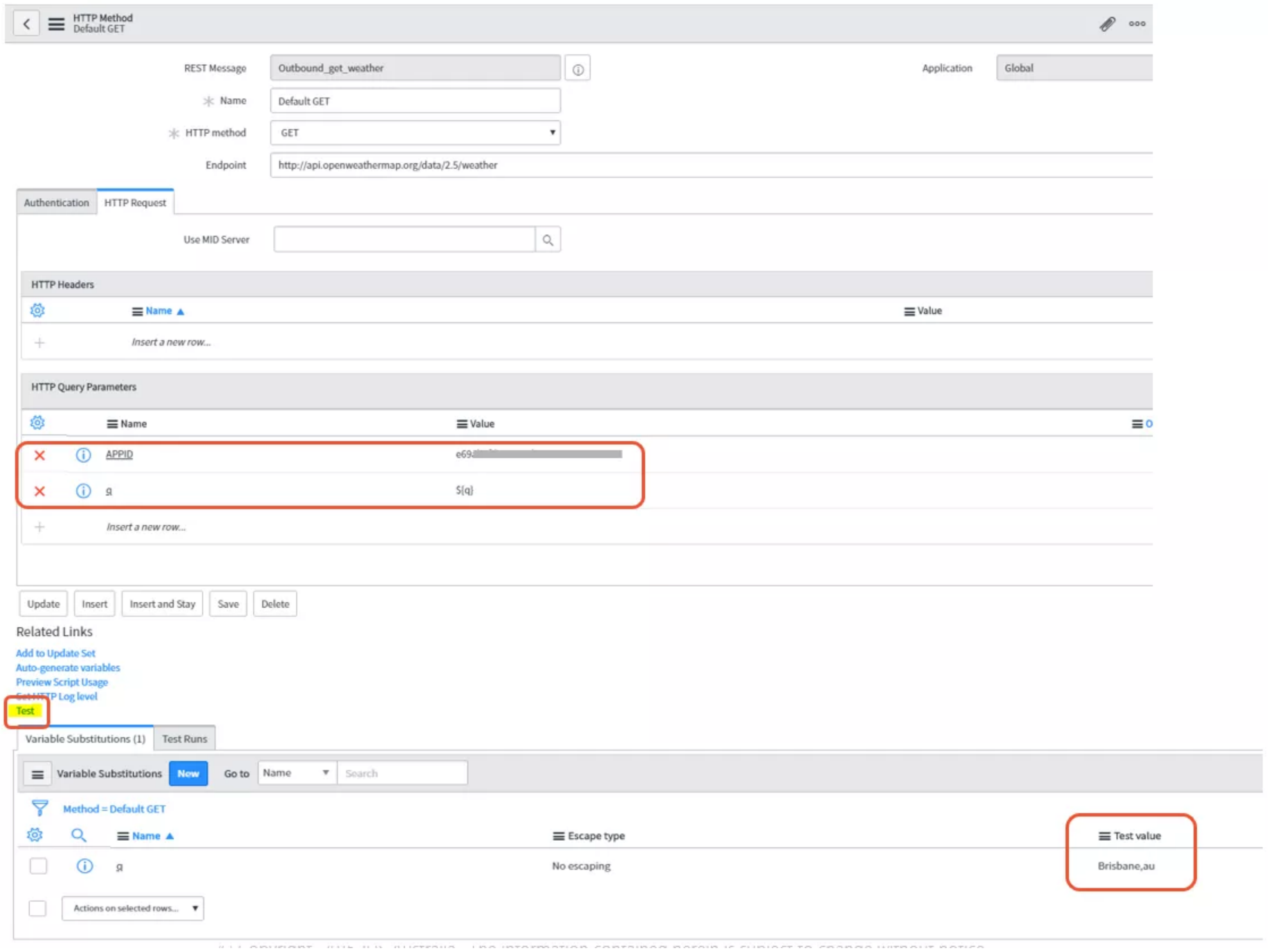
REST Message
Outbound_get_weather

Name	Outbound_get_weather
Application	Global
Accessible from	This application scope only
Description	An outbound REST web service to retrieve the weather
Endpoint	http://api.openweathermap.org/data/2.5/weather

Name ▲	Value
Accept	application/json
Content-Type	application/json
Insert a new row...	

- q is the query parameter for the city, so we're going to set this to be a variable rather than a static value. Make this **${q}**
- **APPID** is a static value, so this can be hard coded to match the application ID you got from Open Weather

Also, notice I've added a **Variable Substitution** and set this to be **Sydney,AU**, which allows me to click **test** and see fresh results for Sydney.



REST Message	Outbound_get_weather
Application	Global
Name	Default GET
HTTP method	GET
Endpoint	http://api.openweathermap.org/data/2.5/weather

HTTP Headers

Name ▲	Value
Insert a new row...	

HTTP Query Parameters

Name	Value
APPID	e69[redacted]
q	${q}
Insert a new row...	

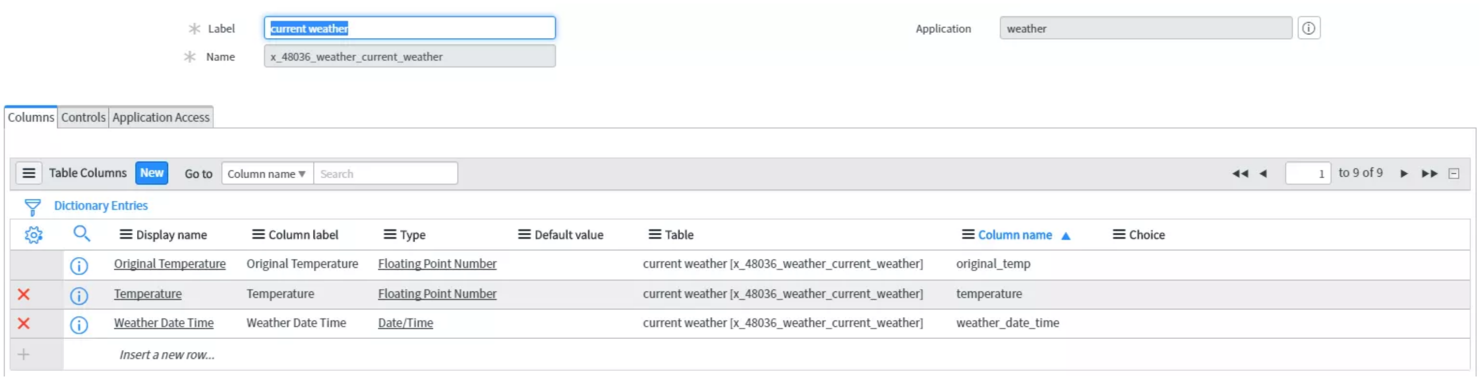
Related Links
Add to Update Set
Auto-generate variables
Preview Script Usage
Set HTTP Log level
Test

Method = Default GET

Name ▲	Escape type	Test value
q	No escaping	Brisbane,au

This is important as when we get to our scripted REST web service, we want to be able to change **q** to refer to other cities as appropriate.

Click **Test** to confirm you've got everything working correctly.



Label	current weather
Name	x_48036_weather_current_weather
Application	weather

Columns | Controls | Application Access

Table Columns New Go to Column name ▾ Search 1 to 9 of 9

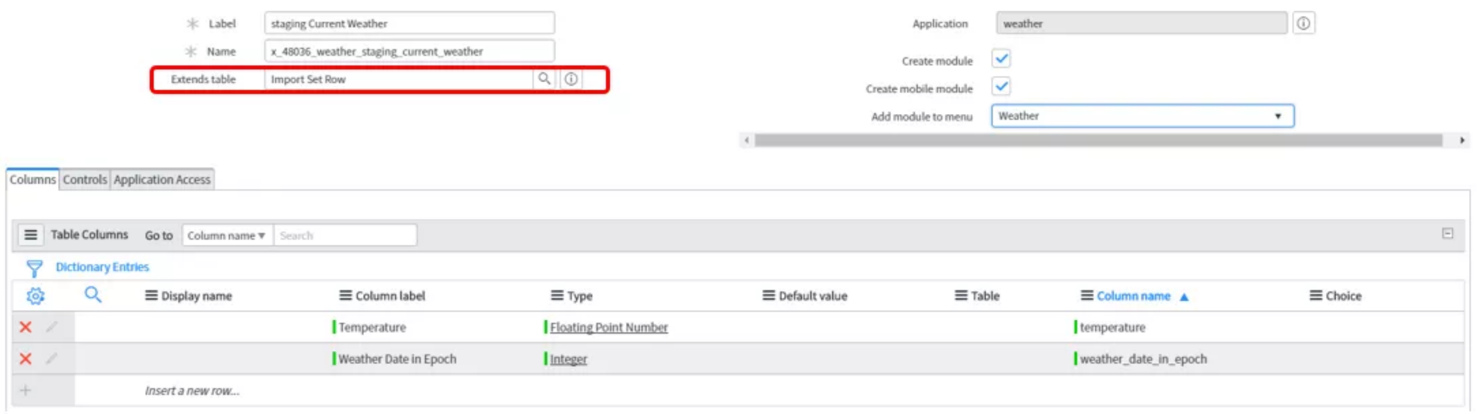
Dictionary Entries

	Display name	Column label	Type	Default value	Table	Column name ▲	Choice
	Original Temperature	Original Temperature	Floating Point Number		current weather [x_48036_weather_current_weather]	original_temp	
✕	Temperature	Temperature	Floating Point Number		current weather [x_48036_weather_current_weather]	temperature	
✕	Weather Date Time	Weather Date Time	Date/Time		current weather [x_48036_weather_current_weather]	weather_date_time	
+	Insert a new row...						

Our second table is a staging table. This is where the raw results of our web service will reside.

When you make an inbound web service, ServiceNow automatically creates a staging table, but when it comes to outbound web services, you need to create a staging table for yourself and manually assign a transform map.

Be careful. You must extend the **Import Set Row** table to create a valid staging transform table for your web service.



Label	staging Current Weather
Name	x_48036_weather_staging_current_weather
Extends table	Import Set Row
Application	weather
Create module	✓
Create mobile module	✓
Add module to menu	Weather

Columns | Controls | Application Access

Table Columns Go to Column name ▾ Search

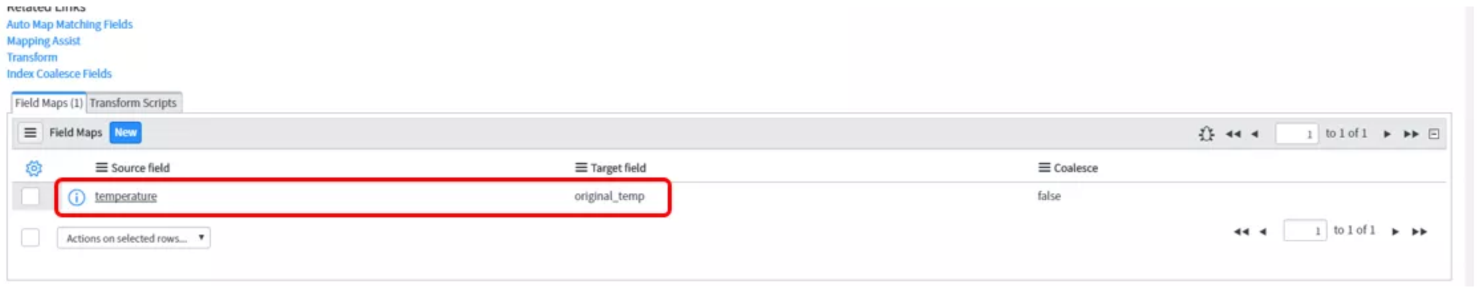
Dictionary Entries

	Display name	Column label	Type	Default value	Table	Column name ▲	Choice
✕		Temperature	Floating Point Number			temperature	
✕		Weather Date in Epoch	Integer			weather_date_in_epoch	
+	Insert a new row...						

Notice how our data types in the staging table match those we saw in the web service results, so the epoch date is being stored as an integer.

Step Three: Transform

Now we can set up the transformation from our staging table to the final table in ServiceNow. Under **System Import Sets** click on **Create Transform Map** and build the following transform.



Related Links
Auto Map Matching Fields
Mapping Assist
Transform
Index Coalesce Fields

Field Maps (1) | Transform Scripts

Field Maps New — 1 to 1 of 1

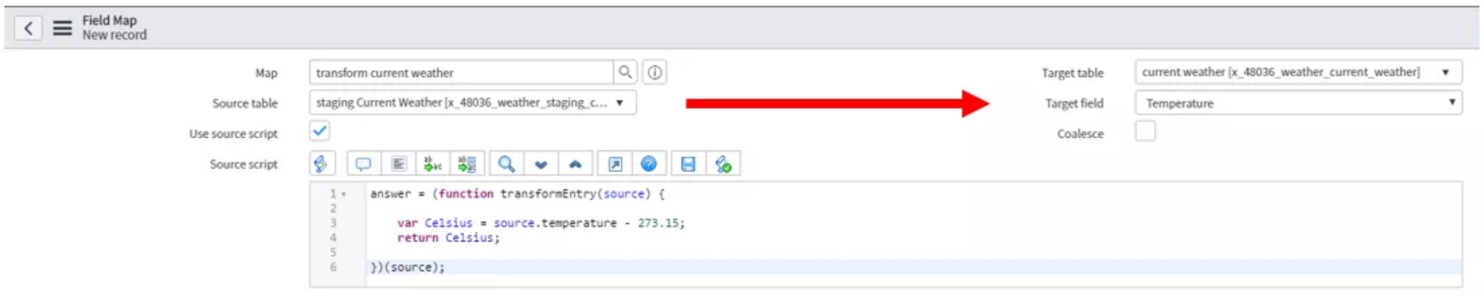
Source field	Target field	Coalesce
temperature	original_temp	false

Actions on selected rows...

Initially, there's only one source field we can match against the target field, for the others, we'll need to use transform scripts.

To convert Kelvin to Celsius, add a new Field Map as follows, using this script:

```
var Celsius = source.temperature - 273.15;
return Celsius;
```



Field Map
New record

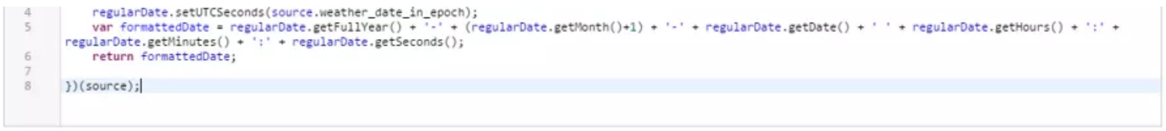
Map	transform current weather	Target table	current weather [x_48036_weather_current_weather]
Source table	staging Current Weather [x_48036_weather_staging_c...	Target field	Temperature
Use source script	✓	Coalesce	

Source script

```
answer = (function transformEntry(source) {

    var Celsius = source.temperature - 273.15;
    return Celsius;

})(source);
```

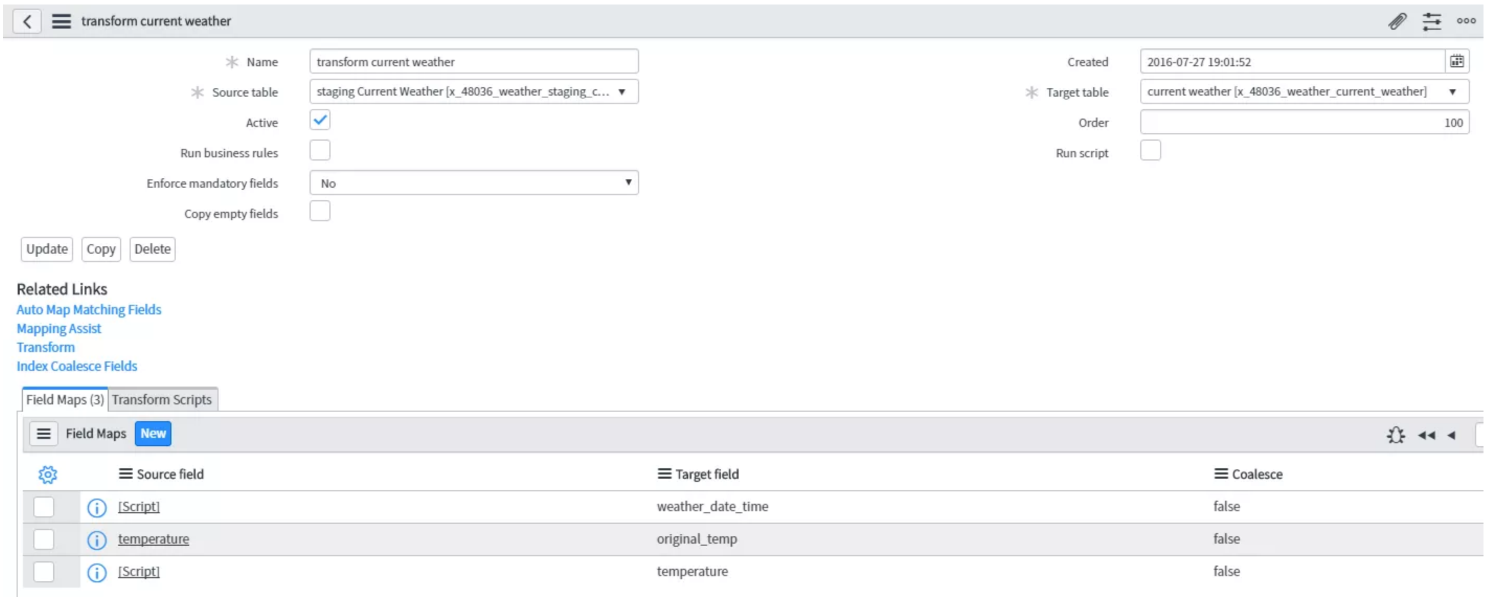
To convert the epoch date to a date ServiceNow will recognise, add another field map using:

```
var regularDate = new Date(0); // 0 forces javascript to use an epoch date

regularDate.setUTCSeconds(source.weather_date_in_epoch);

var formattedDate = regularDate.getFullYear() + '-' + (regularDate.getMonth()+1) + '-' + regularDate.getDate() + ' ' + regularDate.getHours() + ':' + regularDate.getMinutes() + ':' + regularDate.getSeconds();

return formattedDate;
```



```
    regularDate.setUTCSeconds(source.weather_date_in_epoch);
    var formattedDate = regularDate.getFullYear() + '-' + (regularDate.getMonth()+1) + '-' + regularDate.getDate() + ' ' + regularDate.getHours() + ':' +
regularDate.getMinutes() + ':' + regularDate.getSeconds();
    return formattedDate;

})(source);
```

When you're finished, your transform map should appear as:



transform current weather

Name	transform current weather	Created	2016-07-27 19:01:52
Source table	staging Current Weather [x_48036_weather_staging_c...	Target table	current weather [x_48036_weather_current_weather]
Active	✓	Order	100
Run business rules		Run script	
Enforce mandatory fields	No		
Copy empty fields			

Update Copy Delete

Related Links
Auto Map Matching Fields
Mapping Assist
Transform
Index Coalesce Fields

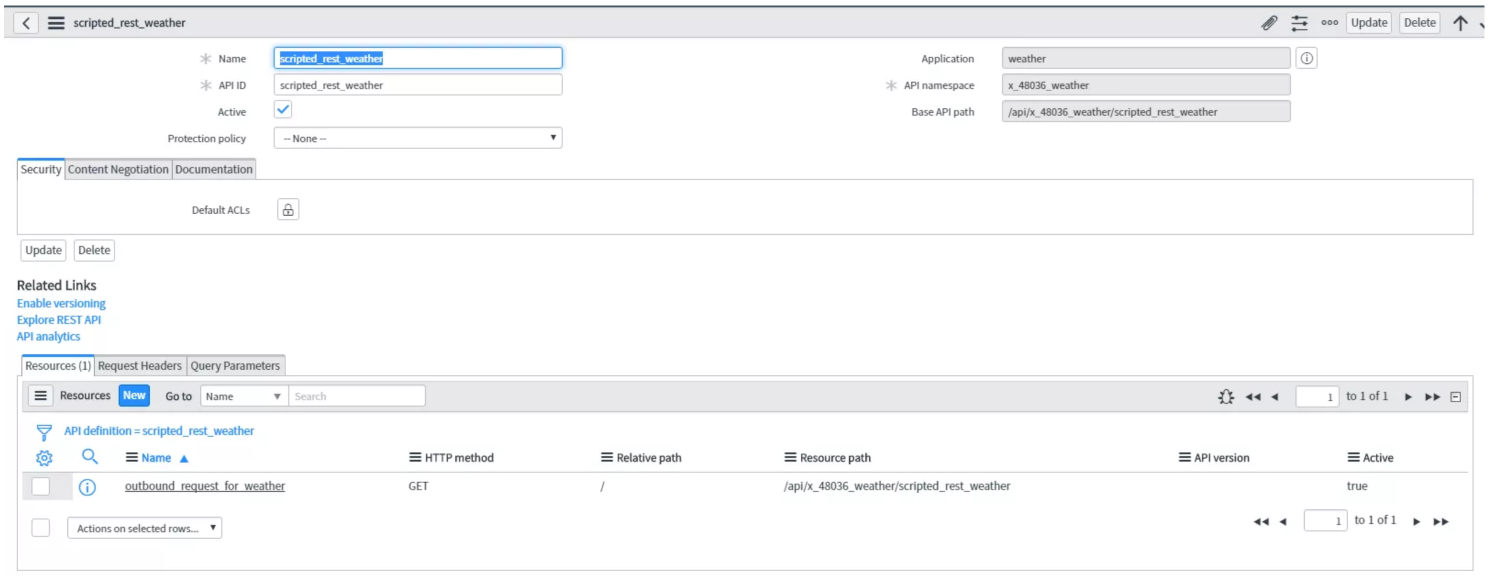
Field Maps (3) | Transform Scripts

Field Maps New

Source field	Target field	Coalesce
[Script]	weather_date_time	false
temperature	original_temp	false
[Script]	temperature	false

Step Four: Tying it all together

We're going to use a scripted REST API to bring everything together, so under **Scripted REST APIs** create the following record.



scripted_rest_weather

Name	scripted_rest_weather	Application	weather
API ID	scripted_rest_weather	API namespace	x_48036_weather
Active	✓	Base API path	/api/x_48036_weather/scripted_rest_weather
Protection policy	-- None --		

Security | Content Negotiation | Documentation

Default ACLs

Update Delete

Related Links
Enable versioning
Explore REST API
API analytics

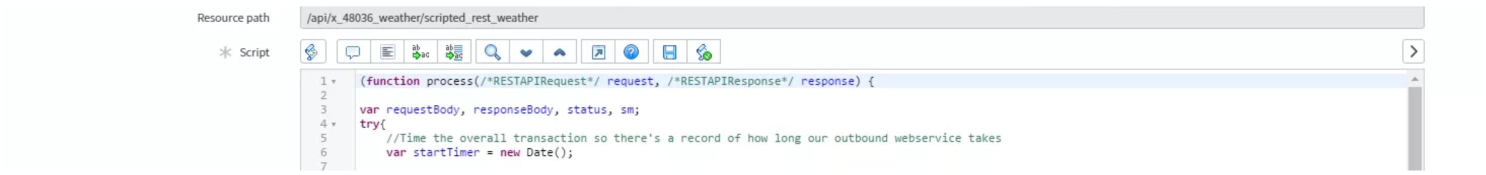
Resources (1) | Request Headers | Query Parameters

Resources New Go to Name Search

API definition = scripted_rest_weather

Name	HTTP method	Relative path	Resource path	API version	Active
outbound_request_for_weather	GET	/	/api/x_48036_weather/scripted_rest_weather		true

Actions on selected rows...



Resource path /api/x_48036_weather/scripted_rest_weather

Script

```
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {

var requestBody, responseBody, status, sm;
try{
    //Time the overall transaction so there's a record of how long our outbound webservice takes
    var startTimer = new Date();
```

The logic of our outbound request for the weather is…

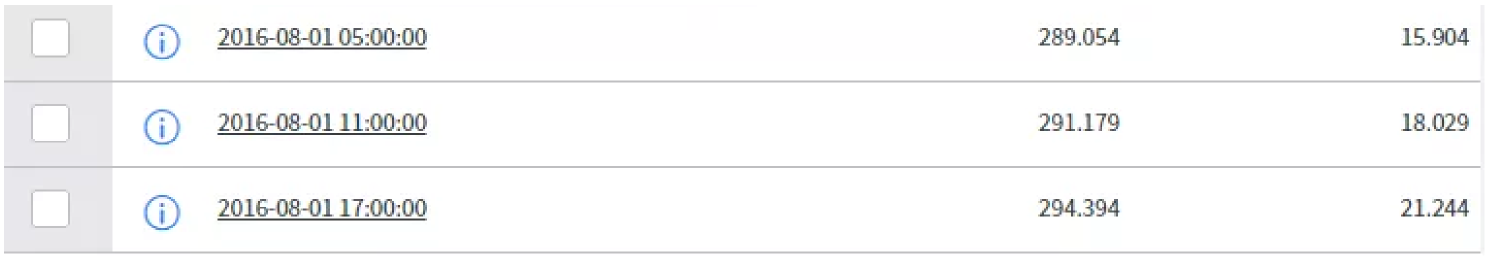
- Time the transaction and record that in the application log
- Use the ServiceNow RESTMessageV2 library to undertake the web service
- Delete any old records
- Set a custom parameter to grab the weather for a specific location
- Retrieve the results from the response body and add these to the staging table
- Our transform map will then run automatically in the background and populate the actual table with results

Here’s the script I used.

```
(function process(/*RESTAPIRequest*/ request, /*RESTAPIResponse*/ response) {

var requestBody, responseBody, status, sm;

try{

                //Time the overall transaction so there's a record of how long our outbound webservice takes

                var startTimer = new Date();


                sm = new sn_ws.RESTMessageV2("outbound_get_weather", "default get");

                //sm.setBasicAuth("admin","admin");

                sm.setStringParameter("q", "Townsville,AU");

                sm.setHttpTimeout(10000) //In milliseconds. Wait at most 10 seconds for response from http request.


                var getresponse = sm.execute();

                 var responseBody = getresponse.getBody();
                var gr = new GlideRecord("x_48036_weather_staging_current_weather");
                gr.query();
                gr.deleteMultiple();

                //Convert the response into JSON
                var JSONdata = new global.JSON().decode(responseBody);

                gr.setValue("weather_date_in_epoch",JSONdata.dt);
                gr.setValue("temperature",JSONdata.main.temp );
                gr.insert();

                //Post a message to the application log noting how long this activity took
                var endTimer = new Date() - startTimer;
                gs.info('***Weather updated in ' + endTimer + ' msec');
} catch(ex) {
                responseBody = "An error has occurred";
                status = '500';
}
})(request, response);
```

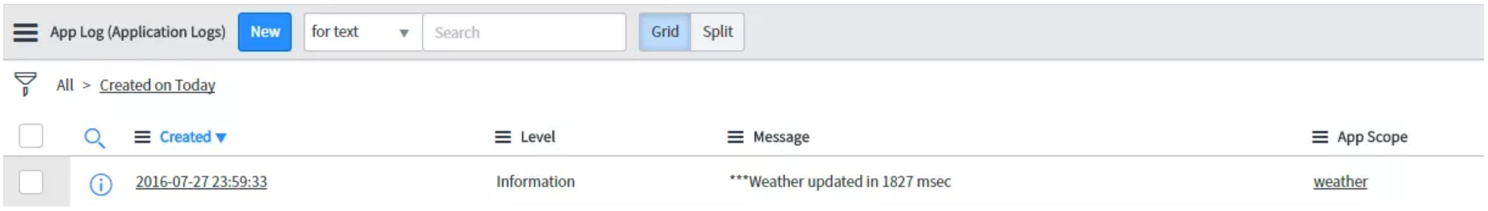
Notice how the **JSONdata.main.temp** matches the structure of web service results we saw earlier.

If we look at the **current weather** table, we can see our results.



	2016-08-01 05:00:00	289.054	15.904
	2016-08-01 11:00:00	291.179	18.029
	2016-08-01 17:00:00	294.394	21.244

If we look at the **application log** we can see how long this activity took.



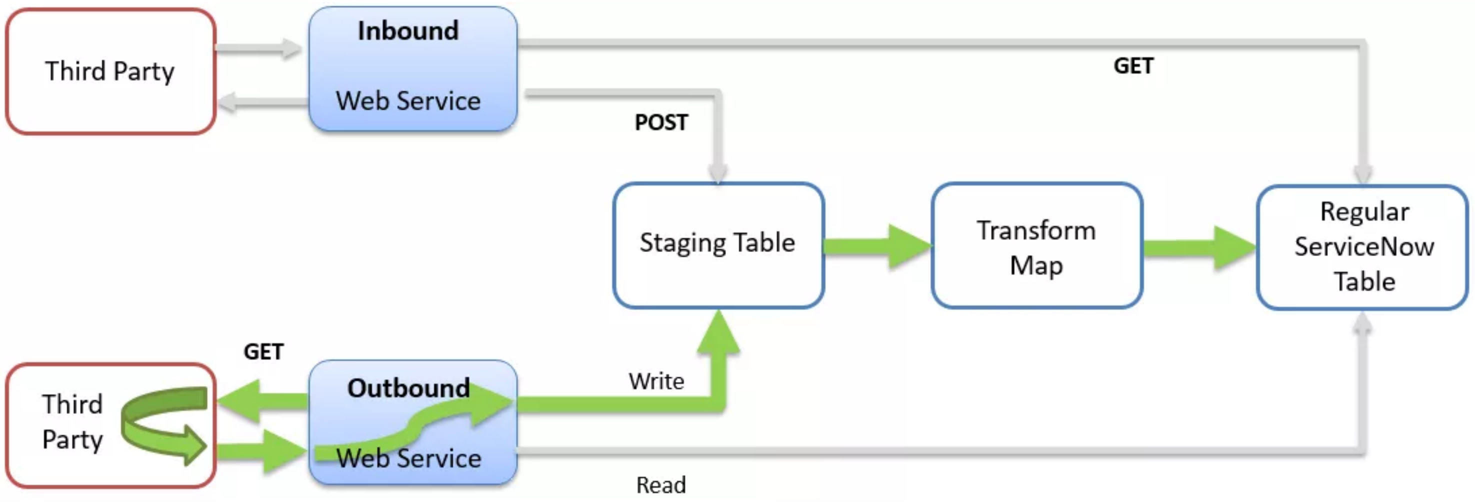
App Log (Application Logs) New for text Search Grid Split

All > Created on Today

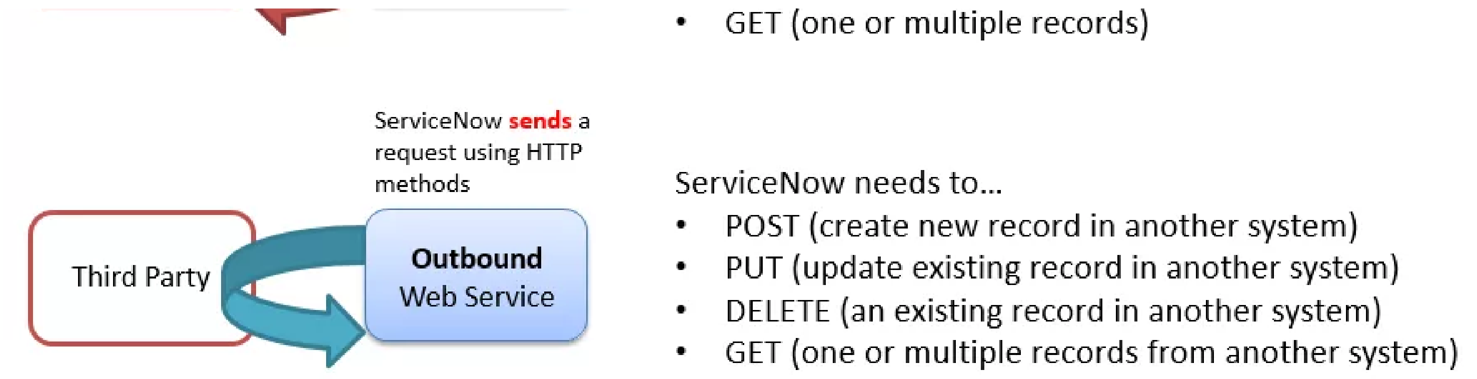
Created	Level	Message	App Scope
2016-07-27 23:59:33	Information	***Weather updated in 1827 msec	weather

This script can now be set up under the **System Definition** as a **Scheduled Job** and run as often as needed.

In summary, we built our outbound web service using the same components as found in an inbound web service.



One common question that arises when dealing with web services is, “What approach should I use?” The answer is… think about who is initiating the web service and what do you want to accomplish.



- GET (one or multiple records)

ServiceNow needs to…
- POST (create new record in another system)
- PUT (update existing record in another system)
- DELETE (an existing record in another system)
- GET (one or multiple records from another system)

Our team on the case

Peter Cawdron

Tags: ServiceNow, Web Services

7 comments

Soeren Corneliussen
February 18, 2019 Reply

Jillian Hunter
February 19, 2019 Reply

It is possible to structure your web services so there's no need for any scripting at all. The option of scripted web services allows for considerable flexibility. As an example, we've developed an inbound scripted web service for Splunk. This allows us to intercept and manipulate/transform the incoming web service before pushing it into a table, etc. In this way, scripted web services allow you to manage the complexity of different web service formats and massage them to suit ServiceNow. In the case of Splunk, the Splunk web service provided a JSON object which we then matched to records in ServiceNow and updated ServiceNow accordingly.

Mike
March 17, 2018 Reply

Peter, first of, great job on the post. Looks like your code within the outbound_request_for_weather is not executing. Evenmore, the application log appears to be empty. Any support is appreciated? Kind Regards!

Peter Cawdron
March 21, 2018 Reply

Hi Mike, thanks for checking out this blog post and trying the code. The weather API has changed, breaking this code, so I've revised it and updated the post. Once you get the initial outbound web service set up, you can test it by running the following as a background-script

```
sm = new sn_ws.RESTMessageV2("outbound_get_weather", "default get");
sm.setStringParameter("q", "Townsville,AU");// <<<——- put your location here
sm.setHttpTimeout(10000);

var getresponse = sm.execute();
var responseBody = getresponse.getBody();

var JSONdata = new global.JSON().decode(responseBody);

gs.info("weather_date_in_epoch " + JSONdata.dt);
gs.info("temperature " + JSONdata.main.temp );
```

and you should see the output as…

```
*** Script: weather_date_in_epoch 1521583200
*** Script: temperature 300.15
```

Shubham Rastogi
December 21, 2017 *Reply*

HI I am also using the same approach to consume the outbound soap service to get the data from the third party database.. Then how i can use that webservice in schedule job so that the process can be automated? any suggestion would be helpful.

Peter Cawdron
January 10, 2018 *Reply*

If you've already developed your outbound soap web service transform map, you should be able to use a script to trigger the transform. Remember, there are three steps in this process. (1) retrieve data via SOAP (2 & 3) load into an import set and transform the data. Have fun and test carefully

These links may help:
* outbound soap web service transform maps https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/import-sets/concept/c_WebServiceImportSets.html
* retrieve data via SOAP https://developer.servicenow.com/app.do#!/api_doc?v=jakarta&id=c_SOAPMessageV2API
* load into an import set and transform the data https://community.servicenow.com/thread/158812

Leave a Reply

Melbourne: 8/2 Russell St
Adelaide: 3/147 Pirie St
Brisbane: 19/123 Eagle St
Sydney: 10/50 Park St

Phone: 1300 780 432
Email: contactus@jds.net.au
JDS Privacy Policy