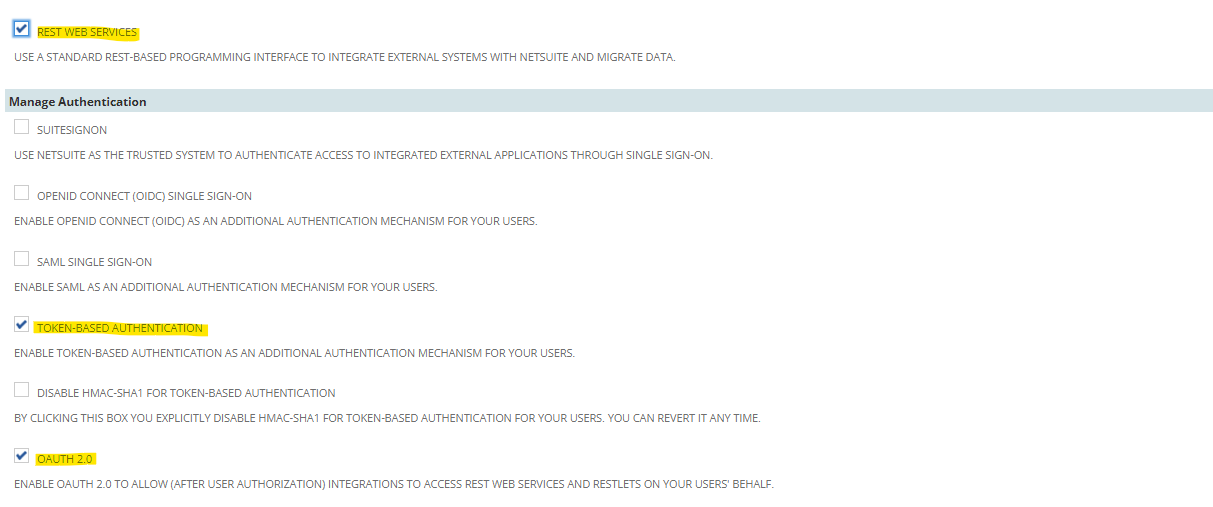
**RESTlet**

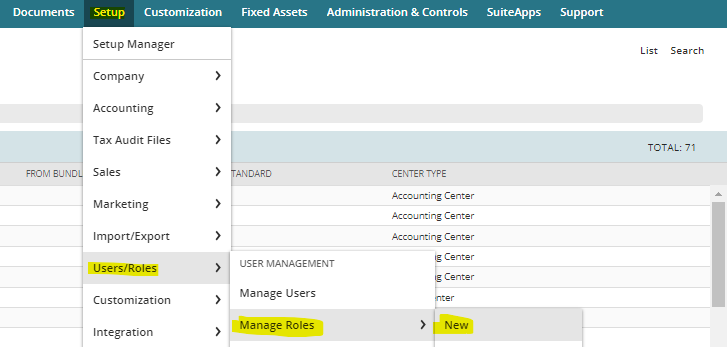
**Q1: Write down the enable features that are required for token-based authentication.**

Go to, Setup -> Company -> Enable features. Then under SuiteCloud subtab check 3 boxes. i.e. Rest Web Services, Token Based Authentication and Oath 2.0.

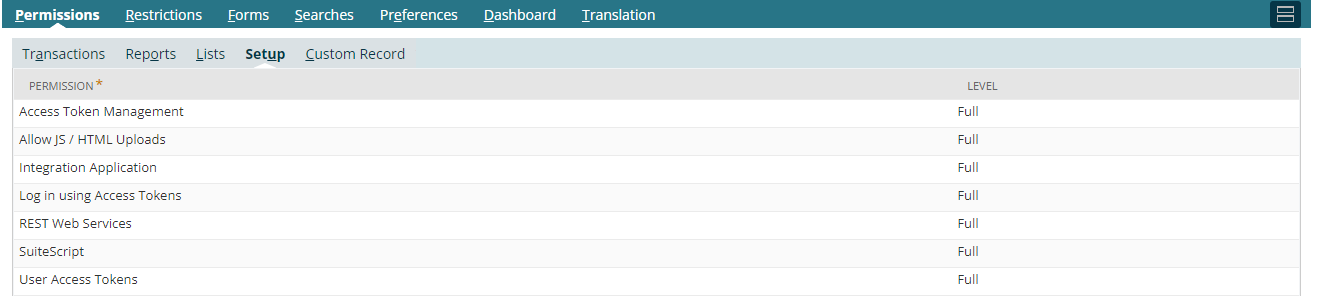


**Q2: Create a role with all the permissions required for token-based authentication.**

Step 1 - To create a role, navigate to, Setup -> Users & Roles -> Manage Roles -> New

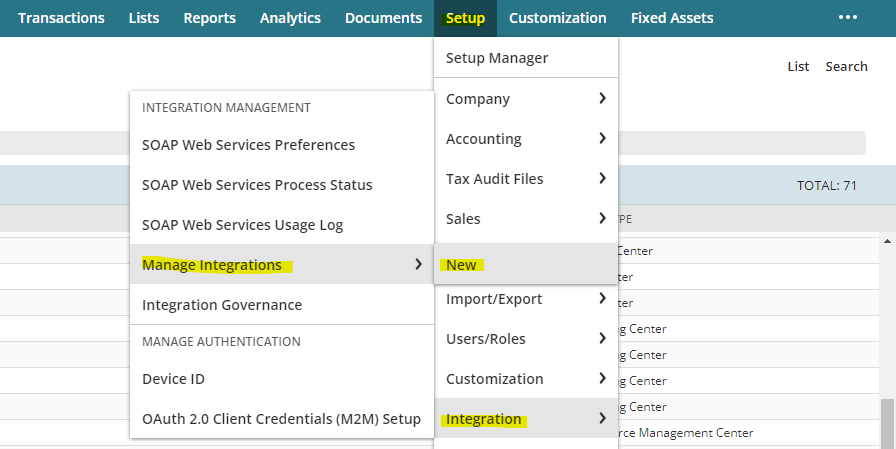


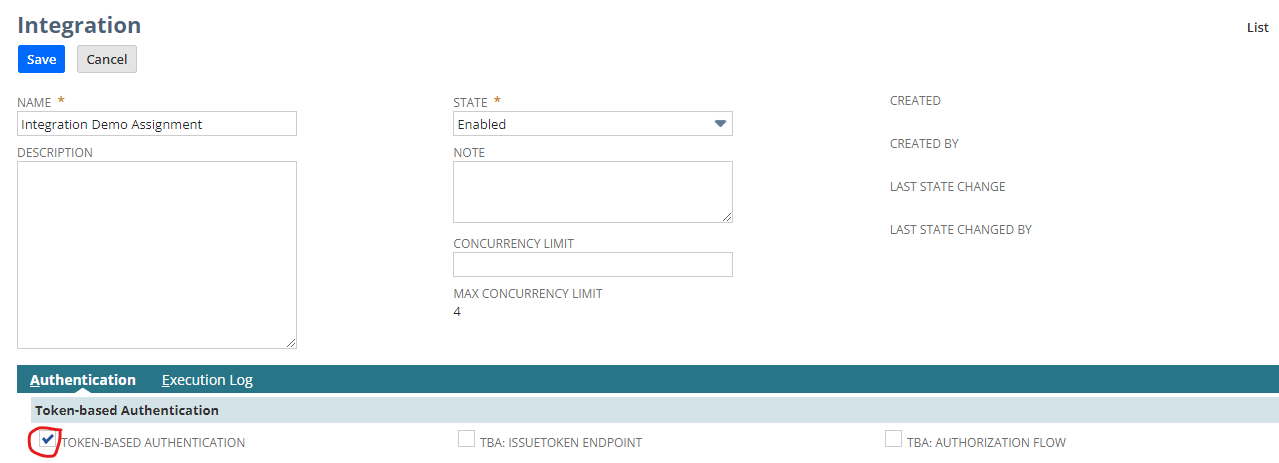
Give a name and id to the role and then give all seven permissions to the role for creating token-based authentication.



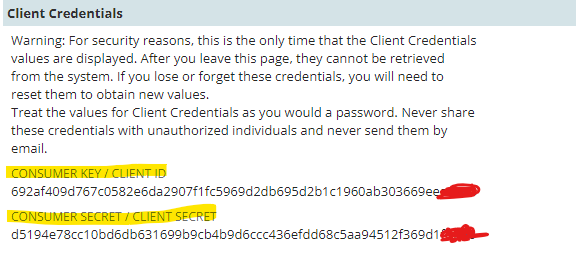
**Q3: Create an integration record and create an access token.**

Step 1 – Go to Setup -> Integration -> Manage Integrations -> New

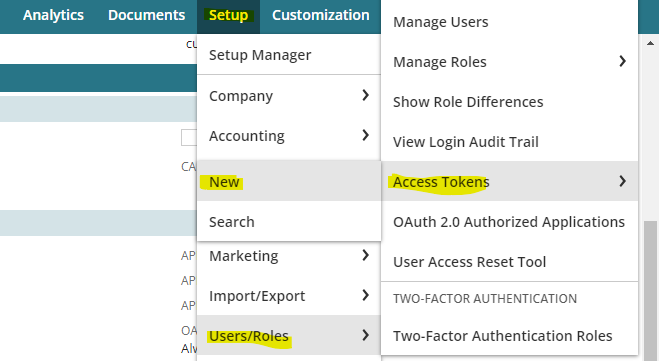
Step 2 – Give a name and check the token-based authentication box. Click on Save.



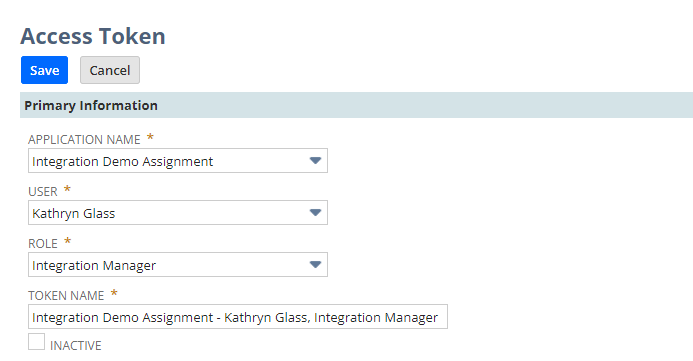
Step 3 – Copy the consumer key and consumer secret.



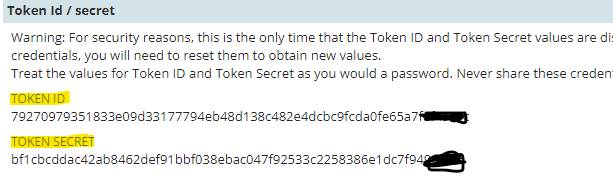
Step 4 – Now to create an access token, navigate to, setup -> users/roles -> Access tokens -> new



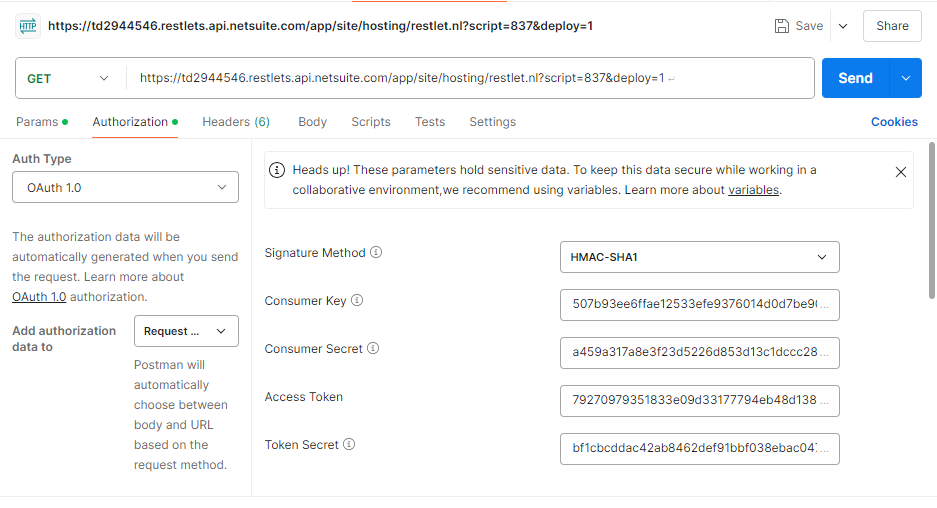
Step 5 – Now give the required value in the field and click on Save.



Step 6 – Copy the Token ID and Token Secret.



* After deploying the script copy the external URL and paste in POSTMAN under HTTPS GET request.
* Under Authorization select OAuth 1.0 and give all the key, secret and token.
* Select Realm and remember TD should be capital. Signature method should be HMAC-SHA256
* Add Authorization data to Request Headers, because OAuth 1.0 passes the data through the headers only.
* In the headers key should be content type and value should be application/json



**Q4: Fetch a Sales Order Record in NetSuite through postman.**

Go to postman and under params section create a new parameter as the variable name and assign a value it (in our case sales order internal id).

Then create a new variable in the script and assign the variable we created in postman to it. I used the **GET** method to fetch an existing record.

const get = (requestParams) => {

var so\_id = requestParams.so\_id\_postman

var so\_record = record.load({

type: record.Type.SALES\_ORDER,

id: so\_id,

isDynamic: true,

});

log.debug('Record got ', so\_record)

return so\_record

}

**Q5: Create a Sales Order Record in NetSuite through postman (don't hardcode data in script).**

I just created a JSON in the body in raw format and I used **POST** method to create a new sales order record.

I took customer id as 319 and with at least one line level item in order to save the record, so I added an item to my sales order having item id 338.

***Code:***

// POST method to create a new record

const post = (requestBody) => {

try {

log.debug('Request body: ', requestBody);

var sales\_order = record.create({

type: record.Type.SALES\_ORDER,

isDynamic: true,

defaultValues: {

entity: requestBody.entity

}

});

requestBody.items.forEach(function (item) {

sales\_order.selectNewLine({

sublistId: 'item'

});

sales\_order.setCurrentSublistValue({

sublistId: 'item',

fieldId: 'item',

value: item.itemid

});

sales\_order.commitLine({

sublistId: 'item'

});

});

var sales\_order\_id = sales\_order.save();

log.debug('Sales order created with ID: ', sales\_order\_id);

return {

success: true,

sales\_order\_id: sales\_order\_id

};

} catch (e) {

return {

success: false,

message: e.message

};

}

};

***JSON data:***

{

"entity": 319,

"items": [

{

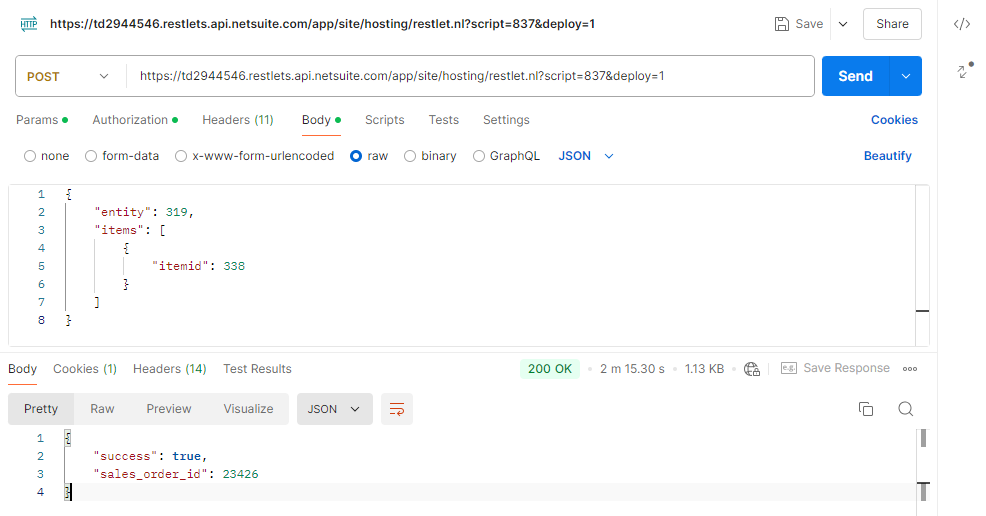
"itemid": 338

}

]

}

***Output:***



So, a new Sales Order created with id **23426**

**Q6: UPDATE the Sales Order Record in NetSuite through postman (don't hardcode data in script) also update a body field and add new line item.**  
Here to update an existing record I used the **PUT** method. Here I updated the customer's name in the newly created sales order i.e. 23426.

I changed the customer having id 319 to 534 by just passing the id of the record I want to edit and the new entity id I want to assign to the record.

***Code:***

// PUT method to update a record

const put = (requestBody) => {

try {

log.debug('Request body: ', requestBody);

var sales\_order = record.load({

type: record.Type.SALES\_ORDER,

id: requestBody.sales\_order\_id,

isDynamic: true

});

var newEntity = requestBody.entity;

sales\_order.setValue({

fieldId: 'entity',

value: newEntity

});

var updated\_so\_id = sales\_order.save();

log.debug('Sales order updated with ID: ', updated\_so\_id);

return {

success: true,

updated\_so\_id: updated\_so\_id

};

} catch (e) {

return {

success: false,

message: e.message

};

}

};

***JSON data:***

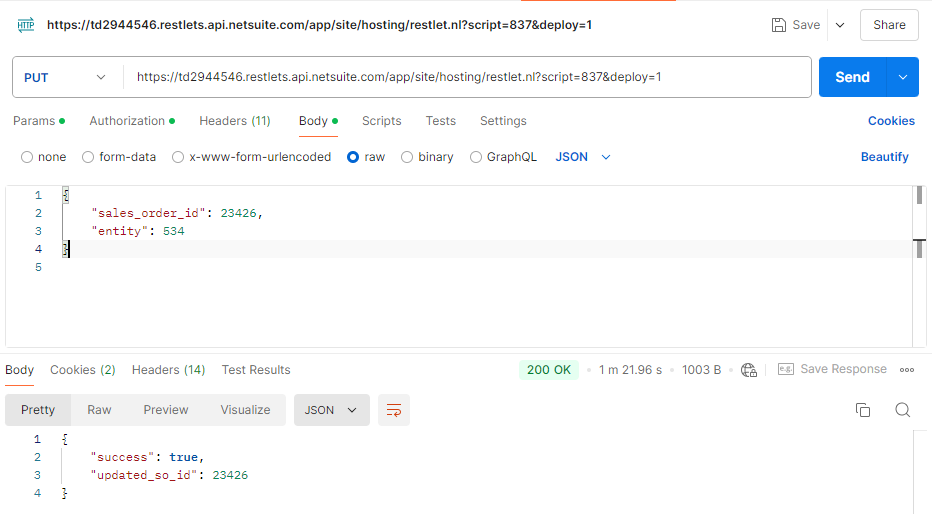
{

"sales\_order\_id": 23327,

"entity": 534

}

***Output:***



***Previous customer of the record:***



***Updated customer of the record:***



**Q7: DELETE the line items of the Sales Order record also delete the whole record.**

Here I deleted the record we just edited in the previous question having internal id 23426.

I deleted by creating a variable as a key and passed the internal id of the record as value and used **DELETE** method to the same.

***Code:***

// DELETE method to delete an existing record

const doDelete = (requestParams) => {

var so\_id\_dlt = requestParams.so\_id\_to\_dlt

log.debug('so id to delete ', so\_id\_dlt)

try{

var deleted\_id = record.delete({

type: record.Type.SALES\_ORDER,

id: so\_id\_dlt

})

log.debug('Sales order deleted with id ', id)

return {

success: true,

deleted\_so\_id: deleted\_id

}

}catch(e){

return{

success: false,

message: e.message

}

}

}

Now we can not find the record having id 23426 in the sales order.

