Techfino

NETSUITE INTEGRATION

Guide to Setting up **Token-Based Authentication** in NetSuite

+1 (877) 563-1405

contact@techfino.com

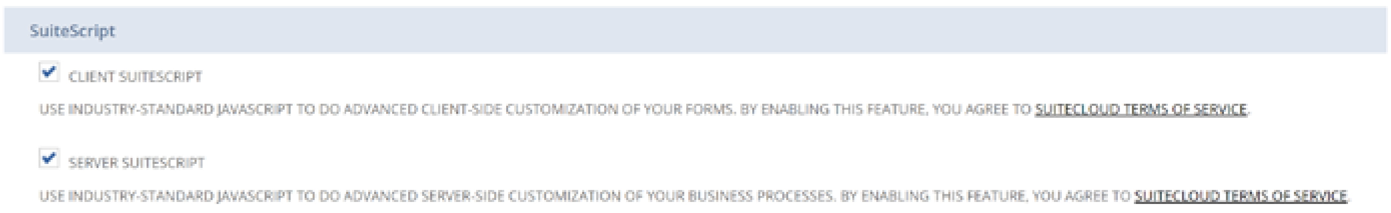
This walk-thru guide will provide a step-by-step guide to getting started with token-based authentication in NetSuite.

In addition, we provide a SuiteScript 2.0 example of using Token Based Authentication to make SuiteTalk calls to get/set Budget values which are not accessible via the Suite-Script API. We have also provided a python script which illlustrates how to externally connect to a RESTlet using TBA.

1.1 Enable Features

Token Based Authentication must first be enabled in the account. Under **Setup > Company > Setup Tasks > Enable Features** navigate to the SuiteCloud subtab. Enable the required features:

- Client SuiteScript (prerequisite for Server side SuiteScript)
- Server SuiteScript (prerequisite for RESTlets)



SuiteScript

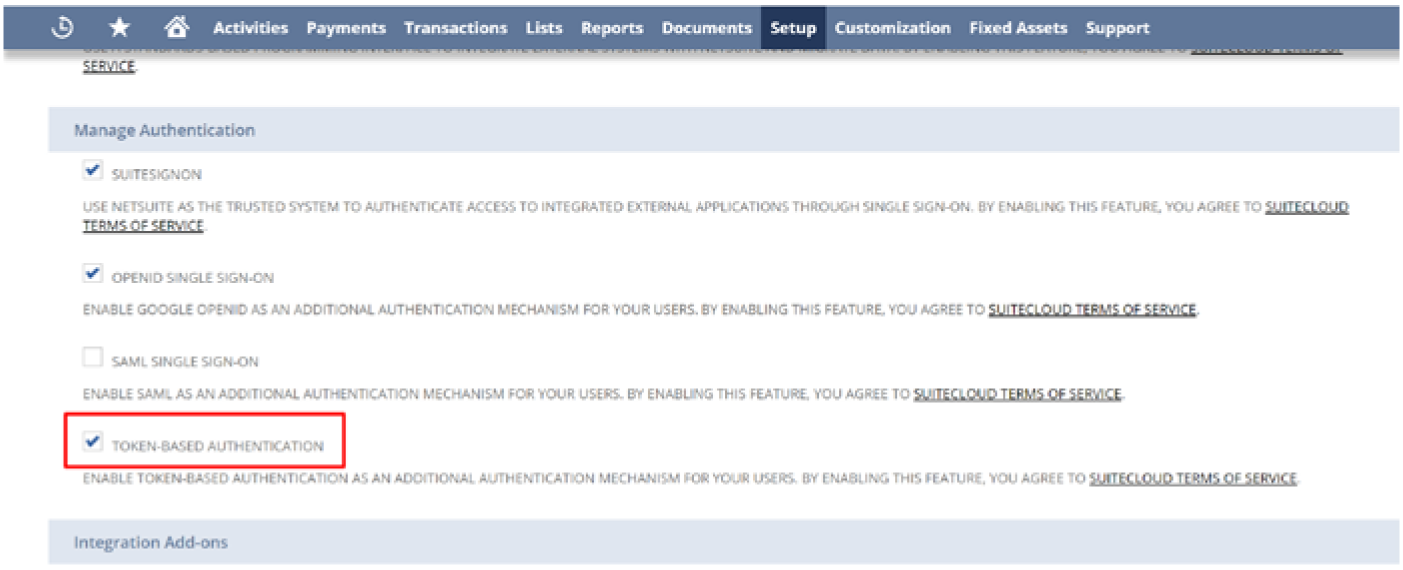
✓ CLIENT SUITESCRIPT

USE INDUSTRY-STANDARD JAVASCRIPT TO DO ADVANCED CLIENT-SIDE CUSTOMIZATION OF YOUR FORMS. BY ENABLING THIS FEATURE, YOU AGREE TO SUITECLOUD TERMS OF SERVICE.

✓ SERVER SUITESCRIPT

USE INDUSTRY-STANDARD JAVASCRIPT TO DO ADVANCED SERVER-SIDE CUSTOMIZATION OF YOUR BUSINESS PROCESSES. BY ENABLING THIS FEATURE, YOU AGREE TO SUITECLOUD TERMS OF SERVICE.

Navigate to the manage authentication section and enable the Token-based Authentication if it is not already enabled.



SERVICE.

Manage Authentication

[✓] SUITESIGNON

USE NETSUITE AS THE TRUSTED SYSTEM TO AUTHENTICATE ACCESS TO INTEGRATED EXTERNAL APPLICATIONS THROUGH SINGLE SIGN-ON. BY ENABLING THIS FEATURE, YOU AGREE TO SUITECLOUD TERMS OF SERVICE.

[✓] OPENID SINGLE SIGN-ON

ENABLE GOOGLE OPENID AS AN ADDITIONAL AUTHENTICATION MECHANISM FOR YOUR USERS. BY ENABLING THIS FEATURE, YOU AGREE TO SUITECLOUD TERMS OF SERVICE.

[ ] SAML SINGLE SIGN-ON

ENABLE SAML AS AN ADDITIONAL AUTHENTICATION MECHANISM FOR YOUR USERS. BY ENABLING THIS FEATURE, YOU AGREE TO SUITECLOUD TERMS OF SERVICE.

[✓] TOKEN-BASED AUTHENTICATION

ENABLE TOKEN-BASED AUTHENTICATION AS AN ADDITIONAL AUTHENTICATION MECHANISM FOR YOUR USERS. BY ENABLING THIS FEATURE, YOU AGREE TO SUITECLOUD TERMS OF SERVICE.

Integration Add-ons

The configuration page must be **saved** for the changes to take effect.

1.2 Role

Token based authentication is a per user authentication and requires certain permissions in NetSuite. An existing role can be used (recommended) or a new role can be created.

The relevant role permissions are under the 'Setup' subtab the following details were gleaned from SuiteAnswer(41898).

Access Token Management

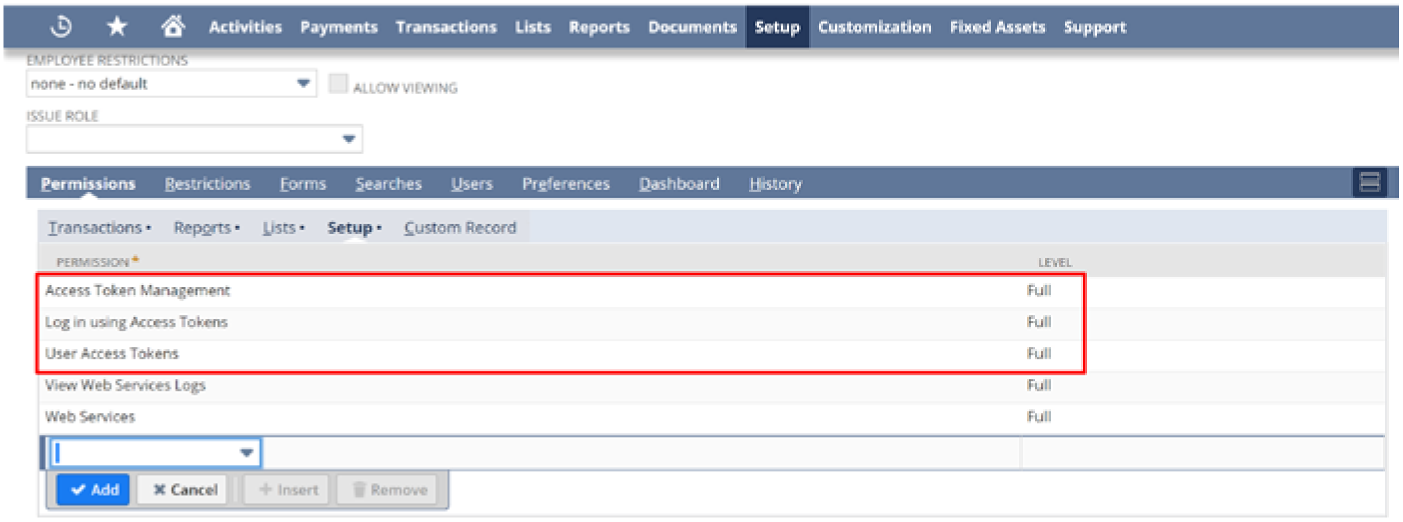
- Users with this permission can create, assign, and manage tokens for any user in the company.
- Users with this permission cannot use token–based authentication to log in to the NetSuite UI.

Log in using Access Tokens

- Users with this permission can manage their own tokens using the Manage Access Tokens link in the Settings portlet, and they can log in using a token.

User Access Tokens

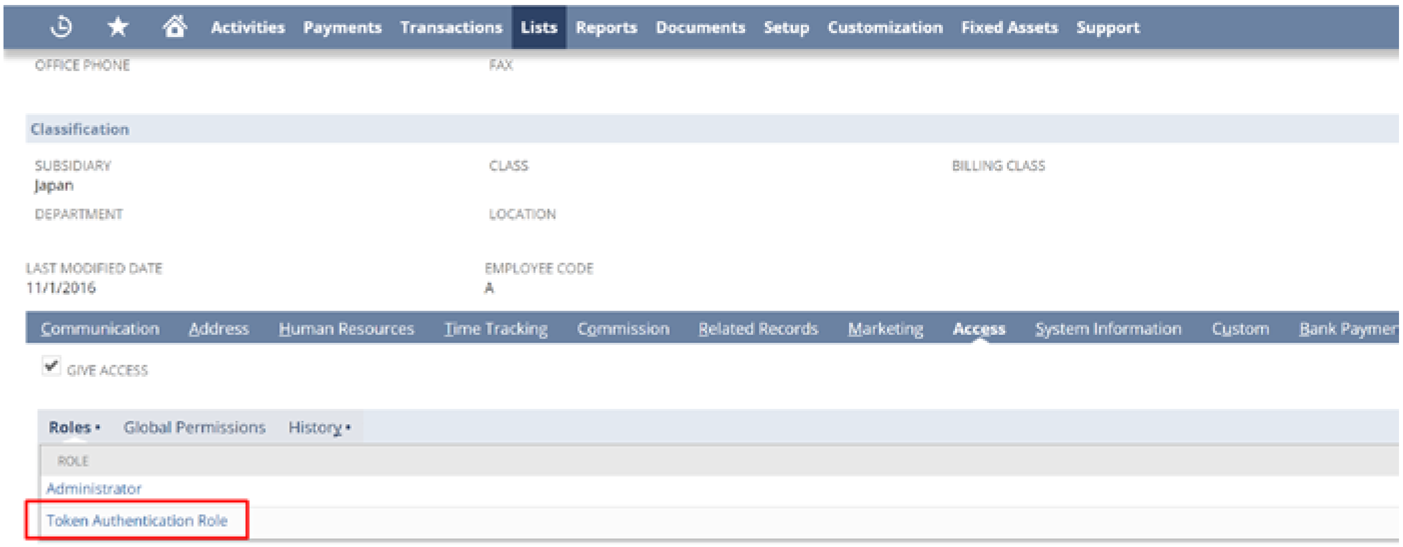
- Users with only this permission can log in using a token, that is, they can to use tokens to call a RESTlet.
- Users with only this permission cannot manage tokens or access pages where tokens are managed.


Activities Payments Transactions Lists Reports Documents Setup Customization Fixed Assets Support
Permissions Restrictions Forms Searches Users Preferences Dashboard History
Transactions • Reports • Lists • Setup • Custom Record
Add Cancel Insert Remove

EMPLOYEE RESTRICTIONS
none - no default
ALLOW VIEWING
ISSUE ROLE

PERMISSION*	LEVEL
Access Token Management	Full
Log in using Access Tokens	Full
User Access Tokens	Full
View Web Services Logs	Full
Web Services	Full

The Token Authentication Role will need to be assigned to all employees associated with the integration under 'Access' subtab on their employee record


Activities Payments Transactions Lists Reports Documents Setup Customization Fixed Assets Support
Communication Address Human Resources Time Tracking Commission Related Records Marketing Access System Information Custom Bank Paymen
Roles • Global Permissions History •

OFFICE PHONE
FAX

Classification

SUBSIDIARY
Japan
CLASS
BILLING CLASS
DEPARTMENT
LOCATION
LAST MODIFIED DATE
11/1/2016
EMPLOYEE CODE
A

✓ GIVE ACCESS

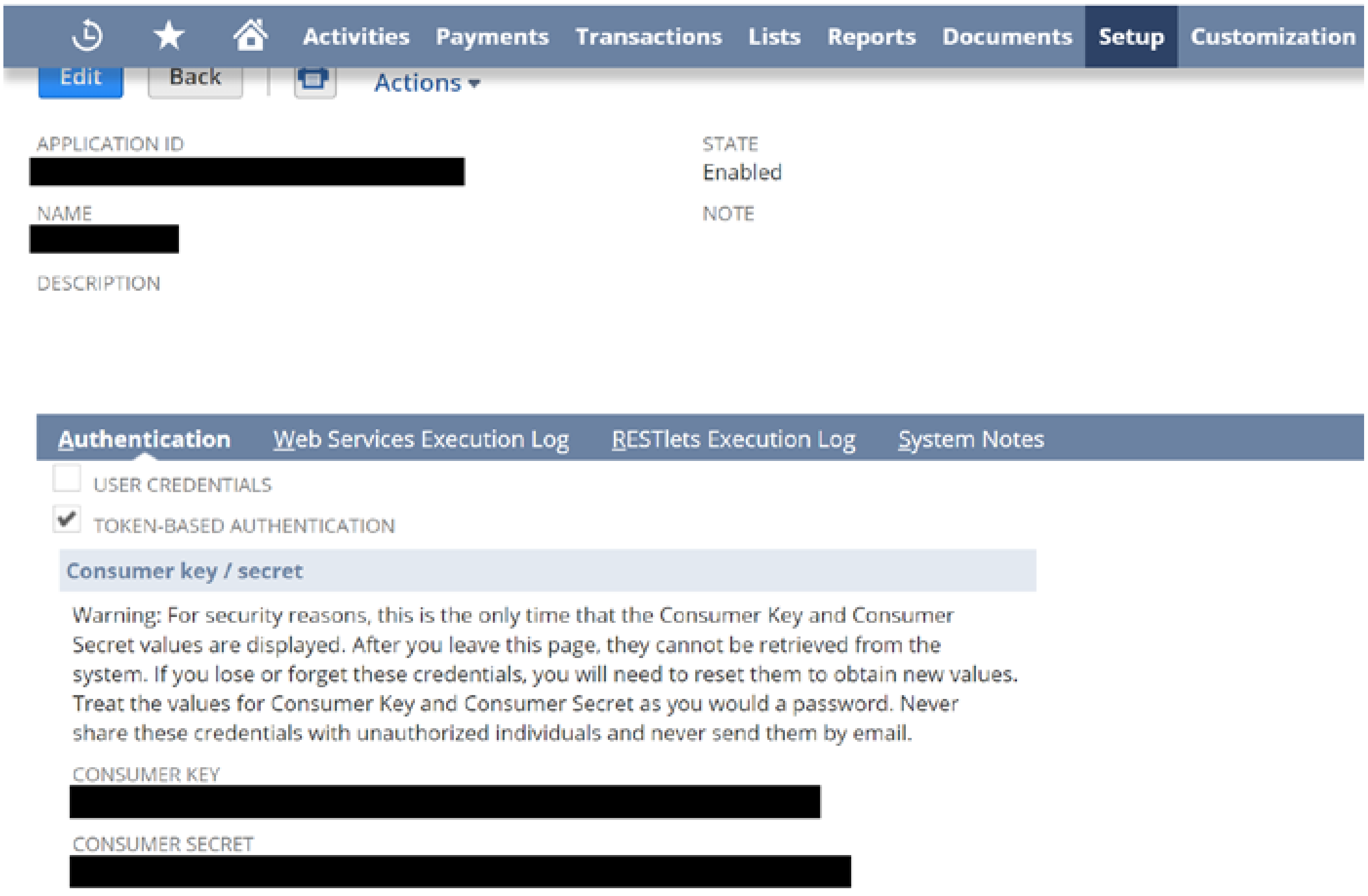
ROLE
Administrator
Token Authentication Role

1.3 Integration Record

Before connecting with a token, an integration record is required for authentication. A new integration record should be used and can be created by navigating to **Setup > Manage Integrations > New**.

The name field should be filled in along with ensuring that the 'TOKEN-BASED AUTHENTICATION' checkbox is checked. Upon saving you will be given a Consumer key / Consumer secret.

NOTE: These values will not show up again after navigating away from the page for security concerns. Store the values somewhere securely treating them as you would a password. The values will be utilized in authentication later.



APPLICATION ID
[redacted]

STATE
Enabled

NAME
[redacted]

NOTE

DESCRIPTION

☐ USER CREDENTIALS

☑ TOKEN-BASED AUTHENTICATION

Consumer key / secret

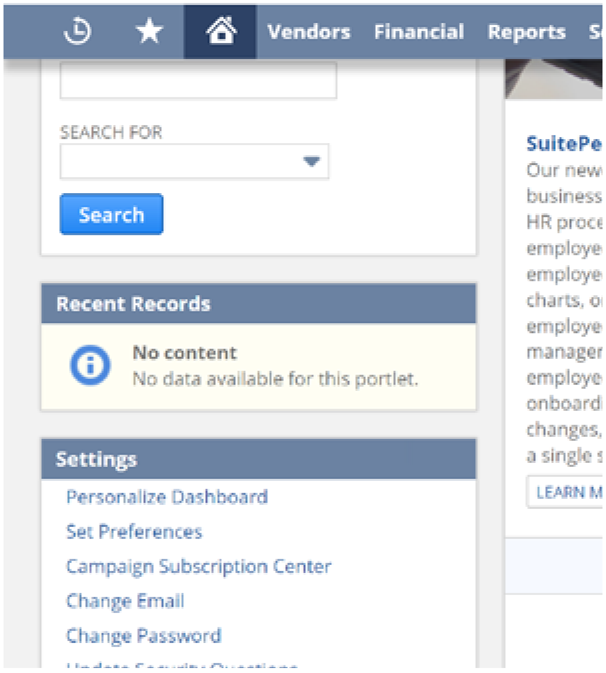
Warning: For security reasons, this is the only time that the Consumer Key and Consumer Secret values are displayed. After you leave this page, they cannot be retrieved from the system. If you lose or forget these credentials, you will need to reset them to obtain new values. Treat the values for Consumer Key and Consumer Secret as you would a password. Never share these credentials with unauthorized individuals and never send them by email.

CONSUMER KEY
[redacted]

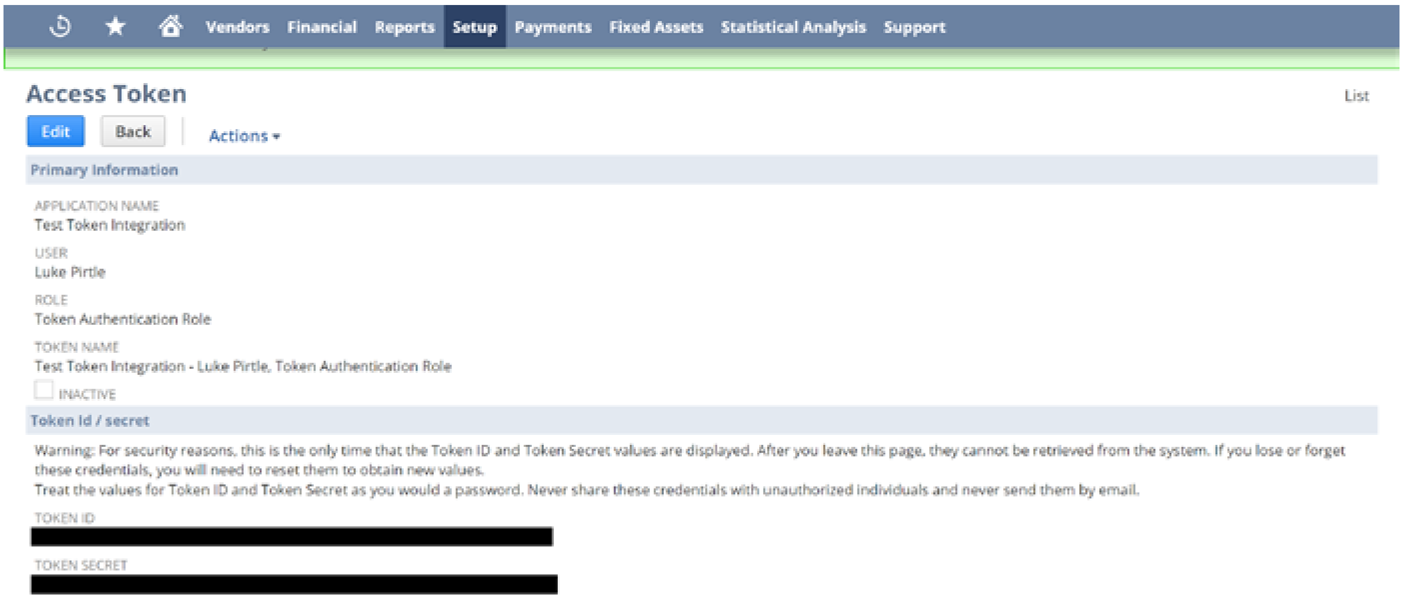
CONSUMER SECRET
[redacted]

1.4 Creating the Token

With the integration record created and proper role assigned, a token can be created for authentication. To create a token, have the user with the token authentication role login. Click the ‘Manage Access Tokens’ link available on the home dashboard under settings



Create a new token and select the Application Name that corresponding to the associated integration record created earlier. Again, an ID and secret will be provided.



Access Token

Primary Information

APPLICATION NAME
Test Token Integration

USER
Luke Pirtle

ROLE
Token Authentication Role

TOKEN NAME
Test Token Integration - Luke Pirtle, Token Authentication Role

☐ INACTIVE

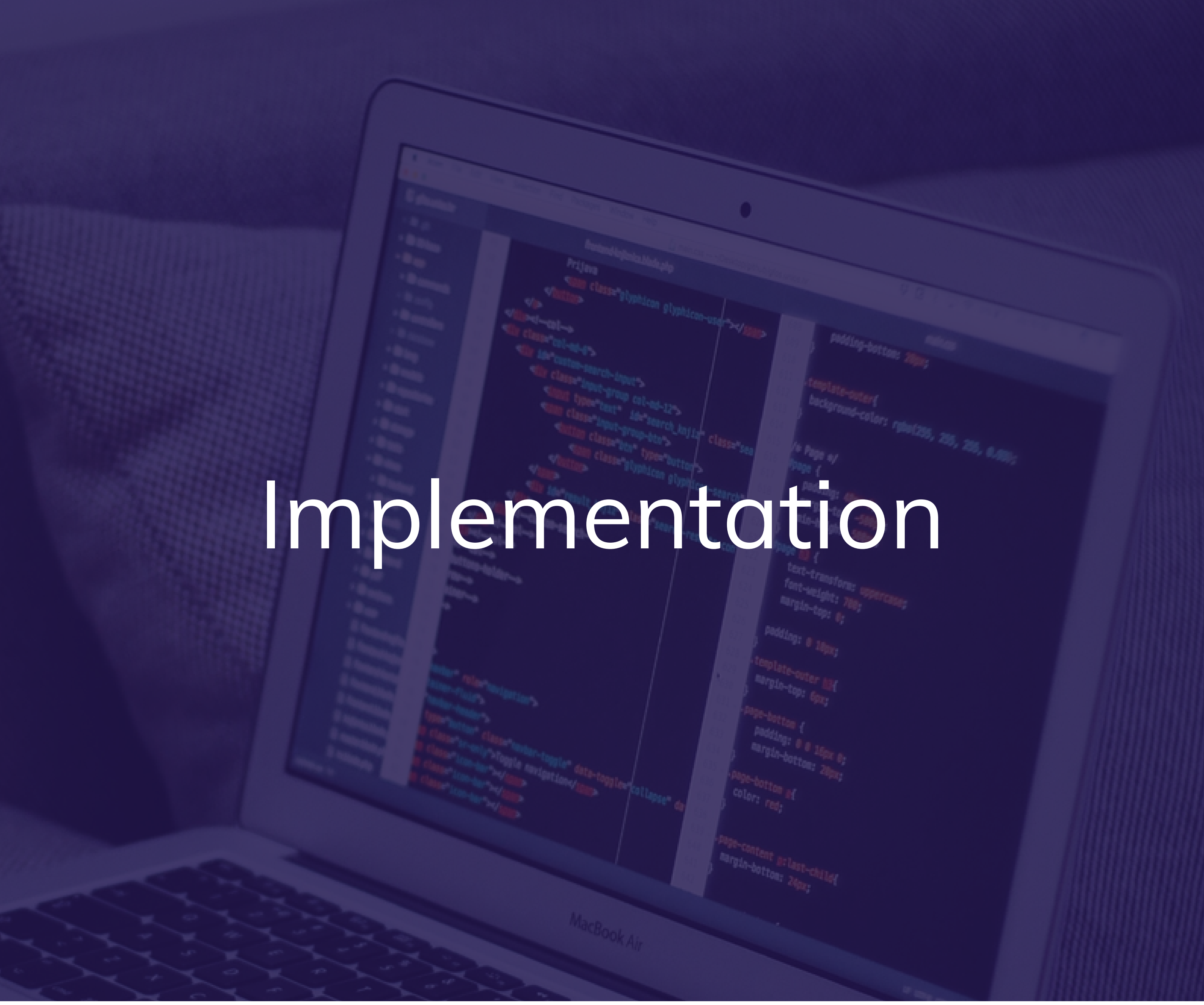
Token Id / secret

Warning: For security reasons, this is the only time that the Token ID and Token Secret values are displayed. After you leave this page, they cannot be retrieved from the system. If you lose or forget these credentials, you will need to reset them to obtain new values.
Treat the values for Token ID and Token Secret as you would a password. Never share these credentials with unauthorized individuals and never send them by email.

TOKEN ID
[redacted]

TOKEN SECRET
[redacted]

NOTE: These values will not show up again after navigating away from the page for security concerns. Store the values somewhere securely treating them as you would a password. The values will be utilized in authentication later



Implementation

NetSuite Token Based Authentication is simply an implementation of OAuth V1.0, meaning that any OAuth library can be used in order to send requests in a preferred language.

While a library solution works well for RESTlet connections, web services stores this information in the payload itself and isn't as simple. Two examples showing authentication for both are included. To download or view the entire code for this tutorial, visit our GitHub page at: https://github.com/Techfino/TokenBasedAuthentication/

2.3 Python Post RESTlet Implementation (RESTlet)

RESTlets are designed to take third party Oauth requests and are very simple to implement. Below is some slightly altered python code from SuiteAnswer (42165) used to test a Token RESTlet integration. NetSuite has numerous other code examples for Token Based Authentication in SuiteAnswer (42171) for Java, JavaScript, PHP and Python.

```
import oauth2 as oauth
import json
import requests
import time

url = “https://rest.netsuite.com/.../restlet.nl?script=X&deploy=Y”
token = oauth.Token(key=”____________________________”,
secret=”____________________________”)
consumer = oauth.Consumer(key=”____________________________”,
secret=”____________________________”)
http_method = “POST”
realm = “123456” # NetSuite account ID
# the JSON data to be sent to the RESTlet
payload = {
   name:value,
   foo:bar,
   duck:hunt,
}
params = {
         ‘oauth_version’: “1.0”,
         ‘oauth_nonce’: oauth.generate_nonce(),
         ‘oauth_timestamp’: str(int(time.time())),
         ‘oauth_token’: token.key,
         ‘oauth_consumer_key’: consumer.key
}

req = oauth.Request(method=http_method, url=url, parameters=params)
```

```
signature_method = oauth.SignatureMethod_HMAC_SHA1()
req.sign_request(signature_method, consumer, token)
header = req.to_header(realm)
headery = header['Authorization'].encode('ascii', 'ignore')
headerx = {"Authorization": headery, "Content-Type": "application/json"}
print(headerx)
conn = requests.post(url, headers=headerx, data=json.dumps(payload))
print("Result: " + conn.text)
print(conn.headers)
```

SuiteScript SuiteTalk Integration (Webservices)

SuiteScript is a powerful tool inside NetSuite but it has its limitations. Not all records are scriptable and support a look-up by External ID. The most common of records which are not accessible via SuiteScript that we run into are Budgets. The Budget record can be searched, however even this is limited. Looking at the Schema Browser however it can be noted that SuiteTalk supports Budgets for all normal operations.

Under the right circumstances, a SuiteScript to SuiteTalk integration can be a good solution to this problem. Below we show a portion of a Techfino SuiteScript 2.0-SuiteTalk web service library that uses Token Based Authentication or the legacy NLAuth method to authenticate and make SuiteTalk web service calls. For those interested in the full code, be sure to visit our GitHub page at: https://github.com/Techfino/TokenBasedAuthentication/

```
/**
 * Class encapsulating the webservices suitetalk functionality
 * @return {void}
 */
function webservicesObject(credentials) {

    this.accountNumber = runtime.accountId;
    // determine which type of authentication to use from initialization object
    if (credentials.username){
        this.username = credentials.username;
        this.password = credentials.password;
        this.roleId = credentials.roleId;
        this.applicationId = credentials.applicationId;
        this.generatePassport = this.generateNLAuthPassport;
    } else if (credentials.tokenId) {
        this.tokenId = credentials.tokenId;
        this.tokenSecret = credentials.tokenSecret;
        this.consumerKey = credentials.consumerKey;
        this.consumerSecret = credentials.consumerSecret;
        this.generatePassport = this.generateTokenPassport;
    } else {
        throw 'Invalid Credentials Object';
    }

    var xmlArray = [];
    xmlArray.push('<?xml version="1.0" encoding="UTF-8"?>');
    xmlArray.push('<soapenv:Envelope xmlns:xsd="http://www.w3.org/2001/
XMLSchema" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.
w3.org/2001/XMLSchema">');
    xmlArray.push('<soapenv:Header>');
    xmlArray.push('{AUTHENTICATION}');
    xmlArray.push('</soapenv:Header>');
    xmlArray.push('<soapenv:Body xmlns:platformMsgs="urn:messages_2016_1.
platform.webservices.netsuite.com" xmlns:platformCore="urn:core_2016_1.
platform.webservices.netsuite.com" xmlns:tranInvt="urn:inventory_2016_1.
transactions.webservices.netsuite.com” xmlns:tranInvtTyp=”urn:types.
inventory_2016_1.transactions.webservices.netsuite.com”
xmlns:platformCommon=”urn:common_2016_1.platform.webservices.netsuite.
com” xmlns:platformCommonTyp=”urn:types.common_2016_1.platform.
webservices.netsuite.com” xmlns:listRel=”urn:relationships_2016_1.lists.
webservices.netsuite.com” xmlns:listAcctTyp=”urn:types.accounting_2016_1.
lists.webservices.netsuite.com” xmlns:listAcct=”urn:accounting_2016_1.lists.
webservices.netsuite.com” xmlns:tranFin=”urn:financial_2016_1.transactions.
webservices.netsuite.com” xmlns:tranFinTyp=”urn:types.financial_2016_1.
transactions.webservices.netsuite.com”>’);
        xmlArray.push(‘{BODY}’);
        xmlArray.push(‘</soapenv:Body>’);
        xmlArray.push(‘</soapenv:Envelope>’);

        this.envelope = xmlArray.join(‘’);

        this.queryURL = this.getDataCenterUrl();
    }

    /**
     * creates an NLAuthentication header
     * @return {string} passport with proper NLAuth authentication verification
     */
    webservicesObject.prototype.generateNLAuthPassport = function() {
        var xmlArray = []

        xmlArray.push(‘<ns1:passport soapenv:mustUnderstand=”0”
xmlns:ns1=”urn:messages_2016_1.platform.webservices.netsuite.com”>’);
        xmlArray.push(‘<ns:email xmlns:ns=”urn:messages_2016_1.platform.
webservices.netsuite.com”>’ + this.username + ‘</ns:email>’);
        xmlArray.push(‘<ns:password xmlns:ns=”urn:messages_2016_1.platform.
webservices.netsuite.com”>’ + this.password + ‘</ns:password>’);
        xmlArray.push(‘<ns:account xmlns:ns=”urn:messages_2016_1.platform.
webservices.netsuite.com”>’ + this.accountNumber + ‘</ns:account>’);
        xmlArray.push(‘<ns:role xmlns:ns=”urn:messages_2016_1.platform.
webservices.netsuite.com” internalId=”’ + this.roleId + ‘”/>’);
```

```
        xmlArray.push('</ns1:passport>');
        xmlArray.push('<ns1:applicationInfo soapenv:mustUnderstand="0"
xmlns:ns1="urn:messages_2016_1.platform.webservices.netsuite.com">');
        xmlArray.push('<ns:applicationId xmlns:ns="urn:messages_2016_1.
platform.webservices.netsuite.com">' + this.applicationId + '</ns:applicationId>');
        xmlArray.push('</ns1:applicationInfo>');

        return xmlArray.join('');

    }

    /**
     * Creates a token authentication header
     * @return {string} passport with proper token authentication verification
     */
    webservicesObject.prototype.generateTokenPassport = function() {

        if (!this.queryURL) return "";

        var timestamp = Math.round(new Date().getTime() / 1000);
        var nonce = this.generateNonce();
        var xmlArray = [];
        // create string for signature
        var baseString = escape(this.accountNumber) + '&' +
            escape(this.consumerKey) + '&' +
            escape(this.tokenId) + '&' +
            escape(nonce) + '&' +
            escape(timestamp);

        var key = escape(this.consumerSecret) + '&' +
            escape(this.tokenSecret);

        // run google signature algorithm. NetSuite api for HMAC function needs a
secret key GUID not a raw string which is difficult to create dynamically.
```

```
            var signature = CryptoJS.HmacSHA1(baseString, key).toString(CryptoJS.enc.
Base64);

            xmlArray.push('<ns:tokenPassport xmlns:ns="urn:messages_2016_1.
platform.webservices.netsuite.com">');
            xmlArray.push('<ns:account>' + this.accountNumber + '</ns:account>');
            xmlArray.push('<ns:consumerKey>' + this.consumerKey + '</
ns:consumerKey>');
            xmlArray.push('<ns:token>' + this.tokenId + '</ns:token>');
            xmlArray.push('<ns:nonce>' + nonce + '</ns:nonce>');
            xmlArray.push('<ns:timestamp>' + timestamp + '</ns:timestamp>');
            xmlArray.push('<ns:signature algorithm="HMAC_SHA1">' + signature + '=</
ns:signature>');
            xmlArray.push('</ns:tokenPassport>');

            return xmlArray.join('');
        };

        /**
         * Generates random string of alphanumeric characters for use in encryption
         * @return {string}        string of 20 random alphanumeric characters
         */
        webservicesObject.prototype.generateNonce = function() {
            var chars =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
            var NONCE_LENGTH = 20; // default to 20
            var result = '';
            for (var i = NONCE_LENGTH; i > 0; --i) result += chars[Math.floor(Math.
random() * chars.length)];
            return result;
        };
```

About Luke Pirtle

Luke is a Technical Consultant at Techfino with over three years of NetSuite development experience with deep NetSuite expertise in the area of Integration, Customization, and SuiteScript 2.0. If you found this article helpful, you can reach out to Luke at luke.pirtle@techfino.com

Techfino

Techfino is a leading provider of NetSuite Consulting services and specializes in serving clients in the Retail, Wholesale, Technology and Non-Profit verticals and helping them leverage best-in-breed software for maximum utility.

our blog www.techfino.com linkedin