JDPOWER

Shared Secret Security Protocol

Integration Guide

June 2025

TABLE OF CONTENTS

INTRODUCTION **4**
CONTACTING CLIENT SUPPORT **4**
ABOUT THE AUTHTOKEN HEADER **4**
Including Token in Request Message **4**
Using Security in the Portal **5**
Setting the Key ID **5**
Setting the Date **6**
Setting the Signature Digest **6**
Developer Steps for Creating the Signature Digest 6
Understanding the SHA-1 Secure Hash Algorithm 6
Postman Example: Creating a Shared Signature Digest **7**
JAVA Script Example : Creating a Shared Signature Digest **8**
PHP Example: Creating a Shared Signature Digest **8**
JAVA Example: Creating a Shared Signature Digest **9**
C# Example: Creating a Shared Signature Digest **10**
Python Example: Creating a Shared Signature Digest **11**
VBA HMAC-SHA1 Signature Generator for API Authentication **11**

INTRODUCTION

To properly secure your usage of services you must integrate a security token protocol into the Authtoken header of the request to the service. This document describes how a developer would build and integrate the token into their request.

CONTACTING CLIENT SUPPORT

Client Support is available by phone toll-free at (800) 937-3661, Monday through Friday, from 6:00 a.m. to 5:00 p.m. Pacific Time, or you can reach Client Support by email at support@chromedata.com. This team can help you with product support, billing questions, and other inquiries.

ABOUT THE AUTHTOKEN HEADER

You must enter four keys into the request headers to the service, as shown in the following table.

Key	Value
Accept	application/json
Content-Type	application/json
Date	Date used to create the token
Authtoken	Token

The rest of this document describes how the Authtoken value is generated.

Including Token in Request Message

When you call a REST service over HTTPS, you are expected to build a token and set the Authtoken header to the token value. At a high level, the token is built using a comma delimited list of name value pairs. The name fields that need to be set are:

Name	Description
Tyk.keyId	The API key assigned to the customer in the Portal. This id can be acquired on the Access page of the Portal
Algorithm	A value indicating the signature method being used. For our customers this should always be set to "hmac-sha1".
date	Current Timestamp expressed as Date and Time field.
Signature	A value produced by creating a HMAC hash of "date: " + $Date using SHA-1 with the assigned Api Secret and base 64 encoding the result.

To build a token, the developer should concatenate the above name/value pairs as follows:

```
Authtoken = 'Signature keyId="' + Tyk.KeyID + '", algorithm="hmac-sha1", signature="'
+ encodeURIComponent(StringtoSign) + '"'

signatureContentString = "date: Thu, 15 May 2025 17:40:21 GMT"
StringtoSign =  HmacSHA1(signatureContentString,apiSecret)
```

Sample Token using the following keyId/secret and Date:

```
Date = Thu, 15 May 2025 17:40:21 GMT
KeyId  =
"eyJvcmciOiI2NDI0YTEwNjY2NDU4MDAwMDFmMjk5ODAiLCJpZCI6Ijk3Y2Q0NmE0ZmI1NTQ4Yjk5YzIxYmUxMzgzODgwMDg0IiwiaCI6Im11cm11cjEyOCJ9"
API Secret: "ZDRjM2JjMzYwZDdkNGQ3MjgwNWE4N2Q5NTMyOGYxOGE="
```

Note: Both Authtoken and Date Headers must be set as follows:

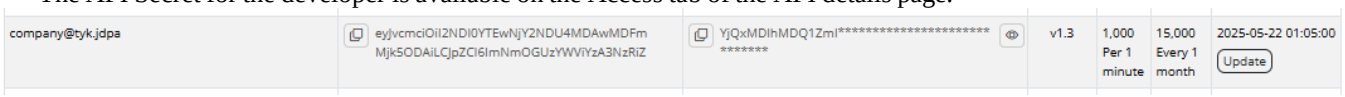
```
Authtoken: Signature
           keyId="eyJvcmciOiI2NDI0YTEwNjY2NDU4MDAwMDFmMjk5ODAiLCJpZCI6Ijk3Y2Q0NmE0ZmI1NTQ4Yjk5YzIxYmUxMzgzODg
           wMDg0IiwiaCI6Im11cm11cjEyOCJ9",algorithm="hmac-sha1",signature="nY9NLuyne8IGbo4KHTcIj9DRpi8%3D"

Date: Thu, 15 May 2025 17:40:21 GMT
```

Using Security in the Portal

When you are using the test client in the Portal, the Portal will generate the security token and set it in the header automatically.

Setting the Key ID

The API Secret for the developer is available on the Access tab of the API details page.



The Key apiID/APISecret are static and can be set as a constant in the developer's code base.

Setting the Date

Dates must be created in UTC format as follows “Thu, 15 May 2025 17:40:21 GMT”. It's important that the timestamp in the message is accurate; if the timestamp is off, the message might be rejected. The same timestamp value sent in the message header is also used in creating the Signature Digest.

A JavaScript example of generating the timestamp is shown below:

```
date = (new Date()).toUTCString()
```

Setting the Signature Digest

The secret digest is the complex value to build in this security algorithm. It must be rebuilt each time you call the service, and uses the following algorithm in pre-request script in postman:

signatureContentString = 'date: ' + $date

signatureString = CryptoJS.HmacSHA1(signatureContentString, apiSecret).**toString**(CryptoJS.enc.Base64) Where:

apiSecret is a secret key which you can obtain from the Access tab of the API Details page in the Portal. SHA1 is a secure hashing algorithm

Base64 is a byte encoding algorithm.

Developer Steps for Creating the Signature Digest

Follow these steps to build the signature digest:

1. Generate signature content string by creating the date string – “date:” + $date
2. Hash the result using the HMAC SHA-1 secure hashing algorithm using the API Secret.
3. Encode the hashed value using the Base64 encoding scheme.
4. URL-encode the result. For example, you could use the encodeURIComponent() JavaScript function
5. Assign the result as the value for the Signature parameter.
6. Include this parameter in the HTTP Authorization message header.

Understanding the SHA-1 Secure Hash Algorithm

SHA-1 is a cryptographic hash function, broadly used and trusted.

When you hash a value with SHA-1, the hash function returns a 160-bit string. This is the secret digest. The value is hashed and sent with the message; at the receipt point, the value is hashed again, and the two hash values are

compared. When the two hash values match, it is a secure, reliable indication that the message hasn't changed; the message at the receipt point is an accurate duplication of the message at the send point.

Postman Example: Creating a Shared Signature Digest

Below is an example of how to create the entire authorization header in Postman using JavaScript.

1. Place the following JavaScript in the "Pre-request Script":

```
const apiKey = pm.environment.get('tyk.api_id');
const apiSecret = pm.environment.get('tyk.api_secret');
const date = new Date().toUTCString();
const signatureContentString = `date: ${date}`;
const signature = CryptoJS.HmacSHA1(signatureContentString, apiSecret);
const signatureBase64 = CryptoJS.enc.Base64.stringify(signature);
const encodedSignature = encodeURIComponent(signatureBase64);
const Authtoken = `Signature keyId="${apiKey}",algorithm="hmac-sha1",signature="${encodedSignature}"`;
pm.environment.set("Authtoken", Authtoken);
pm.environment.set("date", date);
```

2. Create the following Environment:

	Variable	Type	Initial value	Current value
☑	tyk.api_secret	default	ODgxMGI2MT...	ZDRjM2JjMzYwZDdkNGQ3MjgwNWE4N2Q5NTMyOGYxOGE=
☑	tyk.api_id	default	eyJvcmciOiI2...	eyJvcmciOiI2NDI0YTEwNjY2NDU4MDAwMDFmMjk5ODAiLCJpZCI6Ijk3Y2Q0NmE0ZmI1NTQ4Yjk5YzIxYmUxMzgzODgwM...
☑	date	default	Thu, 20 Apr 2...	Thu, 15 May 2025 18:54:49 GMT
☑	baseURL	default	https://p-api-t...	https://apiqa.autodatacorp.org
☑	Authtoken	default		Signature keyId="eyJvcmciOiI2NDI0YTEwNjY2NDU4MDAwMDFmMjk5ODAiLCJpZCI6Ijk3Y2Q0NmE0ZmI1NTQ4Yjk5YzIxY...

3. In your request, specify the following headers (assuming the API you are calling has a request and response of JSON):

Params Authorization Headers (14) Body Scripts Settings Cookies

Headers 9 hidden

	Key	Value	Description	Bulk Edit Presets
☑	Accept	application/json		
☑	Content-Type	application/json		
☑	Authtoken	{{Authtoken}}		
☑	Date	{{date}}		

4. Specify the endpoint URL with the appropriate operation (GET, PUT, POST, …). This can be found in the test client page in the customer portal.

5. Place the body of the request in the body using either form data or raw format.

JAVA Script Example : Creating a Shared Signature Digest

```
const crypto = require('crypto');
// Define the key and shared secret
const apiKey =
'eyJvcmciOiI2NDI0YTEwNjY2NDU4MDAwMDFmMjk5ODAiLCJpZCI6Ijk3Y2Q0NmE0ZmI1NTQ4Yjk5YzIxYmUxMzgzODgwMDg0IiwiaCI
6Im11cm11cjEyOCJ9'; // Replace with your API key
const apiSecret = 'ZDRjM2JjMzYwZDdkNGQ3MjgwNWE4N2Q5NTMyOGYxOGE='; // Replace with your API secret

// Generate the message to sign
const date = new Date().toUTCString();
const signatureContentString = `date: ${date}`; // Message to sign

// Function to generate HMAC signature
function generateHmacSignature(apiSecret, message) {
    const hmac = crypto.createHmac('sha1', apiSecret);
    hmac.update(message);
    return hmac.digest('base64');
}

const hmacSignature = generateHmacSignature(apiSecret, signatureContentString);
console.log('HMAC Signature:', hmacSignature);
console.log('Date:', date);

// Additional lines
const signature = crypto.createHmac('sha1', apiSecret).update(signatureContentString).digest();
const signatureBase64 = signature.toString('base64');
const encodedSignature = encodeURIComponent(signatureBase64);

const Authtoken = `Signature keyId="${apiKey}",algorithm="hmac-sha1",signature="${encodedSignature}"`;
console.log('Authtoken:', Authtoken);
```

PHP Example: Creating a Shared Signature Digest

```
<?php
// Assuming environment variables are set in PHP environment. Alternatively, you can use a config
file or hardcode the values.

$apiKey = getenv('tyk.api_key');
$apiSecret = getenv('tyk.api_secret');

// Get the current date in UTC
$date = gmdate('D, d M Y H:i:s \G\M\T');

// Create the signature content string
$signatureContentString = 'date: ' . $date;

// Compute HMAC-SHA1 signature
$signatureString = base64_encode(hash_hmac('sha1', $signatureContentString, $apiSecret, true));

// Create the authorization header
$authHeader = 'Signature keyId="' . $apiKey . '",algorithm="hmac-sha1",signature="' .
urlencode($signatureString) . '"';

// HTTP Headers
$headers = array(
                "Accept:
                application/json"
                ,      "Authtoken:
                {$authHeader}",
                "Date: $date
                );
// Use the headers generated in a curl call to the
service. url_setopt($curl, CURLOPT_HEADER, 1);
curl_setopt($curl, CURLOPT_HTTPHEADER , $headers);
?>
```

JAVA Example: Creating a Shared Signature Digest

```
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import
java.security.NoSuchAlgorithmException;
import java.security.spec.KeySpec;
import java.util.Base64;
import javax.crypto.Mac;
import
javax.crypto.spec.SecretKeySpec;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;

public class SignatureGenerator {

    public static void main(String[]
        args) { try {
            // Replace these with your actual method to fetch environment variables
            String apiKey = System.getenv("tyk.api_key");
            String apiSecret = System.getenv("tyk.api_secret");

            // Get the current date in
            UTC String date =
            getCurrentUtcDate();

            // Create the signature content string
            String signatureContentString = "date: " + date;

            // Compute the HMAC-SHA1 signature
            String signatureString = computeHmacSha1(signatureContentString, apiSecret);

            // Create the authorization
            header String authHeader =
            String.format(
                "Signature  keyId=\"%s\",algorithm=\"hmac-sha1\",signature=\"%s\"",
                apiKey,  java.net.URLEncoder.encode(signatureString,  StandardCharsets.UTF_8.toString())
            );


            // Optionally print or use these variables as
            needed System.out.println("Authtoken: " +
            authHeader); System.out.println("Date: " +
            date);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static String getCurrentUtcDate() {
        SimpleDateFormat sdf = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss 'GMT'");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        return sdf.format(new Date());
    }

    private static String computeHmacSha1(String data, String key) throws NoSuchAlgorithmException,
java.security.InvalidKeyException {
        Mac mac = Mac.getInstance("HmacSHA1");
        SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(StandardCharsets.UTF_8), "HmacSHA1");
        mac.init(secretKeySpec);
        byte[] rawHmac = mac.doFinal(data.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(rawHmac);
    }
}
```

C# Example: Creating a Shared Signature Digest

```
// See https://aka.ms/new-console-template for more information
using System;
using System.Text;
using System.Security.Cryptography;
using System.Web;
using System.Net;
using static System.Runtime.InteropServices.JavaScript.JSType;

class SignatureGenerator
{
    static void Main()
    {
        try
        {
            // Fetch API key and secret from environment variables
            string apiKey = "your-apikey-here";
            string apiSecret = "your_shared_secret_here";

            // Get the current UTC date
            string date = GetCurrentUtcDate();

            // Create the signature content string
            string signatureContentString = "date: " + date;

            // Compute the HMAC-SHA1 signature
            string signatureString = ComputeHmacSha1(signatureContentString, apiSecret);

            //URL - encode the signature
            string encodedSignature = WebUtility.UrlEncode(signatureString);

            // Construct the Authorization header
          string authHeader=$"SignaturekeyId=\"{apiKey}\",algorithm=\"hmac-sha1\",signature
=\"{encodedSignature}\"";


  // Print the results
            Console.WriteLine("Authtoken: " + authHeader);
            Console.WriteLine("Date: " + date);

        }
        catch (Exception e)
        {
            Console.WriteLine("Error: " + e.Message);
        }
    }

    // Get current date in UTC (RFC1123 format)
    private static string GetCurrentUtcDate()
    {
        return DateTime.UtcNow.ToString("R"); // Same as "ddd, dd MMM yyyy HH:mm:ss 'GMT'"
    }

    // Compute HMAC-SHA1 and return Base64-encoded result
    private static string ComputeHmacSha1(string data, string key)
    {
        byte[] keyBytes = Encoding.UTF8.GetBytes(key);
        byte[] dataBytes = Encoding.UTF8.GetBytes(data);

        using (var hmac = new HMACSHA1(keyBytes))
        {
            byte[] hash = hmac.ComputeHash(dataBytes);
            string base64Signature = Convert.ToBase64String(hash);
            return base64Signature;
        }
    }
}
```

Python Example: Creating a Shared Signature Digest

```
import hmac
import hashlib
import base64
import urllib.parse
from datetime import datetime

# Simulated environment variables (replace with actual secrets or env fetch)
api_key = "your-api-keyid-here"
api_secret = "your-shared-secret-here="

# ☑ Step 1: Format the date like "Tue, 22 Apr 2025 14:21:48 GMT"
date = datetime.utcnow().strftime('%a, %d %b %Y %H:%M:%S GMT')

# ☑ Step 2: Create the signing string
signature_content_string = f"date: {date}"

# ☑ Step 3: Compute the HMAC-SHA1 digest and base64 encode
hmac_digest = hmac.new(
    api_secret.encode('utf-8'),
    msg=signature_content_string.encode('utf-8'),
    digestmod=hashlib.sha1
).digest()

signature_base64 = base64.b64encode(hmac_digest).decode('utf-8')

# ☑ Step 4: URL encode the Base64 signature
signature_encoded = urllib.parse.quote(signature_base64, safe='')

# ☑ Step 5: Build the Authorization header
auth_header = f'Signature keyId="{api_key}",algorithm="hmac-sha1",signature="{signature_encoded}"'

# ✅ Step 6: Output as if setting environment vars
print("Authtoken:", auth_header)
print("Date:", date)
```

VBA HMAC-SHA1 Signature Generator for API Authentication
Version: 1.1 (Production Ready)

REQUIREMENTS:

- Microsoft Excel or any Office application with VBA
- PowerShell (pre-installed on Windows 7+)

SETUP:

1. Press Alt+F11 to open VBA Editor
2. Insert > Module
3. Paste this entire code
4. Update API credentials in the function below or use environment variables

USAGE:

- Call GenerateSignature() to generate and display the signature
- Or use GetAuthHeaders() to get both Authtoken and Date for your API calls

```
Option Explicit

' Main function - Generates signature and displays it
Public Sub GenerateSignature()
    Dim headers As Variant

    ' Generate headers
    headers = GetAuthHeaders()

    ' Display results
    Debug.Print "Authtoken: " & headers(0)
    Debug.Print "Date: " & headers(1)

    MsgBox "Authtoken: " & headers(0) & vbCrLf & vbCrLf & _
           "Date: " & headers(1), vbInformation, "Signature Generated"
End Sub



' Returns array with Authtoken and Date headers for API calls
' Usage: headers = GetAuthHeaders()
'        Authtoken = headers(0)
'        Date = headers(1)
Public Function GetAuthHeaders() As Variant
    Dim apiKey As String
    Dim apiSecret As String
    Dim currentDate As String
    Dim signatureString As String
    Dim authToken As String

    ' Get API credentials from environment variables
    apiKey = Environ("tyk.api_key")
    apiSecret = Environ("tyk.api_secret")

    ' Validate credentials are set
    If apiKey = "" Or apiSecret = "" Then
        MsgBox "Error: API credentials not found!" & vbCrLf & vbCrLf & _
               "Please set environment variables:" & vbCrLf & _
               "  tyk.api_key" & vbCrLf & _
               "  tyk.api_secret", vbCritical, "Missing Credentials"
        Err.Raise vbObjectError + 1, "GetAuthHeaders", "API credentials not configured"
    End If

    ' Generate current UTC date
    currentDate = GetUtcDateString()

    ' Compute HMAC-SHA1 signature
    signatureString = ComputeHmacSha1("date: " & currentDate, apiSecret)

    ' Create Authtoken header
    authToken = "Signature keyId=""" & apiKey & """,algorithm=""hmac-sha1"",signature=""" &
UrlEncode(signatureString) & """"

    ' Return array: [Authtoken, Date]
    GetAuthHeaders = Array(authToken, currentDate)
End Function



' Generate UTC date string in RFC 1123 format (matches JavaScript toUTCString)
Private Function GetUtcDateString() As String
    Dim utcDate As Date

    ' Get current time and convert to UTC
    utcDate = Now()
    utcDate = DateAdd("h", -GetTimezoneOffsetHours(), utcDate)

    ' Format: "Tue, 07 Jan 2026 14:30:45 GMT"
    GetUtcDateString = Format(utcDate, "ddd, dd mmm yyyy hh:nn:ss") & " GMT"
End Function
```

```
' Compute HMAC-SHA1 using PowerShell
Private Function ComputeHmacSha1(ByVal data As String, ByVal key As String) As String
    Dim shell As Object, exec As Object, cmd As String
    data = Replace(data, "'", "''")
    key = Replace(key, "'", "''")
    cmd = "powershell.exe -NoProfile -Command ""$h=New-Object
System.Security.Cryptography.HMACSHA1;$h.key=[Text.Encoding]::UTF8.GetBytes('" & key &
"');[Convert]::ToBase64String($h.ComputeHash([Text.Encoding]::UTF8.GetBytes('" & data & "')))"""
    Set shell = CreateObject("WScript.Shell")
    Set exec = shell.exec(cmd)
    Do While exec.Status = 0: DoEvents: Loop
    ComputeHmacSha1 = Replace(Replace(Trim(exec.StdOut.ReadAll), vbCrLf, ""), vbLf, "")
End Function



' URL encode a string (RFC 3986)
Private Function UrlEncode(ByVal str As String) As String
    Dim i As Integer
    Dim char As String
    Dim asciiVal As Integer
    Dim result As String

    For i = 1 To Len(str)
        char = Mid(str, i, 1)
        asciiVal = Asc(char)

        If (asciiVal >= 48 And asciiVal <= 57) Or _
           (asciiVal >= 65 And asciiVal <= 90) Or _
           (asciiVal >= 97 And asciiVal <= 122) Or _
           char = "-" Or char = "_" Or char = "." Or char = "~" Then
            result = result & char
        Else
            result = result & "%" & Right("0" & Hex(asciiVal), 2)
        End If
    Next i

    UrlEncode = result
End Function



' Get timezone offset in hours
Private Function GetTimezoneOffsetHours() As Double
    Dim wmi As Object
    Dim tz As Object

    On Error Resume Next
    Set wmi = GetObject("winmgmts:\\.\root\cimv2")
    Set tz = wmi.ExecQuery("Select * from Win32_TimeZone").ItemIndex(0)

    If Not tz Is Nothing Then
        GetTimezoneOffsetHours = tz.Bias / 60
    Else
        GetTimezoneOffsetHours = 0
    End If

    On Error GoTo 0
End Function
```