Abstract

This document provides a step-by-step guide on how to retrieve the history of a document using its SHA-256 hash. The process involves three main steps: generating the SHA-256 hash of the document, obtaining the document ID from the hash, and retrieving the document history using the document ID. Additionally, it is essential to have an API key, which can be generated in the admin website under “account settings”. This API key must be included as an HTTP header in each request to authenticate and authorize access to the API.

General Process

Step 1: Generate the SHA-256 Hash of the Document

First, we need to calculate the SHA-256 hash of the document. This hash uniquely represents the contents of the document. Here's how you can do it:

- Open the document file.
- Read the contents of the file.
- Use a SHA-256 hashing algorithm to compute the hash.
- Convert the hash bytes to a hexadecimal string.

Step 2: Get the Document ID from the API using the document’s Hash

Next, we'll use the generated SHA-256 hash to get the document ID. This involves making an API call:

- Construct the API URL with the hash as a parameter.
- Include the necessary authorization header.
- Make an HTTP GET request to the API.
- Parse the JSON response to extract the Id field, which is the document ID.

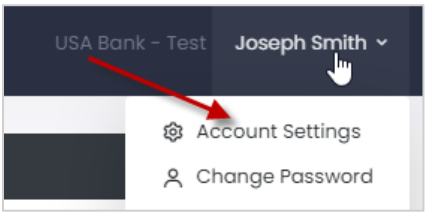
Step 3: Retrieve the Document History Using the Document ID

Finally, we'll use the document ID to retrieve the document's history:

- Construct the API URL for the document history using the document ID.
- Include the necessary authorization header.
- Make an HTTP GET request to the API.
- Process the JSON response, which contains the document history.

Authorization

Log in to the Universal STAMP Web (https://app.universalstamp.com) as an admin user.
At the top right corner, click on your name, then “Account Settings”



Scroll down to the API Settings and select "Generate new API key". Ensure that you do not generate a new key if an active key is already in use, as generating a new key will delete the existing one. After confirming the creation of a new key, you may copy and paste it into your code. This API key needs to be sent in each http request as a header like so: `"Authorization":"{{API Key}}"`

Code Examples

Windows Power Shell

Create a script called “GetDocumentHistory.ps1” and launch it from the PowerShell prompt using
`.\GetDocumentHistory.ps1  -filePath "C:\temp\Document.pdf"`

```
param (
    [string]$filePath
)

# Calculate SHA-256 hash of the document
$sha256 = [System.Security.Cryptography.SHA256]::Create()
$fileStream = [System.IO.File]::Open($filePath, [System.IO.FileMode]::Open,
[System.IO.FileAccess]::Read)
$hashBytes = $sha256.ComputeHash($fileStream)
$fileStream.Close()

# Convert hash bytes to a hexadecimal string
$hashString = -join ($hashBytes | ForEach-Object { $_.ToString("x2") })

# Define the API URL and authorization header
$apiUrl = "https://app.universalstamp.com/api/v2"
$apiKey = "{{ Your API Key here }}"

# Ensure the URL is correctly formatted using the -f operator
$fullUrl = "{0}/document/{1}" -f $apiUrl, $hashString

try {
    # Use Invoke-WebRequest to call the REST API with the hash parameter and authorization header
    $response = Invoke-WebRequest -Uri $fullUrl -Method Get -Headers @{ "Authorization" = $apiKey
}

    # Check the status code and handle the response
    if ($response.StatusCode -eq 200) {
        # Parse the JSON response to extract the Id field
        $jsonResponse = $response.Content | ConvertFrom-Json
        $documentId = $jsonResponse.Id

        # Define the URL for the document history
        $historyUrl =  "{0}/document/{1}/history" -f $apiUrl, $documentId

        # Make a subsequent call to retrieve the document history
        $historyResponse = Invoke-WebRequest -Uri $historyUrl -Method Get -Headers @{
"Authorization" = $apiKey }

        # Display the document history in a formatted way
        $formattedJson = $historyResponse.Content | ConvertFrom-Json | ConvertTo-Json -Depth 10
        Write-Output $formattedJson
    } else {
        Write-Output "HTTP Status Code: $($response.StatusCode)"
    }
} catch {
    # Handle the exception and print the status code if available
    if ($_.Exception.Response -ne $null) {
        $statusCode = $_.Exception.Response.StatusCode
        Write-Output "HTTP Status Code: $statusCode"
    } else {
        Write-Output "An error occurred: $($_.Exception.Message)"
    }
}
```

Microsoft .net C#

Create a console App and paste this code into the Program.cs file. Then run the program.

```
using System;
using System.IO;
using System.Net.Http;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

class Program
{
    static async Task Main(string[] args)
    {
        const string filePath = args[0] ;
        const string apiUrl = "https://app.universalstamp.com/api/v2/";
        const string apiKey = "{{ Your API Key here }}";

        try
        {
            // Calculate SHA-256 hash of the document
            string hashString = CalculateSHA256(filePath);

            using (HttpClient client = new HttpClient())
            {
                client.DefaultRequestHeaders.Add("Authorization", apiKey);
                client.BaseAddress = new Uri(apiUrl);

                // Use HttpClient to call the REST API with the hash parameter and authorization
header
                HttpResponseMessage response = await client.GetAsync($"document/{hashString}");

                // Check the status code and handle the response
                if (response.IsSuccessStatusCode)
                {
                    string jsonResponse = await response.Content.ReadAsStringAsync();
                    dynamic document =
Newtonsoft.Json.JsonConvert.DeserializeObject(jsonResponse);
                    string documentId = document.Id;

                    // Make a subsequent call to retrieve the document history
                    HttpResponseMessage historyResponse = await
client.GetAsync($"document/{documentId}/history");

                    // display document history
                    if (historyResponse.IsSuccessStatusCode)
                    {
                        string history = await historyResponse.Content.ReadAsStringAsync();
                        Console.WriteLine(history);
                    }
                    else
                    {
                        Console.WriteLine($"HTTP Status Code: {historyResponse.StatusCode}");
                    }
                }
                else
                {
                    Console.WriteLine($"HTTP Status Code: {response.StatusCode}");
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"An error occurred: {ex.Message}");
        }
    }
```

```
static string CalculateSHA256(string filePath)
    {
        using (FileStream stream = File.OpenRead(filePath))
        {
            using (SHA256 sha256 = SHA256.Create())
            {
                byte[] hashBytes = sha256.ComputeHash(stream);
                StringBuilder hashString = new StringBuilder();
                foreach (byte b in hashBytes)
                {
                    hashString.Append(b.ToString("x2"));
                }
                return hashString.ToString();
            }
        }
    }
}
```

Full API documentation

Base URL

```
https://app.universalstamp.com/api/v2
```

Endpoints

Documents

```
URL: /document/{id}
Method: GET
Description: Retrieves a document's metadata by its numeric ID. The authenticated account must be
the Guarantor or the assigned Transfer Agent.
Parameters:
        - id (int): The numeric ID of the document.
Responses:
        - 200 OK: Returns the document's metadata.
        - 401 Unauthorized: If the API token is invalid or the account is not the Guarantor or
assigned Transfer Agent.
        - 404 Not Found: If the document is not found.
        - 500 Internal Server Error: If an error occurs on the server.
```

Document Model

```
{
  "Id": "int",
  "Name": "string",
  "AccountId": "int",
  "CreationDate": "string (ISO 8601 format)",
  "ForwardedDate": "string (ISO 8601 format)",
  "Verified": "bool",
  "Rejected": "bool"
}
```

Get Document by Hash

```
URL: /document/{hash}
Method: GET
Description: Retrieves a document's metadata by its SHA-256 hash value. No account ownership
check is performed; possession of the hash is sufficient for access.
Parameters:
        - hash (string): The hexadecimal SHA-256 hash of the document file.
Responses:
        - 200 OK: Returns the document's metadata. See Document Model above.
        - 404 Not Found: If the document is not found.
        - 500 Internal Server Error: If an error occurs on the server.
```

Get Document Audit Trail

```
URL: /document/{id}/history
Method: GET
Description: Retrieves the full audit trail for a document by its numeric ID. Access is not
restricted by account ownership.
Parameters:
        - id (int): The numeric ID of the document.
Responses:
        - 200 OK: Returns the document's audit trail.
        - 404 Not Found: If the document is not found.
        - 500 Internal Server Error: If an error occurs on the server.
```

Document History Model

```
{
  "Id": "int",
  "Name": "string",
  "History": [
    {
      "Timestamp": "string (ISO 8601 format)",
      "Text": "string"
    }
  ]
}
```

Add Document History Event

```
URL: /document/{id}/history/{text}
Method: PUT
Description: Appends a custom history event to a document's audit trail. The authenticated
account must be the Guarantor or the assigned Transfer Agent.
Parameters:
        - id (int): The numeric ID of the document.
        - text (string): The text of the history event to append.
Responses:
        - 200 OK: Returns the updated audit trail. See Document History Model above.
        - 401 Unauthorized: If the API token is invalid or the account is not authorized.
        - 404 Not Found: If the document is not found.
        - 500 Internal Server Error: If an error occurs on the server.
```

Accept Document

```
URL: /document/{id}/accept
Method: POST
Description: Marks a document as accepted (e-delivered). A notification e-mail is sent to the
configured recipient.
Parameters:
        - id (int): The numeric ID of the document to accept.
Responses:
        - 200 OK: The document was accepted successfully.
        - 400 Bad Request: If no document ID was provided.
        - 404 Not Found: If the document or account is not found.
        - 500 Internal Server Error: If an error occurs on the server.
```

Reject Document

```
URL: /document/{id}/reject/{note?}
Method: POST
Description: Marks a document as rejected. A rejection notification e-mail is sent to the
configured recipient. If a note is supplied, it is included in the notification.
Parameters:
        - id (int): The numeric ID of the document to reject.
        - note (string, optional): A rejection reason to include in the notification e-mail.
Responses:
        - 200 OK: The document was rejected successfully.
        - 400 Bad Request: If no document ID was provided.
        - 404 Not Found: If the document or account is not found.
        - 500 Internal Server Error: If an error occurs on the server.
```

Get Account Details

```
URL: /account
Method: GET
Description: Retrieves the account details for the authenticated API user.
Responses:
        - 200 OK: Returns the account's details.
        - 500 Internal Server Error: If an error occurs on the server.
```

Account Model

```
{
  "Id": "int",
  "Name": "string",
  "Active": "bool",
  "NotificationEMailAddress": "string",
  "Type": "int"
}
```

Get User by ID

```
URL: /user/{id}
Method: GET
Description: Retrieves a user's details by numeric ID. The user must belong to the authenticated
account.
Parameters:
        - id (int): The numeric ID of the user.
Responses:
        - 200 OK: Returns the user's details.
        - 401 Unauthorized: If the API token is invalid or the account is not authorized.
        - 404 Not Found: If the user is not found.
        - 500 Internal Server Error: If an error occurs on the server.
```

User Model

```
{
  "Id": "int",
  "UserName": "string",
  "FullName": "string",
  "Email": "string",
  "Active": "bool",
  "LastSignIn": "string (ISO 8601 format)",
  "CustomFields": {
    "CustomField1": "string",
    "CustomField2": "string",
    "CustomField3": "string"
  },
  "Roles": ["string"]
}
```

Get All Users

```
URL: /users
Method: GET
Description: Retrieves all users belonging to the authenticated account.
Responses:
        - 200 OK: Returns all users in the authenticated account.
        - 500 Internal Server Error: If an error occurs on the server.
```

User List Model

```
[
  {
    "Id": "int",
    "UserName": "string",
    "FullName": "string",
    "Email": "string",
    "Active": "bool"
  }
]
```

Search Users

```
URL: /users/search
Method: GET
Description: Searches for users within the authenticated account by e-mail and/or full name. At
least one query parameter must be provided.
Parameters:
        - email (string, optional): E-mail address to search for. Exact, case-insensitive match.
        - fullName (string, optional): Full name to search for. Partial, case-sensitive match.
Responses:
        - 200 OK: Returns matching users. See User List Model above.
        - 400 Bad Request: If neither email nor fullName is provided.
        - 404 Not Found: If no matching users are found.
        - 500 Internal Server Error: If an error occurs on the server.
```

Get Stamps

```
URL: /stamps
Method: GET
Description: Retrieves all stamps belonging to the authenticated account.
Responses:
        - 200 OK: Returns all stamps in the authenticated account.
        - 500 Internal Server Error: If an error occurs on the server.
```

Stamp Model

```
[
  {
    "Id": "int",
    "Name": "string",
    "Description": "string",
    "FinancialInstituteNumber": "string",
    "LocationNumber": "string",
    "Active": "bool"
  }
]
```

Deactivate Stamp

```
URL: /stamps/deactivate/{id}
Method: PUT
Description: Deactivates a stamp by its numeric ID. The stamp must belong to the authenticated
account. A history event is recorded on the stamp.
Parameters:
        - id (int): The numeric ID of the stamp to deactivate.
Responses:
        - 200 OK: Returns the updated stamp record. See Stamp Model above.
        - 404 Not Found: If the stamp is not found or does not belong to
          the authenticated account.
        - 500 Internal Server Error: If an error occurs on the server.
```