DIGITAL CAB

DIGITAL CAB REST API DOCUMENTATION

TABLE OF CONTENTS

AuthHeaderS 2

API-KEY 2

AES-Ticket 2

ReadMessage 3

MessageReceivedAck 5

ReadReceptionLog 6

ReadTransmissionLog 7

SendMessage 8

Generate AES-Ticket 9

Additional Information 10

The base URL for DIGITALCAB's REST API is stated below.

API URL: https://login.digitalcab.dk/restapi/

AUTHHEADERS

When making API calls to DIGITAL CAB, you need to make use of API key in the header.
We have 2 types of authentication:

1. API-KEY
2. AES-Ticket

API-KEY

When using the API-key, the header should contain the following.

```
"APIKEY": [Your APIKEY]
```

AES-TICKET

When using the AES-ticket the header should contain the following.

```
"Ticket": [Your AES-Ticket]
```

Here is a C# example for adding one of the headers:

```
var baseAdress = "https://login.digitalcab.dk/restapi/";
var requestUri  = "readmessage";

using (var client = new HttpClient())
{
    client.BaseAddress = new Uri(baseAdress);
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
    //insert API key or AES ticket:
    client.DefaultRequestHeaders.Add("APIKey", "Your APIKEY");
    client.DefaultRequestHeaders.Add("Ticket", "Your AES-Ticket");
}
```

```
var baseAdress = "https://login.digitalcab.dk/restapi/";
var requestUri = "readmessage";


using (var client = new HttpClient())
{
    Console.WriteLine("Hello World!");
    client.BaseAddress = new Uri(baseAdress);
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
    //insert API key or AES ticket
    client.DefaultRequestHeaders.Add("APIKey", "API-KEY");
    client.DefaultRequestHeaders.Add("Ticket", "AES-Ticket");
}
```

READMESSAGE

This API endpoint is used to read messages which have the status “Klar til Webservice”. It’s important that you remember to send an acknowledgement using the API call MessageReceivedAck with the message ID as parameter after you have called ReadMessage. If you forget to do this, you will keep receiving the same message every time. If there are no more messages left, the endpoint will return a Json object containing the id “0”.

URL: /readmessage
Type of call: GET
Request Parameters:

- receiver (Optional).
- DocType (Optional).
- extractembeddedattachments (Optional), (Default = False).

Response:

```
{[
"id": Int,
"sender": "",
"receiver": "",
"docNumber": "",
"reference": "",
"docType": "",
"message": "",
"filename": "",
"attachments": []
]}
```

Example requests:

`/readmessage?receiver=5790001394486&DocType=OIOUBL`

`/readmessage?receiver=5790001394486&extractembeddedattachments=True`

Please note: This API call is limited to 10 calls within 10 minutes per combination of user account and receiver, unless there is a document to retrieve in which case the counter is reset.

Here is a C# example for reading all messages:

```
ReadAllMessage()

        {
            //AcknowledgeReceivedButton.Enabled = false;
            var baseAddress = "https://login.digitalcab.dk/restapi/";
            var requestUri = "readmessage";
            bool moreDocs = true;

            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri(baseAdress);
                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
                client.DefaultRequestHeaders.Add("APIKey", "Your APIKEY");
                while (moreDocs)
                {
                    var response = await client.GetAsync(requestUri);
                    Console.WriteLine(response.StatusCode);
                    if (response.IsSuccessStatusCode)
                    {
                        var readMessage = await response.Content.ReadAsAsync<ReadMessageResponse>();
                        ReadMessageResponse MessageReceived = readMessage;
                        if (MessageReceived.id <= 0)
                        {
                            moreDocs = false;
                            break;
                        }
                        await AcknowledgeMessage(MessageReceived.id);
                    }
                }
            }

        }
```

MESSAGERECEIVEDACK

This API endpoint is used to send an acknowledgement. This endpoint is only used when you have received a message. The body of this post call should be the ID of the message one has received.

URL: /readmessage/ack

Type of call: POST

Required Parameters:

- AuthHeader (AuthHeader)
- Id(int) - *This is the ID of the received message.*

Here is a C# example code for making this call:

```
private async Task AcknowledgeMessage(int id)
{
    var baseAdress = "https://login.digitalcab.dk/restapi/";
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri(baseAdress);
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
        client.DefaultRequestHeaders.Add("APIKey", "Your API-KEY");
        HttpContent content = new StringContent(
                            id.ToString(),
                            Encoding.UTF8,
                            "application/json"
                        );
        HttpResponseMessage response = await client.PostAsync("readmessage/ack", content);
    }

}
```

It is necessary to acknowledge the received message otherwise the next ReadMessage call will return the same message again.

Please note: This API call is limited to 10 calls within 10 minutes but is reset every time a message has been received with ReadMessage.

READRECEPTIONLOG

This API endpoint is used to get the ReceptionLog.

URL: /readreceptionlog
Type of call: GET
Required Parameters:

- AuthHeader (AuthHeader)

Here is a C# example code for making this call

```
public async Task GetReceptionLog()
{
    var baseAdress = "https://login.digitalcab.dk/restapi/";
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri(baseAdress);
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
        client.DefaultRequestHeaders.Add("APIKey", "Your API-Key");
        var response = await client.GetAsync("readreceptionlog");
        if (response.IsSuccessStatusCode)
        {
            var receptionLog = await response.Content.ReadAsAsync<IEnumerable<LogResponse>>();
        }

    }

}
```

Returns a list of received documents. You will get information about the latest 2.500 documents.

Please note: This API call is limited to 10 calls within 10 minutes

READTRANSMISSIONLOG

This API endpoint is used to get the TransmissionLog.

URL: /readtransmissionlog
Type of call: GET
Required Parameters:

- AuthHeader (AuthHeader)

Here is a C# example code for making this call.

```
public async Task GetTransmissonLog()
{
    var baseAdress = "https://login.digitalcab.dk/restapi/";
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri(baseAdress);
        client.DefaultRequestHeaders.Accept.Clear();
        client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
        client.DefaultRequestHeaders.Add("APIKey", "Your API-Key");
        var response = await client.GetAsync("readtransmissionlog");
        if (response.IsSuccessStatusCode)
        {
            var transmissionLog = await response.Content.ReadAsAsync<IEnumerable<LogResponse>>();
        }

    }

}
```

Returns a list of sent documents. You will get information about the latest 2.500 documents.

Please note: This API call is limited to 10 calls within 10 minutes

SENDMESSAGE

This API endpoint is used to send a message/document. Only call the endpoint when a message is ready to be sent.

URL: /sendmessage
Type of call: POST
Required Parameters:

- AuthHeader (AuthHeader)
- Message
- reference (String) - *can be empty string.*
- communicationid (int) - *should be 0 by default. Only change this to another value if you have an agreement with DIGITAL CAB.*
- Attachment - *consist of two attributes "Data" (byte) and Filename (String)*

Example of the POST body:

```
{
   "Message":"your message",
   "Filename":"dokument.txt",
   "Reference":"your reference",
   "CommunicationId":0,
   "Attachment":{
      "Data":null,
      "Filename":null

    }
}
```

Here is a C# example code for making this call.

```
public async Task SendMessage()
        {
            var baseAdress = "https://login.digitalcab.dk/restapi/";
            using (var client = new HttpClient())
            {
                client.BaseAddress = new Uri(baseAdress);
                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
                client.DefaultRequestHeaders.Add("APIKey", "Your API-Key");
                String path2Document = "Path to file";
                var msg = new SendMessageRequest();
                msg.Message = File.ReadAllText(path2Document);
                msg.CommunicationId = 0;
                msg.Reference = "your reference";
                msg.Filename = Path.GetFileName(path2Document);
                var attch = new Attachment();
                msg.Attachment = attch;
                HttpContent content = new StringContent(
                                JsonConvert.SerializeObject(msg),
                                Encoding.UTF8,
                                "application/json"
                            );
                HttpResponseMessage response = await client.PostAsync("sendmessage", content);

                if (response.IsSuccessStatusCode)
                    Console.WriteLine("Transmission success");
                else
                    Console.WriteLine("Transmission failed");
            }

        }
```

GENERATE AES-TICKET

This section contains code snippets, which can help one create the AES-ticket.

The code below is a method which can be used to generate the AES-Ticket:

```
public static string MakeTicket(string Aes256Key_Base64)
        {
            var payload = $"{ DateTime.Today.ToString("dd-MM-yyyy") }:{ Guid.NewGuid() }";
            var key = Convert.FromBase64String(Aes256Key_Base64);
            var encr = new Encryption(key);
            var encrypted = encr.Encrypt(payload);

            return encrypted;
        }
```

The code below is a class which the method above uses to generate the AES-Ticket.

```
public class Encryption : IDisposable
{
    /// <summary>
    ///     Initialization vector length in bytes.
    /// </summary>
    private const int IvBytes = 16;
    private byte[] iv = new byte[] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

    private readonly UTF8Encoding _encoder;
    private readonly ICryptoTransform _encryptor;
    private readonly AesManaged _aes;

    /// <summary>
    ///     Key must be exactly 32 bytes long.
    /// </summary>)
    public Encryption(byte[] key)
    {
        _aes = new AesManaged { Key = key, Mode = CipherMode.ECB, Padding = PaddingMode.PKCS7, IV = iv };
        _encryptor = _aes.CreateEncryptor();
        _encoder = new UTF8Encoding();
    }

    public string Decrypt(string encrypted)
    {
        return _encoder.GetString(Decrypt(Convert.FromBase64String(encrypted)));
    }

    public void Dispose()
    {
        _aes.Dispose();
        _encryptor.Dispose();
    }

    public string Encrypt(string unencrypted)
    {
        return Convert.ToBase64String(Encrypt(_encoder.GetBytes(unencrypted)));
    }

    private byte[] Decrypt(byte[] buffer)
    {
        using (ICryptoTransform decryptor = _aes.CreateDecryptor(_aes.Key, iv))
        {
            return decryptor.TransformFinalBlock(buffer, 0, buffer.Length);
        }
    }

    private byte[] Encrypt(byte[] buffer)
    {
        byte[] inputBuffer = _encryptor.TransformFinalBlock(buffer, 0, buffer.Length);
        return inputBuffer;
    }
}
```

ADDITIONAL INFORMATION

More code examples in the API demo application via the link below (requires login). If you do not have an account yet, you can sign up for a FREE account here digitalcab.dk/priser

https://login.digitalcab.dk/filer/vejledning/REST-API/API_DEMO_APP/DIGITAL_CAB_API_demo.zip