

SIEMENS



Access Control

SiPass integrated

RESTful HR API Developer Guide

MP 2.95

Copyright

Technical specifications and availability subject to change without notice.

We reserve all rights in this document and in the subject thereof. By acceptance of the document the recipient acknowledges these rights and undertakes not to publish the document nor the subject thereof in full or in part, nor to make them available to any third party without our prior express written authorization, nor to use it for any purpose other than for which it was delivered to him.

Edition: 15.09.2023

Document ID: A6V11366088

© Siemens Switzerland Ltd, 2023

Table of Contents

1	Document Updates after Previous Release	4
2	Introduction.....	5
2.1	Scope.....	5
2.2	Document References	5
2.3	Definitions, Acronyms and Abbreviations, Conventions	5
3	Installation Requirements	6
4	Object Model Overview.....	7
5	API Overview.....	8
5.1	Authentication.....	8
5.1.1	Login Procedure.....	8
5.1.2	Logout Procedure	10
5.2	Cardholder.....	13
5.2.1	Get All Cardholders.....	13
5.2.2	Get Cardholder By ID.....	16
5.2.3	Get Cardholder Image by Cardholder ID.....	18
5.2.4	Create Cardholder.....	18
5.2.5	Update Cardholder.....	25
5.2.6	Delete Cardholder.....	32
5.2.7	Get All Cardholders With Credentials And Access Privileges	34
5.3	Session Management	37
5.3.1	Consume GET API: Language C#.....	38
5.4	Frequently Asked Questions	38
6	API Documentation.....	39
7	Test Client Source Code.....	40

1 Document Updates after Previous Release

The following updates have been done to this document as below:

SiPass integrated MP 2.85

Section	Details
Cardholder	<ul style="list-style-type: none">• Date and Time Format are updated.

2 Introduction

This document helps and guides the programmer to integrate the SiPass integrated HR (Human Resource) API with any third-party application.

2.1 Scope

This document guides the developer to consume the SiPass integrated HR API.

2.2 Document References

- SiPass integrated HR API – Object Model Guide.pdf
- SiPass integrated HR API – Specification Guide.pdf
- SiPass integrated HR API – Installation Guide.pdf
- Swagger - <https://HOSTNAME:PORT/swagger/ui/index>



All these documents will be available in the **Installation Suite**.

2.3 Definitions, Acronyms and Abbreviations, Conventions

Term	Description
API	Application Programming Interface.
HOST NAME	HOST Computername or IP of where the SiPass integrated HR API is installed.
PORT	PORT that is used to host the HR API.

3 Installation Requirements



NOTICE

1. The SiPass integrated HR API is a windows based service and will be available in the system after the successful installation.
2. The connectivity between the Third party application and the HR RESTful API application are secure.
3. It is recommended to have a good network connectivity between the Third party application and the HR API, to avoid the network failures during the API request/response and message transmission.
4. When there is a connectivity failure between the Third party application and the HR RESTful API, the third party application has to reconnect with the HR API application.

4 Object Model Overview

Refer to the SiPass integrated HR API – Object Model document for more information about the object models used across the API's.

5 API Overview

The SiPass integrated HR API is a windows based service and will be available in the system after the successful installation.

- The connectivity between the Third party application and the HR RESTful API application are secure.
- It is recommended to have a good network connectivity between the Third party application and the HR API, to avoid the network failures during the API request/response and message transmission.
- When there is a connectivity failure between the Third party application and the HR RESTful API, the third party application has to reconnect with the HR API application.

The SiPass integrated HR API Service provides the APIs to perform the following operations:

- Authentication (Login and Logout)
- Cardholder (Create, Read, Update, and Delete)
- Session Management (Get Session Timeout Interval, and Renew Session)

5.1 Authentication

This API is used to authenticate (login) the users to consume the SiPass integrated HR APIs.

5.1.1 Login Procedure

To log on to the system using SiPass integrated HR API, use the following URL:
<https://HOSTNAME:PORT/api/V1/hr/Authentication>

5.1.1.1 Consume Login API: Language C#

Perform the following steps to login:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **POSTNOTICE!** .
4. Under **HEADERS**,
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **clientUniqueld** and its **value**
 - enter the **name** as **language** and **value** as **English**
5. Under **BODY** enter the **Username** and **Password** and their relevant values.
6. Send **POST** request to the server.
 - ⇒ The request is sent to the server.
7. The API service authenticates the request.
 - ⇒ On successful authentication, the API returns a unique token.
NOTICE! The unique token helps to identify the client. or...
 - ⇒ On failed authentication, the API returns an empty token.

**NOTICE**

For the subsequent requests to the API,

- The authentication token must be assigned to the Authorization header.
- The clientUniqueld (which is sent during the login request must be available to the subsequent requests of the API) must be available in the header.
- The language header must be available in the header.

```
1 {  
2   "Token": "96734DBC08A49576FBF3356A22AC35BDA1EB302E5238479D6940F258D593F4:siemens"  
3 }
```

Fig. 1: Unique token

Request

The Post Request can be sent as shown below:

```
public bool Authenticate(string userName, string password)  
{  
    AuthenticationInfo.ClearInfo();  
    SessionInfo.ClearSessionInfo();  
    var dicLoginRequest = new Dictionary<string, string> { { Authentication, "" } };  
    dicLoginRequest.Add("clientUniqueld", AuthenticationInfo.ClientUniqueld);  
    var response = _authObj.Authenticate(  
        new UserCredential { UserName = userName, Password = password },  
        dicLoginRequest);  
    var hrAuthResponse = GetConvertToHrAuthResponseType(response);  
    AuthenticationInfo.Token = hrAuthResponse.Token;  
    return (!string.IsNullOrEmpty(hrAuthResponse.Token));  
}  
  
public HttpResponseMessage PostWebRequest<T>(string resource, IReadOnlyDictionary<string,  
string> dicReqHeaders, T payload)  
{  
    try  
    {  
        var result = Task.Run(() => PostAsyncHttpRequest(resource, dicReqHeaders, payload)).Result;  
        return result;  
    }  
    catch (AggregateException ae)  
    {  
        HandleAggregateException(ae);  
    }  
    return null;  
}  
  
private async Task<HttpResponseMessage>  
PostAsyncHttpRequest<T>(string resource, IReadOnlyDictionary<string, string> dicReqHeaders, T  
payload)  
{  
    CancellationTokenSource postCancellationTokenSource = new CancellationTokenSource();  
    try
```

```

{
var cancelToken = postcancellationTokenSource.Token;
//To resolve certification issue
ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };
HttpResponseMessage response;
using (var httpClient = new HttpClient { Timeout = TimeSpan.FromSeconds(_webTimeout) })
{
AddHeaderValue(httpClient, resource);
response = await httpClient.PostAsJsonAsync(resource, payload, cancelToken);
CheckForError(response,false);
}
return response;
}
finally
{
postcancellationTokenSource.Dispose();
}
}

```

Response

Get the Authentication token from the response as follows:

```
response.Content.ReadAsAsync<AuthenticationResponse>().Result
```

5.1.2 Logout Procedure

To logout from the system using SiPass integrated HR API, use the following URL
https://HOSTNAME:PORT/api/V1/hr/Authentication

5.1.2.1 Consume Logout API: Language C#

Perform the following steps to logout:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **DELETE**.
4. Under **HEADERS**,
 - enter the **name** as **clientUniqueld** and enter its corresponding **value**.
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **language** and **value** as **English**
 - enter the **name** as **Authorization** and its **value** as **Authorization token**.
5. Send **DELETE** request.
 - ⇒ The request is sent to the server.
6. The API service authenticates the request and the parameters.
 - ⇒ On successful logout, the API returns the same token that was provided in the authorization header.

**NOTICE**

If the logged in client has reached beyond the session timeout limit, the session will expire and the client will be logged out automatically. After the client is logged out, if user tries to send any request, an Authentication exception will be thrown.

Code Snippet:**Request**

The Delete Request can be sent as shown below:

```
private async Task<HttpResponseMessage> DeleteAsyncHttpRequest<T>(string resource, T
payload)
{
    HttpResponseMessage response;
    CancellationTokenSource cancellationTokenSource = new CancellationTokenSource();
    try
    {
        #region HAL
        var cancelToken = cancellationTokenSource.Token;
        //To resolve certification issue
        ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };
        #endregion

        using (var httpClient = new HttpClient { Timeout = TimeSpan.FromSeconds(_webTimeout) })
        {
            AddHeaderValue(httpClient, resource);
            if (resource.Equals(UriUtils.URI_DELETE_Logout, StringComparison.CurrentCultureIgnoreCase))
            {
                httpClient.DefaultRequestHeaders.Add("Authorization", AuthenticationInfo.Token);
            }
            if(payload != null)
            {
                var request = new HttpRequestMessage(HttpMethod.Delete, resource)
                {
                    Content = new StringContent
                    (JsonConvert.SerializeObject(payload),
                    Encoding.UTF8, "application/json")
                };
            }
        }
    }
}
```

```
response = await httpClient.SendAsync(request);
}
else
{
response = await httpClient.DeleteAsync(resource, cancelToken);
}

CheckForError(response,false);
}
}
finally
{
cancellationTokenSource.Dispose();
}
return response;
}

private void AddHeaderValue(HttpClient client,string resource)
{
CreaetHeaderAttributes(client);
if (!resource.Equals(UriUtils.URI_POST_Login, StringComparison.CurrentCultureIgnoreCase))
{
client.DefaultRequestHeaders.Add("Authorization", AuthenticationInfo.Token);
}
client.DefaultRequestHeaders.Add("clientUniqueId", AuthenticationInfo.ClientUniqueId);
}
```

Response:

The response returns the same token that was provided in the authorization header.

5.2 Cardholder

The APIs related to the Cardholder operation are explained in this section.

5.2.1 Get All Cardholders

To retrieve all the Cardholders from the system, use the URL:

```
https://HOSTNAME:PORT/Api/v1/hr/Cardholders?searchString=&apId=Cardholders&fields=FirstName,LastName&sortingOrder={"FieldName":"LastName","Value":"","SortingOrder":0}&filterExpression={"Identifier": "", "LogicalOperator": 0, "FilterExpressions": [ { 'Identifier': "", 'LogicalOperator': 0, 'IsNegated': false } ], 'ConditionExpressions': [ { 'FieldName': 'FirstName', 'Value': 'A', 'ConditionalOperator': 2 }, ], 'IsNegated': false, }&startIndex=0&endIndex=3
```

5.2.1.1 Consume GET API: Language C#

Perform the following steps to get all cardholders:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **GETNOTICE!**.
4. Under **HEADERS**,
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **clientUniqueId** and enter its corresponding **value**.
 - enter the **name** as **language** and **value** as **English**
 - enter the **name** as **Authorization** and its **value** as **Authorization token**.

For English language, the following format is supported:

	Language	Date Format	Sample	Time Formats Supported	Sample	Date Time Format	Sample	Operations Supported
1.	English	MM/dd/yyyy	10/29/2020	HH:mm:ss	17:10:15	MM/dd/yyyy HH:mm:ss	10/29/2020 17:10:15	Create and Update
			10/29/2020	hh:mm a	05.30 PM	MM/dd/yyyy hh:mm a	10/29/2020 05:30 PM	Create, Update, and Search
		YYYY-MM-DD	2020-10-29	HH:mm:ss	20:11:08	YYYY-MM-DD HH:mm:ss	2020-10-29 20:11:08	Create and Update
			2020-10-29	hh:mm a	05:11 PM	YYYY-MM-DD hh:mm a	2020-10-29 05:11 PM	
		yyyy/mm/dd	2020/10/29	HH:mm:ss	14:11:08	yyyy/mm/dd HH:mm:ss	2020/10/29 14:11:08	Create and Update
			2020/10/29	hh:mm a	03:11 PM	yyyy/mm/dd hh:mm a	2020/10/29 03:11 PM	

5. Send GET request.

⇒ The request is sent to the server.

6. The API service authenticates the request and the parameters.

⇒ On successful authorization, the API returns all the cardholders.

– or...

⇒ On failed authorization, an exception will be thrown.

Sample Response data

```
{
  "Records":[
    {
      "BaseCardNumber":"52514",
      "EmployeeNumber":"e12348",
      "EndDate":"2017-12-05T00:00:00",
      "FirstName":"John",
      "LastName":"Kennedy",
      "PrimaryWorkgroupName":"<None>",
      "StartDate":"2015-08-17T00:00:00",
      "Status":63,
      "Token":"5024",
      "Potrait":"api/V1/cardholders/images?imgId=5024&token=7CCAAC41FB15F1CF9D1D1CC50CA
      CCA7509AA3FE4488A143BAC2FD8F69F98E33&noCache=664178496",
      "LastUpdatedDateTime":"2017-12-08T11:11:32"
    },
    {
      "BaseCardNumber":"52512",
      "EmployeeNumber":"E12346",
      "EndDate":"2116-01-25T00:00:00",
      "FirstName":"John",
      "LastName":"Smith",
      "PrimaryWorkgroupName":"Conference_Room_Incharger",
      "StartDate":"2015-08-17T00:00:00",
      "Status":61,
      "Token":"5022",
      "Potrait":"api/V1/cardholders/images?imgId=5022&token=7CCAAC41FB15F1CF9D1D1CC50CA
      CCA7509AA3FE4488A143BAC2FD8F69F98E33&noCache=664178496",
      "LastUpdatedDateTime":"2016-01-25T11:21:30"
    }
  ],
  "TotalRecords":-1,
  "TotalDisplayRecords":-1,
  "ResourceList":[]
}
```

Code Snippet:

Request

The GET request can be sent as shown below:

```
var dicQryParam = Helper.FormDictionary(GetresourceUri(UriUtils.URI_GET_ALLEmployees),
startIndex.ToString(CultureInfo.InvariantCulture), endIndex.ToString(CultureInfo.InvariantCulture),
filterExpression, sortingOrder, "", Helper.AppId, Helper.ReturnFields);

var requestUrl = SetQueryParameterForGellAllCardholder(dicQryParam);

var employeeList = ClientService.GetWebRequest<RepresentationBase<Cardholder>>(requestUrl);

public TR GetWebRequest<TR>(string resource)
{
var response = GetWebRequest<TR>(resource, false);
return response;
}

public TR GetWebRequest<TR>(string resource, bool suppressDeserialize)
{
var result = default(TR);

try
{
result = Task.Run(() => GetAsyncHttpRequest<TR>(resource, suppressDeserialize)).Result;
}

Catch
return result;
}
```

Response:

Response returns the Cardholder list as collection type

IEnumerable<Cardholder>.

```
result = Task.Run(() => ReadContent<TR>(response, suppressDeserialize)).Result;
```

5.2.2 Get Cardholder By ID

To retrieve a cardholder by an id from the system, use the URL
<https://HOSTNAME:PORT/Api/v1/hr/cardholders>

5.2.2.1 Consume GET API: Language C#

Perform the following steps to get a cardholder by an id:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Repeat the steps from 3,4,5, and 6 as mentioned in the GETALL Cardholders section.
4. Send **GET** request.
 - ⇒ The request is sent to the server.
5. The API service authenticates the request and the parameters.
 - ⇒ On successful authorization, the API returns a cardholder by ID.
 - or...
 - ⇒ On failed authorization, an exception will be thrown.

Successful Response data (sample)

```
{
  "Credentials": [
    {
      "CardNumber": "75020148",
      "EndDate": "2018-01-31T00:00:00",
      "Pin": 1234,
      "ProfileId": 2,
      "ProfileName": "Base1",
      "StartDate": "2018-01-24T00:00:00",
      "FacilityCode": 100,
      "CardTechnologyCode": 8
    }
  ],
  "AccessRules": [],
  "EmployeeNumber": "",
  "EndDate": "2018-01-31T04:37:00",
  "FirstName": "Nizar",
  "GeneralInformation": "",
  "LastName": "Khan",
  "PersonalDetails": {
    "Address": "",
    "ContactDetails": {
      "Email": "",
      "MobileNumber": "",
      "MobileServiceProviderId": "0",
      "PagerNumber": "",
      "PagerServiceProviderId": "0",
      "PhoneNumber": ""
    }
  },
}
```

```
"DateOfBirth": "",
"PayrollNumber": "",
"Title": "",
"UserDetails": {
  "Password": "",
  "UserName": ""
}
},
"PrimaryWorkgroupId": 2000000000,
"ApbWorkgroupId": 2000000000,
"PrimaryWorkgroupName": "<Visitor>",
"NonPartitionWorkGroups": [],
"SmartCardProfileId": "0",
"StartDate": "2018-01-24T16:37:00",
"Token": "9626",
"TraceDetails": {},
"Vehicle1": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
"Vehicle2": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
"Portrait": "api/V1/cardholders/images?imgId=9626&token=2176D7A5FF681FD9BFF68EED5EB8DCC8101F961A7F97B641B49D2CF3D03EDF&noCache=1745370457",
"VisitorDetails": {
  "VisitorCardStatus": 0,
  "VisitorCustomValues": {
    "Company": "CROMPTON",
    "Profile": "Profile1",
    "Reason": "reason",
    "License": "driver license",
    "Email": "",
    "CompanyCode": ""
  }
},
"CustomFields": []
}
```

Code Snippet:

Request

The GET request can be sent as shown below:

```
string url = GetresourceUri(UriUtils.URI_GET_ALLEmployees);
url = SetQueryParameter(url, "id", cardholderId.ToString(CultureInfo.InvariantCulture));
Cardholder cardholder = ClientService.GetWebRequest<Cardholder>(url);
```

Response:

```
result = Task.Run(() => ReadContent<TR>(response, suppressDeserialize)).Result;
```

5.2.3 Get Cardholder Image by Cardholder ID

To retrieve the image of a cardholder from the system, use the following URL
`https://HOSTNAME:PORT/api/V1/hr/Cardholders?imgId={Cardholder id}&token={session token}`

5.2.3.1 Consume GET API: Language C#

Perform the following steps to get an image of a cardholder:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **GETNOTICE!** .
4. Under the **HEADERS**,
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **clientUniqueld** and its **value**.
 - enter the **name** as **language** and **value** as **English**
 - enter the **name** as **Authorization** and its **value** as **Authorization token**
5. Send **GET** request.
 - ⇒ The request is sent to the server.
6. The API service authenticates the request and the parameters.
 - ⇒ On successful authorization, the API returns a cardholder image of the associated cardholder id in binary format.
 - or...
 - ⇒ On failed authorization, an exception will be thrown.

Response

If the status code is success, the response returns byte stream of cardholder/cardholder image

If no specific image is uploaded for the cardholder/while configuring a cardholder, a default image will be returned in byte stream format.

5.2.4 Create Cardholder

To create a cardholder in the system, use the URI **`https://HOSTNAME:PORT/Api/v1/hr/cardholders`**

5.2.4.1 Consume POST API: Language C#

Perform the following steps to create a cardholder:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **POSTNOTICE!** .
4. Under the **HEADERS**,
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **clientUniqueld** and its **value**.
 - enter the **name** as **language** and **value** as **English**
 - enter the **name** as **Authorization** and its **value** as **Authorization token**

Successful request data (sample)

```
Request
{
  "Attributes": {
    "Accessibility": false,
    "ApbExclusion": false,
    "ApbReEntryExclusion": false,
    "Isolate": false,
    "SelfAuthorize": false,
    "Supervisor": false,
    "Visitor": false,
    "Void": false,
  },
  "Credentials": [
    {
      "Active": false,
      "CardNumber": 75020147,
      "EndDate": "2018-01-31T00:00:00",
      "Pin": 1234,
      "RevisionNumber": 0,
      "PinErrorDisabled": false,
      "ProfileId": 2,
      "ProfileName": "Base1",
      "StartDate": "2018-01-24T00:00:00",
      "FacilityCode": 100,
      "CardTechnologyCode": 8
    }
  ],
}
```

```
"BaseCardNumber": null,
"AccessRules": [],
"EmployeeNumber": "",
"EndDate": "2018-01-31T04:37:00",
"FirstName": "Nizar",
"GeneralInformation": "",
"LastName": "Khan",
"EmployeeName": null,
"PersonalDetails": {
  "Address": "",
  "ContactDetails": {
    "Email": "",
    "MobileNumber": "",
    "MobileServiceProvider": null,
    "MobileServiceProviderId": "0",
    "PagerNumber": "",
    "PagerServiceProvider": null,
    "PagerServiceProviderId": "0",
    "PhoneNumber": "",
    "UseEmailforMessageForward": false
  },
  "DateOfBirth": "",
  "PayrollNumber": "",
  "Title": "",
  "UserDetails": {
    "Password": "",
    "UserName": ""
  }
},
"PrimaryWorkgroupId": 2000000000,
"ApbWorkgroupId": 2000000000,
"PrimaryWorkgroupName": "<Visitor>",
"NonPartitionWorkGroups": [],
"SmartCardProfileId": "0",
"SmartCardProfileName": null,
"StartDate": "2018-01-24T16:37:00",
"Status": 0,
"Token": "-1",
"TraceDetails": {
  "CardLastUsed": null,
  "CardNumberLastUsed": null,
  "LastApbLocation": null,
  "PointName": null,
  "TraceCard": false
},
```

```
"Vehicle1": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
"Vehicle2": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
"Portrait": null,
"PrimaryWorkGroupAccessRule": null,
"NonPartitionWorkgroupAccessRules": null,
"VisitorDetails": {
  "VisitedEmployeeFirstName": null,
  "VisitedEmployeeId": 0,
  "VisitedEmployeeLastName": null,
  "VisitorCardIssueTime": null,
  "VisitorCardReturnTime": null,
  "VisitorCardStatus": 0,
  "VisitorCustomValues": {
  },
  "CustomFields": [],
  "FingerPrints": null,
  "CardholderPortrait": null,
  "IsImageChanged": false,
  "IsSignatureChanged": false,
  "CardholderSignature": null,
  "ElevatorRole": 0,
  "ElevatorLight": 0,
  "ElevatorLanguage": 0,
  "StartDateWithoutTime": null,
  "EndDateWithoutTime": null,
  "LastUpdatedDateTime": null,
  "Reference": 0,
  "_links": [],
  "_embedded": null
}
}
```

1. Send **POST** request.

- ⇒ The request is sent to the server.

2. The API service authenticates the request and the parameters.

- ⇒ On successful authorization, the API service creates the cardholder and returns the cardholder information with token.

- or...

- ⇒ On failed authorization, an exception will be thrown.

Successful response data (sample)**Response**

```
{
  "Attributes": {
  },
  "Credentials": [
  {
    "CardNumber": "75020146",
    "EndDate": "2018-01-31T00:00:00",
    "Pin": 1234,
    "ProfileId": 2,
    "ProfileName": "Base1",
    "StartDate": "2018-01-24T00:00:00",
    "FacilityCode": 100,
    "CardTechnologyCode": 8
  }
  ],
  "AccessRules": [],
  "EmployeeNumber": "",
  "EndDate": "2018-01-31T04:37:00",
  "FirstName": "Nizar",
  "GeneralInformation": "",
  "LastName": "Khan",
  "PersonalDetails": {
    "Address": "",
    "ContactDetails": {
      "Email": "",
      "MobileNumber": "",
      "MobileServiceProviderId": "0",
      "PagerNumber": "",
      "PagerServiceProviderId": "0",
      "PhoneNumber": ""
    }
  },
```

```
"DateOfBirth": "",
  "PayrollNumber": "",
  "Title": "",
  "UserDetails": {
    "Password": "",
    "UserName": ""
  }
  },
  "PrimaryWorkgroupId": 2000000000,
  "ApbWorkgroupId": 2000000000,
  "PrimaryWorkgroupName": "<Visitor>",
  "NonPartitionWorkGroups": [],
  "SmartCardProfileId": "0",
  "StartDate": "2018-01-24T16:37:00",
  "Token": "9625",
  "TraceDetails": {},
  "Vehicle1": {
```

```
"CarColor": "",
"CarModelNumber": "",
"CarRegistrationNumber": ""
},
"Vehicle2": {
"CarColor": "",
"CarModelNumber": "",
"CarRegistrationNumber": ""
},
"Portrait": "api/V1/cardholders/images?imgId=9625&token=FC794612EF1FB2B6DA4684B4255FA7E9AE465422E4844DB76809A6161AAE5&noCache=2120765990",
"VisitorDetails": {
"VisitorCustomValues": {
}
},
"CustomFields": [],
}
```

Code Snippet**Request**

The **POST** request can be sent as shown below.

```

var uri = empValue.Attributes != null && empValue.Attributes.Visitor
? UriUtils.URI_Visitors
: UriUtils.URI_GET_ALLEmployees;
var url = GetresourceUri(uri);
return ClientService.PostWebRequest(url, empValue);

public HttpResponseMessage PostWebRequest<T>(string resource, T payload)
{
    var result = PostWebRequest(resource, new Dictionary<string, string>(), payload);
    return result;
}

public HttpResponseMessage PostWebRequest<T>(string resource, IReadOnlyDictionary<string,
string> dicReqHeaders, T payload)
{
    try
    {
        var result = Task.Run(() => PostAsyncHttpRequest(resource, dicReqHeaders, payload)).Result;
        return result;
    }
    CatchRegion
}

private async Task<HttpResponseMessage> PostAsyncHttpRequest<T>(string resource,
IReadOnlyDictionary<string, string> dicReqHeaders, T payload)
{
    CancellationTokenSource postcancellationTokenSource = new CancellationTokenSource();
    try

```

```

{
    var cancelToken = postcancellationTokenSource.Token;
    //To resolve certification issue
    ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };
    HttpResponseMessage response;
    using (var httpClient = new HttpClient { Timeout = TimeSpan.FromSeconds(_webTimeout) })
    {
        AddHeaderValue(httpClient, resource);
        response = await httpClient.PostAsJsonAsync(resource, payload, cancelToken);
        CheckForError(response, false);
    }
    return response;
}
finally
{
    postcancellationTokenSource.Dispose();
}
}

```

Response:

If the status code is success, the response returns cardholder information.

If the status code is fail, an empty information will be retrieved.

5.2.5 Update Cardholder

To update an existing cardholder in the system, use the following URL **https://
HOSTNAME:PORT/Api/v1/hr/cardholders?id={{cardholdertoken}}**

5.2.5.1 Consume PUT API: Language C#

Perform the following steps to update a cardholder:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **PUTNOTICE!** .
4. Under the **HEADERS**,
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **clientUniqueld** and its **value**.
 - enter the **name** as **language** and **value** as **English**
 - enter the **name** as **Authorization** and its **value** as **Authorization token**

Successful request data (sample)

```
{
  "Attributes": {
    "Accessibility": false,
    "ApbExclusion": false,
    "ApbReEntryExclusion": false,
    "Isolate": false,
    "SelfAuthorize": false,
    "Supervisor": false,
  },
  "Credentials": [
    {
      "Active": false,
      "CardNumber": 75020148,
      "EndDate": "2018-01-31T00:00:00",
      "Pin": 1234,
      "RevisionNumber": 0,
      "PinErrorDisabled": false,
      "ProfileId": 2,
      "ProfileName": "Base1",
      "StartDate": "2018-01-24T00:00:00",
      "FacilityCode": 100,
      "CardTechnologyCode": 8
    }
  ],
  "BaseCardNumber": null,
  "AccessRules": [],
  "EmployeeNumber": "",
  "EndDate": "2018-01-31T04:37:00",
  "FirstName": "Nizar",
  "GeneralInformation": "",
  "LastName": "Khan",
  "EmployeeName": null,
  "PersonalDetails": {
    "Address": "",
    "ContactDetails": {
      "Email": "",
      "MobileNumber": "",
      "MobileServiceProvider": null,
      "MobileServiceProviderId": "0",
      "PagerNumber": "",
      "PagerServiceProvider": null,
      "PagerServiceProviderId": "0",
      "PhoneNumber": "",
      "UseEmailforMessageForward": false
    }
  },
}
```

```
"DateOfBirth": "",
"PayrollNumber": "",
"Title": "",
"UserDetails": {
  "Password": "",
  "UserName": ""
}
},
"PrimaryWorkgroupId": 2000000000,
"ApbWorkgroupId": 2000000000,
"PrimaryWorkgroupName": "<Visitor>",
"NonPartitionWorkGroups": [],
"SmartCardProfileId": "0",
"SmartCardProfileName": null,
"StartDate": "2018-01-24T16:37:00",
"Status": 0,
"Token": "-123",
"TraceDetails": {
  "CardLastUsed": null,
  "CardNumberLastUsed": null,
  "LastApbLocation": null,
  "PointName": null,
  "TraceCard": false
},
"Vehicle1": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
"Vehicle2": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
}
```

```
"Potrait": null,
"PrimaryWorkGroupAccessRule": null,
"NonPartitionWorkgroupAccessRules": null,
"VisitorDetails": {
  "VisitedEmployeeFirstName": null,
  "VisitedEmployeeId": 0,
  "VisitedEmployeeLastName": null,
  "VisitorCardIssueTime": null,
  "VisitorCardReturnTime": null,
  "VisitorCardStatus": 0,
  "VisitorCustomValues": {
  },
  "CustomFields": [],
  "FingerPrints": null,
  "CardholderPortrait": null,
  "IsImageChanged": false,
  "IsSignatureChanged": false,
  "CardholderSignature": null,
  "ElevatorRole": 0,
  "ElevatorLight": 0,
  "ElevatorLanguage": 0,
  "StartDateWithoutTime": null,
  "EndDateWithoutTime": null,
  "LastUpdatedDateTime": null,
}
```

1. Send **PUT** request.

- ⇒ The request is sent to the server.

2. The API service authenticates the request and the parameters.

- ⇒ On successful authorization, the API service updates the cardholder.

- or...

- ⇒ On failed authorization, an exception will be thrown.

Successful response data (sample)

```
{
  "Attributes": {
  },
  "Credentials": [
    {
      "CardNumber": "75020148",
      "EndDate": "2018-01-31T00:00:00",
      "Pin": 1234,
      "ProfileId": 2,
      "ProfileName": "Base1",
      "StartDate": "2018-01-24T00:00:00",
      "FacilityCode": 100,
      "CardTechnologyCode": 8
    }
  ],
  "AccessRules": [],
  "EmployeeNumber": "",
  "EndDate": "2018-01-31T04:37:00",
  "FirstName": "Nizar",
  "GeneralInformation": "",
  "LastName": "Khan",
  "PersonalDetails": {
    "Address": "",
    "ContactDetails": {
      "Email": "",
      "MobileNumber": "",
      "MobileServiceProviderId": "0",
      "PagerNumber": "",
      "PagerServiceProviderId": "0",
      "PhoneNumber": ""
    },
    "DateOfBirth": "",
    "PayrollNumber": "",
    "Title": "",
    "UserDetails": {
      "Password": "",
      "UserName": ""
    }
  }
}
```

```
"PrimaryWorkgroupId": 2000000000,
"ApbWorkgroupId": 2000000000,
"PrimaryWorkgroupName": "<Visitor>",
"NonPartitionWorkGroups": [],
"SmartCardProfileId": "0",
"StartDate": "2018-01-24T16:37:00",
"Token": "123",
"TraceDetails": {},
"Vehicle1": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
"Vehicle2": {
  "CarColor": "",
  "CarModelNumber": "",
  "CarRegistrationNumber": ""
},
"Potrait": "api/V1/cardholders/images?imgId=9626&token=2176D7A5FF681FD9BFF68EED5EB8DCC8101F961A7F97B641B49D2CF3D03EDF&noCache=1745370457",
"VisitorDetails": {
  "VisitorCardStatus": 0,
  "VisitorCustomValues": {
  }
},
"CustomFields": [],
}
```

Code Snippet

Request

The PUT request can be sent as shown below:

```
var uri = empValue.Attributes != null && empValue.Attributes.Visitor
? UriUtils.URI_Visitors
: UriUtils.URI_GET_ALLEmployees;
var urlQryParam = SetQueryParameter(url, "id", empValue.Token);
return ClientService.PutWebRequest<Cardholder, HttpResponseMessage>(urlQryParam, empValue);
public TR PutWebRequest<T, TR>(string resource, T payload)
{
    try
    {
        TR result = Task.Run(() => PutAsyncHttpRequest<T, TR>(resource, payload)).Result;
        return result;
    }
    CatchRegion
}

[CompilerGenerated]
private async Task<TR> PutAsyncHttpRequest<T, TR>(string resource, T payload)
{
    HAL
    CancellationTokenSource cancellationTokenSource = new CancellationTokenSource();
    var cancellationToken = cancellationTokenSource.Token;
    HttpResponseMessage response = null;
    try
    {
        {
            using (var client = new HttpClient {Timeout = TimeSpan.FromSeconds(_webTimeout)})
            {
                AddHeaderValue(client, resource);
                response = await client.PutAsJsonAsync(resource, payload, cancellationToken);
                CheckForError(response, false);
            }
        }
    }
    finally
    {
        {
            cancellationTokenSource.Dispose();
        }
    }

    return await ReadContentValue<TR>(response);
}
```

Response:

If the status code is success, the response returns cardholder information.

5.2.6 Delete Cardholder

To delete a cardholder from the system, use the following URL **https://HOSTNAME:PORT/api/v1/hr/cardholders?id={{cardholdertoken}}**

5.2.6.1 Consume DELETE API: Language C#

Perform the following steps to delete a cardholder:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **DELETENOTICE!** .
4. Under the **HEADERS**,
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **clientUniqueld** and its **value**.
 - enter the **name** as **language** and **value** as **English**
 - enter the **name** as **Authorization** and its **value** as **Authorization token**.
5. Send **DELETE** request.
 - ⇒ The request is sent to the server.
6. The API service authenticates the request and the parameters.
 - ⇒ On successful authorization, the API service deletes the cardholder.
 - or...
 - ⇒ On failed authorization, an exception will be thrown.

Code Snippet

Request

The **DELETE** request can be sent as shown below:

```
string url = SetQueryParameter(GetresourceUri(UriUtils.URI_GET_ALLEmployees), "id",
cardholderToken);
return ClientService.DeleteWebRequest(url);

public HttpResponseMessage DeleteWebRequest(string resource)
{
string payload = null ;
var result = DeleteWebRequest(resource, payload);
return result;
}

[CompilerGenerated]
public HttpResponseMessage DeleteWebRequest<T>(string resource, T payload)
{
try
{
var result = Task.Run(() => DeleteAsyncHttpRequest(resource, payload)).Result;
return result;
}
CatchRegion
return null;
}

[CompilerGenerated]
private async Task<HttpResponseMessage> DeleteAsyncHttpRequest<T>(string resource, T
payload)
{
HttpResponseMessage response;
CancellationTokensource cancellationTokensource = new CancellationTokensource();
try
```

```
{
  HAL
  using (var httpClient = new HttpClient { Timeout = TimeSpan.FromSeconds(_webTimeout) })
  {
    //SetHeaders(httpClient, dicReqHeaders);
    AddHeaderValue(httpClient, resource);
    if (resource.Equals(UriUtils.URI_DELETE_Logout, StringComparison.CurrentCultureIgnoreCase))
    {
      httpClient.DefaultRequestHeaders.Add("Authorization", AuthenticationInfo.Token);
    }
    if(payload != null)
    {
      var request = new HttpRequestMessage(HttpMethod.Delete, resource)
      {
        Content = new StringContent(JsonConvert.SerializeObject(payload), Encoding.UTF8,
        "application/json")
      };
      response = await httpClient.SendAsync(request);
    }
    else
    {
      response = await httpClient.DeleteAsync(resource, cancelToken);
    }
    CheckForError(response,false);
  }
}
FinallyRegion
return response;
}
```

Response:

If the status code is success, the response returns empty. If the status code is fail, an error code will be retrieved.

5.2.7 Get All Cardholders With Credentials And Access Privileges

To retrieve all the Cardholders with credentials and access privileges from the system, use the URL:

```
https://HOSTNAME:PORT/Api/v1/hr/Cardholders?searchString=&appId=Cardholders&fields=FirstName,LastName&sortingOrder={"FieldName":"LastName","Value":"","SortingOrder":0}&filterExpression={"Identifier": "", 'LogicalOperator': 0, 'FilterExpressions': [ { 'Identifier': "", 'LogicalOperator': 0, 'IsNegated': false } ], 'ConditionExpressions': [ { 'FieldName': 'FirstName', 'Value': 'A', 'ConditionalOperator': 2 }, ], 'IsNegated': false, }&startIndex=0&endIndex=3&credentialFields=CardNumber,Pin,ProfileId,ProfileName,StartDate,EndDate&accessRulesFields=ObjectToken,ObjectName,TimeScheduleToken,ControlModelId,StartDate
```

5.2.7.1 Consume GET API: Language C#

Perform the following steps to get all cardholders:

1. Create the **HttpClient** instance.
2. Enter the URL (mentioned above).
3. Choose the **HTTP Method** as **GETNOTICE!** .
4. Under **HEADERS**,
 - enter the **name** as **Content-Type** and **value** as **application/json**
 - enter the **name** as **clientUniqueld** and enter its corresponding **value**.
 - enter the **name** as **language** and **value** as **English**
 - enter the **name** as **Authorization** and its **value** as **Authorization token**.

For English language, the following format is supported.

S No	Language	Date Format	Sample	Time Format	Sample	Date Time Format	Sample
1	English	MM/dd/yyyy	04/27/2018	h:mm a	5:30 PM	MM/dd/yyyy h:mm a	04/27/2018 5:30 PM

5. Send **GET** request.
 - ⇒ The request is sent to the server.
6. The API service authenticates the request and the parameters.
 - ⇒ On successful authorization, the API returns all the cardholders.
 - or...
 - ⇒ On failed authorization, an exception will be thrown.

Sample Response data

```
{
  "Records":[
    {
      "Credentials": [
        {
          "CardNumber": "107140",
          "EndDate": "2118-10-16T23:59:59",
          "ProfileId": 1,
          "ProfileName": "Base",
          "StartDate": "2018-10-16T00:00:00",
          "FacilityCode": 0,
          "CardTechnologyCode": 0,
          "PinDigit": 0
        }
      ],
      "BaseCardNumber": "107140",
      "AccessRules": [
        {
          "ObjectToken": "22",
          "ObjectName": "*Access Group-Mix-0001",
          "RuleType": 0,
          "TimeScheduleToken": "0",
          "StartDate": null,
          "EndDate": null,
          "ArmingRightsId": null,
          "ControlModelId": 0,
          "Side": 0
        },
        {
          "ObjectToken": "23",
          "ObjectName": "*Access Group-Mix-0007",
          "RuleType": 0,
          "TimeScheduleToken": "0",
          "StartDate": null,
          "EndDate": null,
          "ArmingRightsId": null,
          "ControlModelId": 0,
          "Side": 0
        }
      ]
    }
  ],
}
```

```
"EmployeeNumber": "uqfhasib",
"EndDate": "2118-10-16T00:00:00",
"FirstName": "FN_200150",
"LastName": "LN_200150",
"PrimaryWorkgroupName": "UQ International",
"StartDate": "2018-10-16T00:00:00",
"Status": 61,
"Token": "9583",
"Potrait":
"api/V1/cardholders/images?imgId=9583&token=5ACF1C3E5532A4477B498A9918CFB2D83C3D5D5
1258AE1468F9CA35DC7E534E8&noCache=472847334",
"CardTemplate": null,
"LastUpdatedDateTime": "2018-10-16T17:47:41",
"_links": []
},

{
"Credentials": [],
"BaseCardNumber": "52512",
"AccessRules": [],
"EmployeeNumber": "E12346",
"EndDate": "2116-01-25T00:00:00",
"FirstName": "John",
"LastName": "Smith",
"PrimaryWorkgroupName": "Conference_Room_Incharger",
"StartDate": "2015-08-17T00:00:00",
"Status": 61,
"Token": "5022",
"Potrait": "api/V1/cardholders/images?imgId=5022&token=7CCAAC41FB15F1CF9D1D1CC50CACCA
7509AA3FE4488A143BAC2FD8F69F98E33&noCache=664178496",
"LastUpdatedDateTime": "2016-01-25T11:21:30"
}
],
"TotalRecords": -1,
"TotalDisplayRecords": -1,
"ResourceList": [
]
}
```

5.3 Session Management

This API is used to keep the session alive for a definite period of time. The interval period should be one third of session timeout for better consistency.

Renew success notification will be received as a message (event) with service type and sub type as alive. This message will be received only if the message receiving is subscribed (Signal R).

To renew the session, use the following URL

<https://{host}/api/V1/hr/Authentication?token={token}>

5.3.1 Consume GET API: Language C#

To renew the session:

1. Modify the key value in the configuration file **Siemens.SiPass.HR.API.Host.config**. This file exists in the server installation path.
 - `<add key="SessionExpirationTimeout" value="300" />`



NOTICE

Session timeout value should always be in seconds. By default, session timeout occurs in 300 seconds.

Perform the following steps to renew the session:

2. Create the HttpClient instance and login.
3. Enter the URL (mentioned above) to renew the session within timeout minutes.
4. Choose the HTTP Method as **GET**.
5. Under the **HEADERS**,
 - enter the name as **Content-Type** and value as **application/json**
 - enter the name as **clientUniqueld** and its value.
 - enter the name as **language** and value as **English**
 - enter the name as **Authorization** and its value as **Authorization token**
6. Send **GET** request.
 - ⇒ The request is sent to the server.
7. The API service renews the request based on the parameters.
 - ⇒ On successful renew, the API returns an Authentication object.

Response

If the status code is success, authorization data is returned.

Refer to API specification document for more information.

5.4 Frequently Asked Questions

1. I have sent an authentication request but why is the token not returned?
 - ⇒ Make sure that you have added the clientUniqueld in the request header along with valid user credentials.
2. I have provided valid data in the request but why is the API not returning the token?
 - ⇒ Make sure your installed port is bound with a valid certificate.

6 API Documentation

1. Browse the URL <https://HOSTNAME:PORT/swagger/ui/index> to view the list of API's in the system.
2. Click on the **Expand Operations** to know more about the API. Sample screenshot of GET Request.

7 Test Client Source Code

The SiPass integrated HR API Test Client sample code is found in the **HRAPISample.7z** file that is available in the installation folder of HR API Test Client.



NOTICE

The HostName and Port has to be updated in the App.config file. The following is the key that needs to be updated with HostName and Port where the API is hosted.

```
<add key="HrApiHostName"  
value="http://SIPASS_SERVER:API_PORT"/>
```

To compile the HR API test client, it is recommended to open the HR API test client solution in **Visual Studio 2017**, and set the target framework as **.Net Framework 4.8**

Issued by
Siemens Switzerland Ltd
Smart Infrastructure
Global Headquarters
Theilerstrasse 1a
CH-6300 Zug
+41 58 724 2424
www.siemens.com/buildingtechnologies

© Siemens Switzerland Ltd, 2023
Technical specifications and availability subject to change without notice.