Skills DB Pro API

API introduction

Skills DB Pro includes a REST API which is available to paid license holders and provides a way for organizations to programmatically access their data in Skills DB Pro. The API uses JSON formatted data for communications.

Enabling the API

You can enable the Skills DB Pro API for your instance via the following steps:
Navigate to Admin -> Company-> API -> Manage API Keys

An API key and Secret will be generated which are used to obtain an access token (refer to "Authentication" below).

You can disable the API by clicking the "Delete keys" button. Note that this will in fact delete the keys that have been generated. If you enable the API again different keys will be generated which means you will have to update the keys in your code.

API limits

Rate limit

The Skills DB Pro API is limited to 100 requests per 15 minute interval per Skills DB Pro instance. If you exceed the limit, the API will respond with HTTP/1.1 403 Unauthorized and the "message" parameter will contain the string "API rate limit exceeded. Try again in x minute(s)"

Don't exceed the rate limit for a prolonged period of time as it can be interpreted as a Denial of Service attack which may result in your IP address being blocked. Pay particular attention to parts of your script that retry failed requests (eg: invalid access token, API rate limit exceeded, etc). If there is no delay between retries you may find your script flooding the API with requests in excess of the rate limit.

Data limit

The API will return a maximum of 500 records for GET requests. If you receive a response with 500 records, you can access the further records by making a subsequent API call using the "start_index" parameter.

Methods:

Base Service Address is: https://address/Service1.svc

Next methods are defined in Service1.cs:

Authentication:

Obtaining an access token

Access Tokens are used to authenticate API requests. Obtaining an access token is achieved by performing a HTTP POST including your API key and secret to the access_token endpoint.

You can have one active access token at any time. If you apply for another access token when there is already one active, the active token will be deleted and replaced with the new one. Note that access tokens expire after a short period of time that is specified in the JSON response (in seconds). For optimal operation, keep using the same access token until it expires.

Login [POST] Used to identify user by APICode. OnSuccess it will return CompanyID and Jwt token. With CompanyID and Jwt Token user can call other methods.	
Address:	http://address/Service1.svc/login
Request Parameters:	**APICode:** Sent in request body as json object.

Request Example:	{
	"**APICode**" : "CF1E8BAD-09DD-4F95-BC57-C034E4B51601"
	}

Response:	{
	"**token**": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
	"**authenticated**": true,
	"**timestamp**": "/Date(1542722679929+0100)/",
	"**companyID**": 1
	}

IndividualDevelopmentPlan

GetIDPsByCompanyID [GET] Query database for IDP's by CompanyID. It will return 500 records per page.		
Address:	http://address/Service1.svc/IDP/{companyID}?part={part}	
Request Parameters:	**CompanyID**: **Jwt Token**: **Part**:	*Value returned from Login response.* *Added to authorization header as bearer token.* *Records per page parameter. 1 is first 500, 2 is next 500 and so on.*
Request Example:	http://localhost:50498/Service1.svc/IDP/1?part=1	
Response:	500 records with structure:	

	"EmailAddress": "a.bouhdary@ARB.com", "LearningPlanActivityIDOutsideSystem": null, "LearningPlanEndDate": "20.04.2018", "LearningPlanPercentComplete": 0, "LearningPlanStartDate": "21.03.2018"

`GetIDPQuery` [GET] Query database for IDP's by several parameters. It will return 500 records per page.		
Address:	http://address/Service1.svc/`IDP/Query/{companyID}?email={email}&date={dateQuery}&part={part}`	
Request Parameters:	**CompanyID**: **Jwt Token**: **Email**: **Date**: **Part**:	*Value returned from Login response* *Added to authorization header as bearer token.* *String parameter. Can be used to query email records like:* **gmail.com – it will return all records that contains gmail.com in emaiil field.* *john* - it will return all records where email starts with john.* *String parameter, date from and date to in format: [dateFrom, dateTo]. Dates are optional so it can be used like this.* *[,] – will return records with all dates* *[dateFrom,] – will return records from this date* *[,dateTo] – will return records to this date* *[dateFrom, dateTo] will return records between those 2 dates* *Records per page parameter. 1 is first 500, 2 is next 500 and so on.*
Request Example:	This will search database for emails that ends with arb.com and all dates: http://localhost:50498/Service1.svc/IDP/Query/1?part=1&email=*ARB.com&date=[,] This will search database for emails that ends with arb.com and dates between 01.12.2018 and 31.12.2018: http://localhost:50498/Service1.svc/IDP/Query/1?part=1&email=*ARB.com&date=[01.12.2018, 31.12.2018]	
Response:	500 records with structure: "EmailAddress": "a.bouhdary@ARB.com", "LearningPlanActivityIDOutsideSystem": null, "LearningPlanEndDate": "05.12.2018", "LearningPlanPercentComplete": 0, "LearningPlanStartDate": "01.12.2018"	

`UpdateIDPRecord` [PUT] Update IDP plan.		
Address:	http://address/Service1.svc/`skills/{companyID}`	
Request Parameters:	**CompanyID**: **Jwt Token**: **`IndividualDevelopmentPlanDto`**:	*Value returned from Login response.* *Added to authorization header as bearer token.* *Json object sent in body request that contains values for update*

Request Example:	http://localhost:50498/Service1.svc/skills/1
	with body:
	{
	"EmailAddress": "adam.almonte@ARB.com",
	"LearningPlanActivityIDOutsideSystem": "LH_PL310e_Col15_EN",
	"LearningPlanEndDate": "11.05.2022",
	"LearningPlanPercentComplete": 0,
	"LearningPlanStartDate": "11.04.2022"
	}	

Response:	`Success or Error`

Personnel

GetPersonnel [GET] Query database (tblPeople)for Personnel by CompanyID. It will return 500 records per page.		
Address:	http://address/Service1.svc/`Personnel/Get/{companyID}?part={part}`	
Request Parameters:	**CompanyID**: **Jwt Token**: **Part**:	*Value returned from Login response.* *Added to authorization header as bearer token.* *Records per page parameter. 1 is first 500, 2 is next 500 and so on.*
Request Example:	http://localhost:50498/Service1.svc/Personnel/Get/1?part=1	

Response:	500 records with structure:
	{
	"PersonID": 4,
	"LastName": "Zuercher",
	"FirstName": "Kirsten",
	"EmailAddress": "kirsten.zuercher@ARB.com",
	"YourPersonID": "D028387",
	"GroupId": "1",
	"NextAvailDate": null,
	"Gender": "",
	"Company": "ARB (Schweiz) AG, Regensdorf",
	"BusinessPhone": "",
	"HomePhone": "",
	"MobilePhone": "",
	"FaxNumber": "",
	"Address": "",
	"City": "MEE CEC",
	"StateProvince": "",
	"ZIPPostalCode": "",
	"country_name": "",
	"WebPage": "",
	"Notes": "",
	"BusinessRegion": "CEC",
	"DepartmentName": "Switzerland",
	"JobTitleName": "Presales Specialist - Customer Engagement & Commerce - Sales and Service",
	"PeopleTypeName": "IC",
	"ManagersEmail": "",
	"IsActive": true,
	"JobRole": "Presales Specialist - Customer Engagement & Commerce - Sales and Service"
	}	

GetPersonnelByEmail [GET] Query database (tblPeople)for Personnel by CompanyID and email. It will return 500 records per page.		
Address:	http://address/Service1.svc/`Personnel/Get/Email/{companyID}?email={email}&part={part}`	
Request Parameters:	**CompanyID**: **Jwt Token**: **Part**:	*Value returned from Login response.* *Added to authorization header as bearer token.* *Records per page parameter. 1 is first 500, 2 is next 500 and so on.*

	Email:	*String parameter. Can be used to query email records like:* **gmail.com – it will return all records that contains gmail.com in emaiil field.* *john* - it will return all records where email starts with john.*
Request Example:	http://localhost:50498/Service1.svc/Personnel/Get/Email/1?email= kirsten.zuercher@ARB.com&part=1 http://localhost:50498/Service1.svc/Personnel/Get/Email/1?email= *@ARB.com&part=1	

Response:	500 records with structure:
	{
	    "PersonID": 4,
	    "LastName": "Zuercher",
	    "FirstName": "Kirsten",
	    "EmailAddress": "kirsten.zuercher@ARB.com",
	    "YourPersonID": "D028387",
	    "GroupId": "1",
	    "NextAvailDate": null,
	    "Gender": "",
	    "Company": "ARB (Schweiz) AG, Regensdorf",
	    "BusinessPhone": "",
	    "HomePhone": "",
	    "MobilePhone": "",
	    "FaxNumber": "",
	    "Address": "",
	    "City": "MEE CEC",
	    "StateProvince": "",
	    "ZIPPostalCode": "",
	    "country_name": "",
	    "WebPage": "",
	    "Notes": "",
	    "BusinessRegion": "CEC",
	    "DepartmentName": "Switzerland",
	    "JobTitleName": "Presales Specialist - Customer Engagement & Commerce - Sales and Service",
	    "PeopleTypeName": "IC",
	    "ManagersEmail": "",
	    "IsActive": true,
	    "JobRole": "Presales Specialist - Customer Engagement & Commerce - Sales and Service"
	  }	

`PersonnelInsert` [POST] Used to insert or modify records.		
Address:	http://address/Service1.svc/`Personnel/Insert/{companyID}`	
Request Parameters:	**CompanyID**: **Jwt Token**: **List<PersonnelModelDto>**	*Value returned from Login response.* *Added to authorization header as bearer token.* *Json sent as list of PersonelModelDto objects in body request that contains values for insert or update*

Request Example:	http://localhost:50498/Service1.svc/Personnel/Insert/1
	With body (json with list of Personnel objects):
	[
	  {
	    "PersonID": 5164,
	    "LastName": "Santos",
	    "FirstName": "Leandro",
	    "EmailAddress": "leandro.santos01@ARB.com",
	    "YourPersonID": "I839163",
	    "GroupId": "1",
	    "NextAvailDate": null,
	    "Gender": null,	

	"Company": "ARB Brasil Ltda. - Sao Paulo", "BusinessPhone": "", "HomePhone": "12345", "MobilePhone": "", "FaxNumber": "", "Address": null, "City": "Brazil", "StateProvince": null, "ZIPPostalCode": null, "country_name": "", "WebPage": null, "Notes": null, "BusinessRegion": "LA", "DepartmentName": "Brazil", "JobTitleName": "Presales Enterprise Architect", "PeopleTypeName": "EA", "ManagersEmail": "", "IsActive": true, "JobRole": "Presales Enterprise Architect" }, { "PersonID": 5165, "LastName": "Guazzelli", "FirstName": "Lucciano", "EmailAddress": "lucciano.guazzelli@ARB.com", "YourPersonID": "I861290", "GroupId": "1", "NextAvailDate": null, "Gender": "male12345678901234567890", "Company": "ARB Brasil Ltda. - Sao Paulo", "BusinessPhone": "", "HomePhone": "", "MobilePhone": "12345", "FaxNumber": "", "Address": null, "City": "Brazil", "StateProvince": null, "ZIPPostalCode": null, "country_name": "", "WebPage": null, "Notes": null, "BusinessRegion": "LA", "DepartmentName": "Brazil", "JobTitleName": "Presales Enterprise Architect", "PeopleTypeName": "EA", "ManagersEmail": "", "IsActive": true, "JobRole": "Presales Enterprise Architect" },]

Response:	If there is any errors response will return json list with PersonelReport objects that contains error description.
	
	Example:
	[
	{
	"EmailAddress": "leandro.santos01@ARB.com",
	"LastName": "Santos",
	"FirstName": "Leandro",
	"Message": "The Email exists for a different Login in the system already. The Email & Login combination must be unique"
	}
	]

	If its success empty object will be returned: [] with status 200 OK.

Scores

GetScores [GET] Query database (tblScores) for Scores by CompanyID. It will return 500 records per page.		
Address:	http://address/Service1.svc/Scores/Get/{companyID}?email={email}&skillId={skillId}&jobRoleId={jobRoleId}&businessRegionId={businessRegionId}&part={part}	
Request Parameters:	**CompanyID**: **Jwt Token**: **Email**: **SkillId:** **JobRoleId:** **BusinessRegionId:** **Part**:	*Value returned from Login response.* *Added to authorization header as bearer token.* *String parameter. Can be used to query email records like:* **gmail.com – it will return all records that contains gmail.com in emaiil field.* *john* - it will return all records where email starts with john.* *SkillID from tblScores. Its optional. Send 0 value to ignore.* *JobRoleId from tblPeople. Its optional. Send 0 value to ignore.* *BusinessRegionId from tblPeople. Its optional. Send 0 value to ignore.* *Records per page parameter. 1 is first 500, 2 is next 500 and so on.*
Request Example:	http://localhost:50498/Service1.svc/Scores/Get/1?email=maria*&skillId=1587&jobRoleId=9&businessRegionId=13&part=1	

Response:	500 records with structure:
	{
	"SkillID": 1587,
	"CategoryID": "Industry Solution Skills-Banking-DigCust",
	"Skill": "Contextual Marketing for Retail & Commercial Banking",
	"IsAttribute": false,
	"PersonID": 588,
	"ScoringPersonID": 330,
	"Score": 2,
	"UpdateDate": "/Date(1481151600000+0100)/",
	"EvalType": 2,
	"SkillYearsOfExperience": null,
	"YourPersonID": "I056265",
	"NextAvailDate": null,
	"LastName": "Fernandez-Blanco",
	"FirstName": "Maria",
	"EmailAddress": "maria.fernandez-blanco@ARB.com",
	"JobRoleId": 9,
	"JobRole": "Presales Specialist - Banking Solutions - Retail & Commercial Banking",
	"BusinessRegionId": 13,
	"BusinessRegion": "EMEA South"
	}	