Syndy Retailer API

The Syndy Platform

Built from the ground up, the Syndy platform provides an efficient solution for distributing any kind of product content. The fact that it can potentially deal with every possible product category places some unique constraints on the way its product content is structured.

For most raw data, the Syndy platform relies on its dynamic templating technology. This enables retailers to very precisely articulate their data needs, while remaining flexible: a template can be updated at any time.

Syndy API Principles

The Syndy API is based on RESTful principles. This means that in general, the assumption should be that it adheres to HTTP standards. Wherever this is not the case, we will do our best to explicitly mention it.

API Applications

The Syndy API is accessible only by stakeholders with a valid Syndy API Application. The API application is the entity that governs the access type and level of the caller. Rate limits and access tokens are attributed to API applications rather than directly to the user account or its governing entity (e.g. the retailer, or a manufacturer). Ownership of an API application can belong to any valid platform stakeholder (e.g. a user, brand, or retailer), but will usually belong to the governing entity. In the case of a retailer, the API application will typically belong to the retailer entity.

Properties of each API application are as follows:

- API credentials
- Rate Limits
- Stakeholder

Authentication

Authentication for the Syndy API is loosely based on OAuth 2.0 principles. With every API application comes a set of API credentials that can be exchanged for a Syndy API access token. The access token is transmitted with each API request in the Authorization HTTP request header.

Access tokens have an expiry date and a 401 HTTP status code can be returned to indicate that a new access token should be obtained.

Rate limiting

Syndy employs rate limiting to prevent service degradation due to too many requests. Rate limits are variable and will depend on agreements between retailer and Syndy. Rate limit information is provided through HTTP response headers with every request. The headers made available through the Syndy API are as follows:

- `X-RateLimit-Limit`
- `X-RateLimit-Remaining`

Security

All requests to the Syndy API take place through SSL.

HTTP Headers

As per REST principles, the Syndy API attempts to use HTTP headers sensibly so individual requests are not bloated with common, generic request data. Below follows an overview of the various request- and response headers used by the Syndy API.

Request Headers

Mandatory	
Authorization	The Authorization header is used for both authentication and authorization. For most requests this will contain an Access Token, but when authenticating, API credentials must be provided here. For more information view the Authentication chapter.
Accept-Language	Mandatory for all localizable resources, such as “Get Product By Id”. Specifies the language the content should be returned in.
Optional	
X-Requested-With	Use this header to tell us which SDK, or custom software is used to execute the request with. (e.g. Syndy API .NET SDK 2.0.1.1283)
Content-Type	Currently only *application/json* is supported, but this header can be used to request XML in the future.
Accept-Encoding	GZIP, not supported yet.
If-Modified-Since	Not yet supported, allows API to return a 304 response code if a resource has not been updated since the requested timestamp.

Response Headers

Content-Language	Returned for localizable resources, indicating the language of the content.
Content-Type	application/json
Last-Modified	Specified for resources that are eligible for change tracking, e.g. “Get Product By Id”
X-RateLimit-Limit	The maximum rate limit allotment for the requesting application
X-RateLimit-Remaining	The amount of requests remaining in the current rate limit time window

OData

The Syndy API attempts, where possible, to adhere to OData querying standards for filtering and pagination. For a complete reference on the principles of OData refer to http://www.odata.org/.

Listed below are the commonly used OData parameters supported by the Syndy API.

Pagination	
$skip	In the context of a paginated result set, this parameter indicates where to start looking in the resource collection.
$top	In the context of a paginated result set, this parameter indicates how many items must be included in the returned result set.

Signature Generation

A signature must be generated for the Authorization header in the case of the Authentication request. Signature generation takes place in order to “sign” the request using the application’s private key, without directly transmitting the private key.

How to generate the signature

Generate a SHA512 hash of the following parameters in the specified order. The parameters should be concatenated and not delimited using any separating character. When extracting the byte array from the concatenated string, UTF8 decoding should be used. The SHA512 hash should be Base64 encoded.

Parameters for signature generation	
Private Key	Unique per application, provided on sign-up.
Http verb	E.g.: GET, POST, PUT, DELETE
Request Url path	The path component of the requested resource, excluding querystring e.g.: /auth
Querystring	Sorted alphabetically with keys and values url (percent) encoded. Encoding should occur according to RFC3986’s guidelines on percent-encoding. Also note that hexadecimal numbers must be in lowercase. E.g. %3a instead of %3A.
Nonce	A client generated token, unique per request e.g.: bb09a7e6-0dca-411b-b32b-351a173547d2.
Timestamp	The client timestamp in Unix UTC time.

A sample concatenated string for a search-manufacturer-by-name request might look something like this:

```
MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAOUspRHGNhIMnZ6say4k3swVMfKER8UYbVOREqq7t71VL89bagEFZW4rSjkTcBg7fVZCWX5ijgQVEX4xfV5lapcCAwEAAQ==GET/authefec3dc1-9a99-4825-9e09-3e7772fc205d1456823909
```

The generated signature from this string will then be:

```
RK+xg4V0hfgKs27ORFA2mVkXhYIX14zRH/f5bl6KqZB8zxJNgDpC6yHVIlN0uPuDXsR0cfdU9jAM1GdG7yzUSg==
```

Syndy API Resources

The Syndy API distinguishes between various different resources. A *resource* can be defined as a "domain of related entities and actions". Singular and plural notations of resources highlight important differences between resources. The singular "retailer" resource is the access point for actions available only for the requesting retailer. In contrast, the plural "retailers" resource is the access point for actions against the entire set of retailers, for example, to request the details of a specified retailer.

Every request to the API starts by specifying the resource in the path portion of the request URL. Here follows a list of resources and their properties:

- AUTH
  The authentication resource is a special resource in the sense that it does not embody any data entity, but instead provides only actions related to authentication and authorization. The following example authenticates and returns an access token:
  `GET /auth`

- PRODUCTS
  The products resource deals with requests related to the retrieval of information about products. The following example gets the product details for the specified product id:
  `GET /products/42ca3c4e-345c-419e-b4ea-e62d8b825837`

- RETAILER
  The retailer resource is the entry point for the various retailer-specific components of the Syndy platform. The following example is the action method for getting the retailer assortment.
  `GET /retailer/assortment`

- MARKETS
  The markets resource contains the full set of markets (countries) known to the Syndy platform. It is possible to filter on whether a market is unlocked or not. The following example gets the details for the specified market ID:
  `GET /markets/42ca3c4e-345c-419e-b4ea-e62d8b825837`

Syndy API Common Types

The Syndy API common types are designed to be reusable components that can either form a response on their own, or that can be found within other data types. For example, you will typically never encounter a Barcode data type on its own, but as a member of the various Product-related entities you will encounter it frequently.

Common Data Types

Barcode

The barcode is a special data type and is represented as an object. Since barcodes come in many shapes and sizes, the purpose of representing barcodes as objects is to provide future support for adding additional barcode type information (e.g. GTIN8, EAN-13, UPC-12, etc.)

```
{
	Value: "8712038123820"
}
```

PagedResultSet

The PagedResultSet data type is introduced to provide a consistent way for API requests that support pagination to return a result set that is broken up into separate pages.

```
{
	TotalCount: 350,
	Offset: 0,
	ResultCount: 10,
	Results: [
	]
}
```

ProductReference

The ProductReference data type represents a single product without any of its data attached. A product reference is typically returned in situations where it is unnecessary to have any more information about a product other than its identifiers. Other product representations typically inherit from ProductReference and will thus also include ProductReference's attributes.

```
{
	Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
	DateLastUpdate: "20160118T082216Z",
	Barcode: {
		Value: "8712031283421"
	}
}
```

RetailerProductReference

The RetailerProductReference inherits from the ProductReference entity, but adds the retailer-specific article number identifier.

```
{
	Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
	DateLastUpdate: "20160118T082216Z",
	Barcode: {
		Value: "8712031283421"
	},
	ArticleNumber: “207212”
}
```

Product

The product data type represents a single product, with its data attached through the Template attribute. A product is always uniquely identified by its ID, which is a GUID. Other identifiers such as the EAN code are also provided (through the Barcode property), but cannot be relied on to uniquely identify a product.

```
{
	Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
	Barcode: {
		Value: “8712038123820”
	},
	DateLastUpdate: "20160118T082216Z",
	Summary: {}, // ProductSummary entity
	Template: {} // ProductTemplate entity
}
```

ProductSummary

The ProductSummary entity provides a summary of basic metadata for a product. The summary can be seen as an extension of the product identification data and contains extra information such as brand, manufacturer, and product name.

```
{
	Name: “Coca Cola 50cL”,
	Description: “Coca Cola 50cL is a delicious soda that will make you smile!”,
	Image: {}, // Image entity
	Brand: {}  // BrandReference entity
}
```

Image

The Image entity is a convenience object that provides parameters to both uniquely identify the image as a resource (through the Id), as well as provide metadata through the Properties object.

```
{
	Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d ",
	OwnershipType: 0,
	Url: “http://domain.com/someimage.url”,
	Type: 10,
	Properties: {
```

```
            Width: 720,
            Height: 720
      }
}
```

BrandReference

The BrandReference entity provides basic identifying information about the brand of a product. It is comprised of an Id and Name only, but also includes a Manufacturer reference. This is to highlight the hierarchical nature of the relationship.

```
{
      Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
      Name: "Coca-Cola",
      Manufacturer: {} // ManufacturerReference
}
```

ManufacturerReference

The ManufacturerReference entity provides basic identifying information about the Manufacturer of a brand (and thus about the product belonging to that brand).

```
{
      Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
      Name: "Coca Cola Nederland B.V."
}
```

ProductTemplate

The *ProductTemplate* data type can be seen to act as a container for a set of fields with values. Although templates can be flattened into just a collection of fields, we've chosen to represent the *ProductTemplate* object separately in order to provide future opportunities for filtering responses based on template IDs.

The *ProductTemplate* object, next to its identifying metadata (id, name), contains a list of "child templates", each of which are *ProductTemplate* objects in themselves, and a list of Field objects, which contain the actual data for the requested product.

```
{
      Id: "5892cc11-c0db-47f9-8c38-cc72d9ff5c6d",
      Name: "COOP_Food_Basic",
      Children: [
            // Array of ProductTemplate entities
      ],
      Fields: [
            // Array of ProductTemplateField entities
      ]
}
```

ProductTemplateField

The ProductTemplateField data type is the most fluid data type in the sense that the various different types of fields have different object representations. For example, it is possible for a product field to be represented by an array of values (for example, a list of “USPs/Claims”). At the same time, it is possible for an array to itself consist of complex types rather than flat values (as is the case with the Nutrients field, which is an array of objects.)

```
{
	Id: "21676e57-40fd-4d42-993c-a2c5977daac2",
	Type: "string",
	Key: "human-readable-field-key",
	Data: {
		Value: "The value of the field, in this case, a string"
	}
}
```

The *Data* property deserves some special attention because its value is fluid. Depending on the value of the *Type* property, the *Data* can either contain a flat value, an array, or an object. This is why we refer to the value of the *Data* property as a ValueContainer. The ValueContainer always contains a *Value* property, the format of which is fluid (whose format changes based on the ProductTemplateField’s *Type* property).

Listed below follow all the possible types of value containers.

EnumValueContainer

The Enumeration Value Container is used for values that can be specified from a set of possible values. List of possible values can be requested separately.

```
{
	EnumId: "8bb207c2-7658-426c-8480-dade13b5a15a",
	Value: {
		Id: "37b3f9ac-857b-46ac-8a5e-bcb3bdf7f198",
		Value: "Energie"
	}
}
```

ArrayValueContainer

The Array Value Container is used as a container for collections of field values:

```
{
	Length: 5,
	Value: [
			// ProductTemplateField objects go here
	]
}
```

ObjectValueContainer

The Object Value Container indicates that the field that contains the value container is actually a complex field, consisting of a fixed number of child fields. The format of the value container is the same as the array value container, but where the array value container contains an arbitrary number of values, the object value container contains a fixed number of fields that together represent the object.

```
{
	Value: [
	]
}
```

FlatValueContainer

There are a number of "Flat" value containers, where the *Value* is neither an object nor an array. These flat values can be of various types, which will be indicated in the field definition. The following sample shows a *string* flat value container:

```
{
	Value: "The value of the field"
}
```

The field types that map onto a flat value container are as follows:

- `string`
- `double`
- `float`
- `decimal`
- `datetime`
- `timespan`
- `date`
- `int`
- `boolean`

AuthenticationResponse

The *AuthenticationResponse* entity is returned in response to the *authenticate* API request and is the container for the access token, and related attributes, such as expiry date.

```
{
	Token: "yRQYnWzskCZUxPwaQupWkiUzKELZ49eM7oWxAQK_ZXw",
	ApplicationId: "2b581bb2-0f16-4f0d-bab4-47ee9d19f6e7",
	DateExpires: "20160127T122134Z"
}
```

Syndy API Requests Documentation

Authenticate

The *Authenticate* API request is usually where a session with the Syndy API necessarily starts (because a previous access token expired, or because it is simply the first time you use the API).

The *Authenticate* request must be executed by the consumer of the Syndy API in order to obtain an access token, which is required for authentication with all other API requests. Access tokens produced by the *Authenticate* request are short-lived, and the consumer of the API is encouraged to take this into account from the very beginning. Code should be designed to always re-authenticate the moment the API responds with a 401 – Not Authorized response.

Resource

```
GET (https://api.syndy.com)/auth
```

Request Format

The request format in the case of an *Authenticate* request is a little different than other API requests. The Authorization header is used to authenticate the application, but because the assumption at this point must be that the application is not yet in the possession of a valid access token, the Authorization header is formatted differently, and instead of an access token & related data, now contains the application's *public key*.

The Authorization header must contain the following fields:

Key	The application's public key.
Timestamp	The UTC timestamp at which the request was created.
Nonce	A unique token that is used only once, that together with timestamp, prevents replay attacks.
Signature	The signature is generated using the application's private key. For information on how to generate the signature, refer to the chapter on signature creation.

Response Format

The response for the *Authenticate* call comes in the form of an AuthenticationResponse entity.

```
{
	Token: "yRQYnWzskCZUxPwaQupWkiUzKELZ49eM7oWxAQK_ZXw",
	ApplicationId: "2b581bb2-0f16-4f0d-bab4-47ee9d19f6e7",
	DateExpires: "20160127T122134Z"
}
```

Get Retailer Assortment

The *retailer* resource provides a "Get Assortment" method in order to access the full range of products in a retailer's assortment. Through the use of filtering this request can be adjusted to return only the range of products that have been changed since a specified "datefrom" time stamp.

Resource

`GET (https://api.syndy.com)/retailer/assortment`

Request Format

The retailer assortment request supports the following request parameters, specified through the resource URL's query string.

Parameters for pagination	
$skip	Offset controls the starting point within the assortment to start returning results
$top	Limit controls the amount of items to return from the assortment
Request-specific parameters	
$filter	Use to filter result set. Example usage could be to request only those products that have seen updates since specified ISO-8601 compliant datetime string (e.g. "2016-01-19T08:31:23.00Z")

Example: `GET /retailer/assortment?$filter=DateLastUpdate gt DateTime'2016-01-19T08:31:23.00Z'&$skip=0&$top=50`

Fetches the first 50 results from the retailer assortment that have been modified since Jan 19th, 2016, 08:31:23 UTC.

Response Format

NOTE: The response entity is still subject to change

The response format is a paged result set, where the results array consists of RetailerProductReference objects. Example:

```
{
	TotalCount: 350,
	Offset: 303,
	ResultCount: 47,
	Results: [
		{
			Id: "92jf0jf0j2-f920jf-2f0j23fj-f032f9j23fj",
			Barcode: {
				Value: "8712038123820"
			},
			ArticleNumber: "2813802",
			DateLastUpdate: "20160119T083123Z"
		},
		etc...
	]
}
```

Get Product Details

The *products* resource provides a “Get Product By Id” method in order to access the full product data template for a product identified by the specified id.

Resource

`GET (https://api.syndy.com)/product/{productId}`

Request Format

This method currently supports no additional request parameters.

Response Format

NOTE: The response entity is still subject to change

The response format is a Product entity. Sample format is shown below:

```
{
	Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
	DateLastUpdate: "20160118T082216Z",
	Summary: {
		Name: “Coca Cola 50cL”,
		Description: “Coca Cola 50cL is a delicious soda that will make you
smile!”,
		Image: {
			Id: “5d3f146d-0ff5-4de5-8100-54c0845b359d”,
			Width: 512,
			Height: 512,
			Url: “http://cdn.syndy.com/media/products/{id}/oajofijwaofj.jpg”
		},
		Brand: {
			Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
			Name: "Coca-Cola",
			Manufacturer: {
				Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
				Name: "Coca-Cola Nederland B.V."
			}
		}
	},
	Template: {
		Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
		Name: "Food_Template",
		Children: [
		],
		Fields: [
			{
				Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
				Type: "string",
				Key: "product_name",
				Data: {
					Value: "Coca Cola 50cL"
				}
			},
```

```
                {
                        Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
                        Type: "array",
                        Key: "usps_claims",
                        Data: {
                                Length: 1,
                                Value: [
                                {
                                        Id: "5d3f146d-0ff5-4de5-8100-54c0845b359d",
                                        Type: "string",
                                        Key: "usp_claim",
                                        Data: {
                                                Value: "Super sparkly"
                                        }
                                }
                                ]
                        }

                }

        ]

    }

}
```

Get Product Media

The *media* resource provides a “Get Media For Products” method in order to access the media for the specified product id.

Resource

`GET (https://api.syndy.com)/product/ {productId}/media`

Request Format

OData parameters for paging.

Response Format

NOTE: The response entity is still subject to change

```
{
	TotalCount: 11,
	Offset: 0,
	ResultCount: 10,
	Results: [
		{
			Id: "92jf0jf0j2-f920jf-2f0j23fj-f032f9j23fj",
			OwnershipType: 0,
			Url: “http://domain.com/someimage.url”,
			Type: 10,
			Properties: {
				Width: 720,
				Height: 720
			}
		},
		etc...
	]
}
```

Product Media – Image Resize

If the standard dimensions of a product image are not close enough to certain specifications, it is possible to instruct the Syndy API to resize product images to more closely meet said specifications.

Note: it is currently only possible to resize images if the new dimensions maintain the aspect ratio of the original image. Also note that the resizing happens asynchronously. URLs will be allocated, but it may take a little bit of time before the image is actually present at the specified location.

Resource

`POST (https://api.syndy.com)/products/media/{mediaId}/resize`

Request Format

The request is, at its core, a list of resolutions to which the original image should be resized. The to-be-resized image is identified by the mediaId specified in the resource URL.

```
{
	“MimeType”: “image/png”,
	“Sizes”: [
		{
			“Width”: 270,		// Width in pixels
			“Height”: 270		// Height in pixels
		},
		etc…
	]
}
```

Response Format

In case of a successful operation, the response format is very similar to the request:

```
{
	“Resources”: [
		{
			“Id”: “adfj0f-0asf00-asdf0sf-70dsf0sf70asfu”,
			“Url”: https://cdn.syndy.com/resizedURL,
			“Properties”: {
				“Width”: 270,
				“Height”: 270
			}
		},
		etc…
	]
}
```