API Reference

NEH - Nordisk E-handel

eValent Group AB

Version 32

CONTENTS

CHAPTER 1

INTRODUCTION

This document describes a web-based API for accessing and manipulating objects in the shop system.

The architecture of this API is heavily inspired by the REST architectural style (see [REST at Wikipedia]) but doesn't follow the philosophy to the letter. Simplifications and changes have been made where necessary.

This API is designed to use standardized and proven technologies and to be simple to implement. To achieve this the API is based on the HTTP 1.1 protocol which has been running the web for over a decade. The API uses XML as a data transfer syntax, for everything except pure binary data, which has the desired properties of being well structured and human readable.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In some of the URIs in the text below "\d+" may occur. This is a regular expression way to write an arbitrary number containing at least one digit.

The main features of this API are:

Using the HTTP protocol

The HTTP 1.1 protocol as defined in [RFC2616] is a very powerful protocol for transferring and manipulating resources of various types. The protocol is most well known for retrieving resources(web pages, images, documents, etc) on the web, but this is a very limited use of the facilities offered by HTTP.

By using the HTTP protocol this API is using a well tested and standardized protocol that every system on the internet supports. As an extra bonus this allows the developer to use a standard

web browser to access the API and look at the objects. This aids in development and debugging since it gives the developer a very simple method to see what is sent over the wire.

Objects as XML

Almost all objects in the API are serialized using the XML language defined in [XML1.0]. The only exceptions are binary objects like images.

Using XML is beneficial since most platforms have excellent support for XML. The plain text nature of XML makes the objects human readable which aids in development and debugging and enables the developer to create the objects in a very simple manner using any means he chooses ranging from high-end XML tool kits to XML documents created by hand.

Similar objects for all operation

The flexibility of XML enables the API to use almost exactly the same object definition when retrieving, creating or updating a resource.

This allows the developer to simply retrieve an object, remove elements he does not wish to change and send the updated object back to the server.

By using the same syntax in both directions the developer will rarely need to dig through documentation to determine which fields exist. He only needs to retrieve the resource and look at what the server sends. Great care has been taken when choosing the element names to reflect the data contained within the element.

Partial Updates

All objects have XML elements that are optional, very few mandatory elements are specified. This enables the client to only send those elements it wishes to update and omit the rest. The system will leave all omitted attribute at their current values. See section 4.4 for more details.

Using partial updates thus allows the client to send minimal objects requiring much less bandwidth while adding simplicity.

CHAPTER 2

API FUNDAMENTALS

2.1 Application Protocol

The protocol used is HTTP version 1.1 as defined in [RFC2616]. Unlike RPC-like APIs based on `SOAP` or `XML-RPC` this API makes full use of the HTTP protocol as an application protocol and not just a transport tunnel. SSL is used to keep the connections secure.

Although on the web rarely more than `GET` and `POST` are used, HTTP supports many other request methods. This API makes use of: `GET` for object retrieval, `PUT` for object update, `POST` for object creation and `DELETE` for object deletion.

The use of HTTP has the added benefit that it is trivial to use a standard web browser as a read-only client. Using other HTTP tools, like *cURL*, one can experiment with all features of this API in a trivial manner. Most programming languages contain HTTP and XML tools that make interfacing with this API simple.

The HTTP status codes are used to inform the client of the outcome of an operation as defined in [RFC2616] section 10. Not all HTTP status codes are used in this API. The codes used for each method are specified below as the method is described. Basically a response code between 200 and 300 means the request was successful, and a response code of 400 or greater means there was some error.

2.2 Character encoding

All exchanges MUST take place using the “ISO 8859-1”, also known as “LATIN-1”, character encoding.

2.2.1 XML Messages

The XML messages are sent as binary 8bit documents in the "ISO 8859-1" character encoding. In addition the XML MUST contain a valid XML header specifying the charset:
`<?xml version="1.0" encoding="ISO-8859-1"?>`

2.2.2 URI

The URI string and any and all query parameters MUST be in ISO-8859-1. However the URI itself cannot contain raw 8bit characters. The query parameters sent over the wire MUST NOT contain any characters outside the "US-ASCII" character set. Any characters present in "ISO-8859-1" and not present in "US-ASCII" MUST be %-encoded according to the rules of [RFC3986]. For example, the parameter "foo=bar" is valid ASCII and can be sent as is, however the parameter "foo=åäö" is NOT valid ASCII and MUST be %-encoded to yield "foo=%e5%e4%f6".

2.3 Resources

A resource is an object of some type that is identified by an URI. The full URI syntax is defined in [RFC3986] but this API uses only a subset of the full syntax.

A complete URI may look like:

```
https://apitest.testbutiken.se/__API__/customer
```

where the `https://` part is the protocol specification, the `apitest.testbutiken.se` is the domain name of the shop being accessed and `/__API__/customer` is the path to the customer resource.

All URIs used in this API are relative URIs containing only the path component which must be specified as an absolute path. This is so the URI can never reference resources that are external to the shop.[1]

It is also required that the paths sent be absolute paths. The full reference resolution algorithm specified in [RFC3986] section 5 is NOT used by the server.

If the client wishes to convert the relative URI (path) used in the API into a complete URI simply prepend the protocol specification and domain used to access the API to the relative URI. If, for example, the shop resides at `apitest.testbutiken.se` and the protocol used is `https` and the relative URI to be resolved is `/__API__/picture/4` then the full URI will be `https://apitest.testbutiken.se/__API__/picture/4`.

[1] Some fields MAY contain full URIs referencing external entities, but these fields are just treated as data by the API and have no special significance within it.

The URIs reported in the API (for example the URI for a specific picture) MUST be treated as opaque strings. There URIs contain no structure that is usable by the client and MUST NOT be interpreted. For example the URI `/__API__/picture/4` MUST NOT be interpreted to mean a picture with ID 4. The URI MUST be treated as the opaque string "/__API__/picture/4" that in its entirety identifies the picture.

2.4 Collections

A collection is a resource that contains or references other resources. For example the "customer" collection (see 3.2.1) lists and references all customers in the system.

Collections can list the resources either in their full form or an abbreviated form depending on the size of the objects in question. The form used is specified in the description for each collection in chapter 3. Most collections support the use of the query parameter `view=long` to ask the collection for the full objects instead of their abbreviated forms, however some collections still impose some limits on the objects sent using `view=long`. This is to lessen the server load and bandwidth requirements.

The objects in the collection usually have a `href=<uri>` attribute containing a *path* referencing this particular item that the client may use to retrieve, update or delete the item.

2.5 Authentication & Authorization

Authentication is performed using the "Basic Authentication Scheme" defined in [RFC2617] using an username and password configured in the shop system.

If the username or password is absent or incorrect, the system will respond with a UNAUTHORIZED(401) message according to [RFC2616] section 10.4.2.

Each user also has an associated set of rights that limit which resources are accessible and what the user may do to them. If the username and password are correct but this user does not have sufficient rights to perform the requested operation, the system will respond with a FORBIDDEN(403) message according to [RFC2616] section 10.4.4.

Since all access to the API is through an encrypted (*https*) connection the passwords are never sent in plain text.

2.6 Language support

Language support is still experimental and incomplete! The language support will most likely change with time until it is finalized. At this point implementing clients that utilize the language support IS NOT RECOMMENDED! And no support will be provided for such clients.

The API user can be configured to support multiple languages and with a primary language. This is done in the shop admin panel by the shop owner. If the API user IS NOT configured with multiple language support then language tags WILL NOT be sent from the server and will be ignored if received. This is to keep backwards compatibility with older versions of this API that didn't have language support.

The API client can determine if the shop and if the current user supports multiple languages by examining the element "MultipleLanguageSupport" in the "Options" and "APIUser" sections of the Root object. (See section 3.1).

For users supporting multiple languages each element that can vary based on language will be sent multiple times with different language attributes. The attribute is "lang" and contains a lower case *ISO-639-1* language code.

The attribute "primary-language" is sent and set to "true" for elements in the primary language set for this API user[2]

For example, the field "Name" without language support would look like:

```
<Name>named thing</Name>
```

With multiple language support and two languages (sv, en) the XML will look like:

```
<Name lang="sv">Namngiven sak</Name>
<Name lang="en" primary-language="true">named thing</Name>
```

If the API user doesn't wish to support multiple languages then the user MUST be configured with one language in the shop administration, and the client MUST NOT send elements with language tags.

A shop may have multiple API users that handle different languages if so required. In this case each user should be configured without multiple language support and with the correct primary language.

[2]This may be different than the primary language for the shop.

The elements that can have multiple language values are marked with a *+lang* value in the type field of the resource specifications in this manual.

2.6.1 Requirements for clients supporting multiple languages

Clients implementing multiple language support MUST be able to receive ALL elements with language tags, even elements that are not specified as supporting multiple languages in this manual. This is for compatibility with future versions of the API where more / other elements MAY be language dependent.

If the client sends a language enabled field without a "lang" attribute the server will either update only the language that the clients API user is configured for, or will update all languages available. The shop owner configures this behaviour for each API user individually. The "APIUser" section of the "Root" object has an element "UpdateAllLanguages" that reflects this setting.

The server will ignore the "primary-language" attribute if sent by the client.

The recommended method of parsing elements when a client supports multiple languages is outlined below:

1. If the client receives an element WITHOUT a "lang" attribute this element is to be treated as a non languaged element and handled according to the clients wishes.
2. If the client receives an element WITH a "lang" attribute AND the client supports multiple languages for this element the element is treated according to the clients wishes.
3. If the client receives an element WITH a "lang" attrubte and the client DOES NOT support multiple languages for this element:
   (a) If the element contains a "primary-language" attribute the element is to be treated exactly as in Case 1 above.
   (b) If the element DOES NOT contain a "primary-language" attribute the element MUST be ignored.

This procedure is to ensure that clients stay compatible with future versions of this API without the client implementor having to update the client.

2.7 Query parameters

If a query is provided in the URI according to [RFC3986] section 3.4 it is parsed into subcomponents, called "Query parameters" or "Parameters" in the form of `name=value` pairs separated by an ampersand(`&`) as is common on the web.

In the URI `http://where.com/ever?some=1&thing=2` the query is `some=1&thing=2` and the two parameters are `some=1` and `thing=2` respectively.

All supported parameters are listed in the `<METHOD> Parameters` lines in the resource descriptions in chapter 3.

NOTE that query parameters need to be properly encoded if they contain characters not part of the “US-ASCII” character set. See 2.2 for details.

2.8 Resource Operations

2.8.1 Retrieving a Resource

To retrieve an object the `GET` method is used with an URI referencing the object. The URI MAY contain a query component, which will be interpreted according to section 2.7. The client MUST NOT send any data in the body of the request.

Neither conditional nor partial GET requests are supported and the result of sending these headers is undefined.

For more details see [RFC2616] section 9.3.

Possible responses are:

Code	Name	[RFC2616]	Description
200	OK	10.2.1	The request completed successfully and the server is sending a response. The Content-Type header contains the type of the response, which is usually "text/xml".
204	No Content	10.2.5	The request completed successfully but the server has no response to send.
400	Bad request	10.4.1	The request was invalid for some reason. The status line will contain more information, and a response body may also be provided containing details about the problem.
401	Unauthorized	10.4.2	The request did not contain a correct username or password.
403	Forbidden	10.4.4	The request contained a correct username and password, but the user is not allowed to access this resource.
404	Not Found	10.4.5	The URI references a non-existent resource.
405	Method Not Allowed	10.4.6	There is no Retrieve operation defined for this resource.
500	Internal Server Error	10.5.1	There was some internal error in the server that prevented it from processing this request. This error SHOULD be reported with accurate information about date and time and the nature of the request.

2.8.2 Creating a Resource

To create a resource the POST method is used with an URI pointing to a resource accepting new subresources, usually a collection. The URI MAY contain a query component, which will be interpreted according to section 2.7. The new resource is sent in the body of the request. POST is defined in [RFC2616] section 9.5.

The object to be created is POSTed to the URI specified in the resource description in chapter 3. The URI will usually refer to a collection, and not to a specific object. This is understood to mean that you "post a new object into the collection". The system will respond with one of the status codes below and will usually provide a "Location:" header containing the URI at which the newly created object is located.

Most resources allow partial objects to be posted, and the remaining items will be set to their default values.

Possible responses are:

Code	Name	[RFC2616]	Description
201	Created	10.2.2	The request completed successfully and the new resource has been created. A "Location:" header provides the URI to the new resource. The response MAY also contain the new resource as its body.
400	Bad request	10.4.1	The request was invalid for some reason. The status line will contain more information, and a response body may also be provided containing details about the problem.
401	Unauthorized	10.4.2	The request did not contain a correct username or password.
403	Forbidden	10.4.4	The request contained a correct username and password, but the user is not allowed to access this resource.
404	Not Found	10.4.5	The URI references a non-existent resource.
405	Method Not Allowed	10.4.6	There is no Create operation defined for this resource.
500	Internal Server Error	10.5.1	There was some internal error in the server that prevented it from processing this request. This error SHOULD be reported with accurate information about date and time and the nature of the request.

2.8.3 Updating a Resource

To update a resource the PUT method is used with an URI identifying the resource to be updated, as described in [RFC2616] section 9.6. The URI MAY contain a query component, which will be interpreted according to section 2.7. The new contents of the resource is sent in the body of the request.

Most resources allow partial objects to be PUT, and will then only update the items that have been sent. Exceptions are noted in the resources description. See the partial update example in section 4.4.

The system will usually respond with code 200(OK) and return the updated object as it appears in the database to the client.

Possible responses are:

Code	Name	[RFC2616]	Description
200	OK	10.2.1	The request completed successfully and the server is sending a response. The Content-Type header contains the type of the response, which is usually “text/xml”.
204	No Content	10.2.5	The request completed successfully but the server has no response to send.
400	Bad request	10.4.1	The request was invalid for some reason. The status line will contain more information, and a response body may also be provided containing details about the problem.
401	Unauthorized	10.4.2	The request did not contain a correct username or password.
403	Forbidden	10.4.4	The request contained a correct username and password, but the user is not allowed to access this resource.
404	Not Found	10.4.5	The URI references a non-existent resource.
405	Method Not Allowed	10.4.6	There is no Update operation defined for this resource.
409	Conflict	10.4.10	The requested update conflicts with the current state of the resource and can not be completed.
500	Internal Server Error	10.5.1	There was some internal error in the server that prevented it from processing this request. This error SHOULD be reported with accurate information about date and time and the nature of the request.

2.8.4 Deleting a Resource

To delete a resource the DELETE method is used as described in [RFC2616] section 9.7 with an URI identifying the resource to be removed.

The client MUST NOT send a request body in a DELETE request. The only thing required is the resource URI for the resource to be deleted. Thus the HTTP request for deleting a particular customer with URI `/__API__/customer/42` will look like this:

```
DELETE /__API__/customer/42 HTTP/1.1
Authorization: Basic ...
Host: your.shop.hostname
```

The system will return a 204(No Content) response if the resource was successfully removed.

Possible responses are:

Code	Name	[RFC2616]	Description
204	No Content	10.2.5	The request completed successfully but the server has no response to send.
400	Bad request	10.4.1	The request was invalid for some reason. The status line will contain more information, and a response body may also be provided containing details about the problem.
401	Unauthorized	10.4.2	The request did not contain a correct username or password.
403	Forbidden	10.4.4	The request contained a correct username and password, but the user is not allowed to access this resource.
404	Not Found	10.4.5	The URI references a non-existent resource.
405	Method Not Allowed	10.4.6	There is no Delete operation defined for this resource.
409	Conflict	10.4.10	The requested update conflicts with the current state of the resource and can not be completed.
500	Internal Server Error	10.5.1	There was some internal error in the server that prevented it from processing this request. This error SHOULD be reported with accurate information about date and time and the nature of the request.

2.9 Error response

If the status code returned is 400 or greater then an error has occurred. The status line will contain additional information about the nature of the error.

The body of the response will contain an error object in XML that contains information about the error. The error document is made up of an `<Error>` element containing one or more `<Message>` elements that make up the lines of the error message. `<Message>` elements MAY contain a `code` attribute that specifies a numeric error code. Appendix A lists all error codes.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Error>
3   <Message code="1009">Typeerror: 'Product/Tax' is not 'float'</Message>
4   <Message code="1002">Unknown tag 'Product/BadTag'</Message>
5 </Error>
```

2.10 Filtering

Some collection resources have filtering capabilities. This is noted in the resource properties fields, if the field `filter` is listed then that resource supports filtering. Note that the filtering attribute is usually specified in the full object description, not the abbreviated description that usually appears in the documentation for each collection.

Filtering is done by constructing a filter expression and passing it as a query parameter of name `filter`[3].

Simple filter expressions are made up of a field name, a logical operator and a value: `field OP value` For example:

```
OrderNo = 140
```

or

```
Created > "2012-09-27T00:00:00Z"
```

The allowed operators are:

=	Equals
!=	Not equals
<	Less than
>	Greater than
<=	Less or equal
>=	Greater or equal

[3] Don't forget to URL-encode the data

Numeric values take the form of floats specified below. Strings MUST be enclosed with double quotes("). Any quotes within the string MUST be escaped by prepending a back-slash: `\"`. Boolean values are expressed as `true` or `false`, and empty values are expressed as `nil`.

Simple expressions can be combined using the `AND`, `OR`, and `NOT` keywords to form more complex expressions:

```
State = "NEW" AND Created > "2012-09-27T00:00:00Z"
```

Parentheses may be used to indicate precedence:

```
State = "NEW" OR (State = "OLD" AND Changed > "2012-09-27T00:00:00Z")
```

2.11 Synchronizing data

Often it is necessary to synchronize data between the shop and an external system. Usually this is data that is changed both in the shop and externally, like order states or customer information. Some objects in the shop support synchronization using the two methods outlined here.

These methods use the `ChangedTime` `CreatedTime` and `SyncedTime` attributes. Those objects that do not have these attributes present do not support synchronization[4], and some objects that do have these attributes may not yet implement all the required parameters.

The client-based method is preferred since it allows multiple clients to synchronize the objects, while the server-based will only allow one client to keep track of synchronization.

2.11.1 Client-based synchronization

Client-based synchronization uses the filtering function described above to filter based on the `ChangedTime` or `CreatedTime` attributes. These attributes keeps track of the date and time when this object was last updated or when the object was created.

When retrieving an object or collection the client simply records the `ChangedTime`. Then when it wishes to check for updated objects it simply retrieves the collection again and compares the `ChangedTime` for each object with the one it has stored locally.

This method can be further simplified by only storing the *latest* `ChangedTime` for the objects retrieved from a collection. Since all objects updated will by necessity get a `ChangedTime` that is later than the one recorded if they are updated after the synchronization is complete.

The filter `ChangedTime >= [datetime]` lists all those objects that have a `ChangedTime` attribute that indicates a time exactly equal to, or later than `<datetime>`. This is useful to avoid the possibility of a race condition, but it MAY also list objects that are in fact synchronized so the client should do some extra checks on the returned data.

[4] Over time more and more objects will be extended to support synchronization.

The filter `ChangedTime > [datetime]` lists all those objects that have a have a `ChangedTime` that is strictly later than `<datetime>`.

Please note that ANY change applied to a resource (be it due to user action, automatic scripts or other API clients) will update the `ChangedTime` attribute! The client MUST be able to correctly handle cases where objects previously synced are listed again.

The same method may be used to synchronize only newly created objects. The filter then uses the `CreatedTime` attribute instead of the `ChangedTime` attribute.

2.11.2 Server-based synchronization

API Users SHOULD use the Client-based synchronization method wherever possible! Using the Server-based method described here severely limits the possibility of having multiple systems that synchronize data with the web shop.

If asked to the server will store the synchronization time in the `SyncedTime` attribute. The client asks the server to do this by specifying `mark_as_synced=yes` in the URI query component. The server then stores the current time as the `SyncedTime` immediately after the update or retrieval is completed.

To retrieve only the objects that have been changed since the last synchronization the client simply adds `synced=no` to the query component. If the client wishes to get all objects that are in fact in sync then the query should be `synced=yes`.

Some objects support additional values for the `synced` parameter, and these values are listed in the resource specification.

2.12 Proxies and Gateways

This protocol fully supports proxies and gateways that obey the rules in [RFC2616]. However since all objects are mutable and may be changed at any time the server currently sends no-cache headers in the response.

2.13 Persistent connections and Pipelining

Persistent connections as defined in [RFC2616] section 8.1 are fully supported by the server. A client SHOULD use persistent connections whenever it wishes to send more than one request due to the huge improvement in efficiency they offer.

Pipelining requests as described in [RFC2616] section 8.1.2.2 is also fully supported, however the client MUST be careful to check the status of each request in turn, and to send requests in such a fashion that a pipelined request doesn't depend on the successful completion of a previous pipelined request. In practice this is rarely a problem.

These two methods will significantly improve performance on mass-update tasks like updating all customers, sending a new product inventory, or updating lots of product pictures. Therefore the client SHOULD use these methods whenever possible.

2.14 Object Syntax

Most of the objects are represented as XML documents according to [XML1.0].

2.14.1 XML Elements

Elements that are not marked as "mandatory" may be omitted. If an element is omitted the system will not change the value of that attribute. Thus a client can send only those elements that it wishes to modify to perform partial updates. See section 4.4 for examples of this.

If an element is present it may either be empty, with an `xsi:nil="true"` attribute representing a `NULL` value, or contain data in the format specified by the data type of the element. Not all elements actually support `NULL` values.

Since this API is in constant evolution and new features are added it is very likely that new elements will appear in the objects. Because of this clients MUST ignore any elements and attributes they do not recognize and MUST NOT send them back to the server in subsequent requests.

2.14.2 Data types

empty — Empty

An element marked as empty does not contain any data within itself but MAY contain an attribute, usually a `href` attribute containing an URI.

All empty elements SHOULD contain the `xsi:nil="true"` attribute.

Empty elements MAY be sent in any of the two forms below, but the first form is preferred since it is shorter:

```
<Empty xsi:nil="true" />
<AlsoEmpty xsi:nil="true"></AlsoEmpty>
```

str — String

Represents an arbitrary string.

Strings SHOULD be trimmed of leading and trailing white space when sending data to the server and MUST be trimmed when receiving. For example both these documents contain the same string, namely "A String" without leading och trailing white space:

```
<Element>A String</Element>

<Element>
     A String
</Element>
```

All strings are coded in the `ISO-8859-1` character set. The characters "<" and "`&`" MUST be encoded as XML entities according to [XML1.0] section 2.4. Any spaces appearing at the beginning or end of the string that the client doesn't wish to be trimmed MUST be sent as the HTML-entity "` `"

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to distinguish them from the empty string.

Examples:

```
<AString>A string, a string. My kingdom for a string!</AString>
<AnotherString>Testing... 1 2 3... testing</AnotherString>
```

int — Integer

Represents a positive or negative integer at least 32 bits wide. That is, it supports AT LEAST all the integers in the range: -2147483647 to 2147483647.

Positive integers are represented as their base-10 value in ASCII. Negative integers are represented as their absolute base-10 value in ASCII with a minus(-) sign prepended to it.

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to distinguish them from an actual integer value.

Examples:

```
<Positive>2</Positive>
<Negative>-2</Negative>
<VeryPositive>424242</VeryPositive>
<VeryNegative>-424242</VeryNegative>
```

float — Real numbers (floating point numbers)

A real valued positive or negative number.

All integers are valid floats, and are encoded as for `int`.

Real-valued numbers are encoded with their integer and decimal components separated by a decimal point(.). Negative values have a minus(-) sign prepended to them.

Real-valued numbers can also be expressed as a exponent value. The value have a integer and a decimal component separated by a decimal point(.). But it also have a exponent describing the base-10 for the entire value. 1230000 can be written as 1.23E+06 and 0.0000123 can be written as 1.23E-05.

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to distinguish them from an actual float value.

Examples:

```
<Positive>2.0</Positive>
<Negative>-2.0</Negative>
<Dotless>42</Dotless>
<BigExponent>1.23E+06</BigExponent>
<SmallExponent>1.23E-05</SmallExponent>
```

bool — Boolean

Represents a boolean value. This type is coded as `true` or `false` for true or false boolean values respectively.

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to signal that this element has no value, however for “bool” this is very rare.

Examples:

```
<ThisIsTrue>true</ThisIsTrue>
<ThisIsFalse>false</ThisIsFalse>
```

date — Date

The `date` type is a restricted form of the XML Schema date type defined in [XML Sch. P.2] section 3.2.9.

The format is `<YYYY>-<MM>-<DD>`:

1. Where YYYY is the year (0000-9999)
2. Where MM is the month (01-12)
3. Where DD is the day of the month (01-31).

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to signal that no date has been set.

Examples:

```
<Today>2009-10-21</Today>
<Tomorrow>2009-10-22</Tomorrow>
```

datetime — Date and time

The `datetime` type is a restricted form of the XML Schema datetime type defined in [XML Sch. P.2] section 3.2.7.

The format is `<date>T<time>Z` where T and Z are literals and:

1. `<date>` is formatted as: `YYYY-MM-DD` where YYYY is the year, MM is the month (01-12), DD is the day of the month (01-31).
2. `<time>` is formatted as: `HH:MM:SS` where HH is the hour (00-23), MM is the minute(00-59), SS is the second(00-59)[5]
3. The `<time>` is in UTC.

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to signal that no date has been set.

Examples:

```
<Today>2009-04-02T12:00:00Z</Today>
<Tomorrow>2009-04-03T12:00:00Z</Tomorrow>
```

[5] Leap seconds are not supported.

ccode — Country Code

This type is used to denote a two-letter country-code according to ISO-3166-1 standard.

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to signal that this element has no value.

Examples:

```
<Country>SE</Country>
```

lang — Language Code

This type is used to denote a two-letter language-code according to ISO-639-1 standard.

Examples:

```
<Language>sv</Language>
```

emailaddr — EMail Address

This type is used to transfer a standard `RFC2822` email address in the form most commonly used on the internet.

Some objects contain elements that may be undefined (NULL). Such elements MUST contain a `xsi:nil="true"` attribute to signal that this element has no value.

Examples:

```
<EMail>test@example.com</EMail>
```

2.15 Meta data

The meta data element is present as the first element in a list and contain useful information about the content of the list.

2.15.1 Pagination

List requests that support pagination return meta data as the first record containing information about the current page, the page size, number of records in the list and the total number of records available.

Query parameters

Name	Values	Description
page		The number of the current page.
page_size		Number of records per page.

page_size may depend on the implementation but is usually set to have a max value of 250 and a default value of 100.

Object description for “metadata-pagination”

Name	Type	Mode	Description
page	*int*	*read*	Number of the current page.
page_size	*int*	*read*	Max number of records in this list.
total	*int*	*read*	Total number of records matching the current selection.
count	*int*	*read*	Number of records in this list.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<VariantList>
  <meta>
    <page>1</page>
    <page_size>100</page_size>
    <total>2</total>
    <count>2</count>
  </meta>
  <Variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/product/8/variant/12" variant_id="12">
    <SKU>1</SKU>
    <Name lang="sv">Produkt nummer ett</Name>
    <Name lang="en" primary-language="true">Produkt nummer ett</Name>
    <State>ACTIVE</State>
    <CreatedTime>2007-11-22T07:00:12Z</CreatedTime>
    <ChangedTime>2012-09-06T09:14:40Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
  </Variant>
  <Variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/product/8/variant/27" variant_id="27">
    <SKU>1.1</SKU>
    <Name lang="sv">Produkt nummer ett A</Name>
    <Name lang="en" primary-language="true">Produkt nummer ett A</Name>
    <State>ACTIVE</State>
    <CreatedTime>2007-11-22T08:57:59Z</CreatedTime>
    <ChangedTime>2014-02-21T09:45:57Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
```

```
26   </Variant>
27   <Variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/product/8/variant/29" variant_id="29">
28     <SKU>1.1b</SKU>
29     <Name lang="sv">Produkt nummer ett B</Name>
30     <Name lang="en" primary-language="true">Produkt nummer ett B</Name>
31     <State>ACTIVE</State>
32     <CreatedTime>2009-04-07T14:37:20Z</CreatedTime>
33     <ChangedTime>2012-09-06T09:14:40Z</ChangedTime>
34     <SyncedTime xsi:nil="true" />
35   </Variant>
36 </VariantList>
```

CHAPTER 3

RESOURCES

3.1 Root

URI:	/__API__/
Actions:	Retrieve(GET)
Content-Type:	text/xml

The Root resource provides some basic shop information, including a list of supported options.

The Root resource also provides a list of resources available within this API that may be used as a starting point for discovering all the resources in the system. Most of the resources provided in the resource list will be collections containing objects and optionally sub-resources.

Object description

Name	Type	Mode	Description
APIVersion	*str*	*read*	The API version running on the server.
Shop	*str*	*read*	The shop identifier.
ShopName	*str*	*read*	The shop name.
ContactEmail	*emailaddr*	*read*	Contact email address.
ServiceEmail	*emailaddr*	*read*	Service email address.
DefaultCurrency	*str*	*read*	Default currency.
MaxNumberOfProductVariants	*int*	*read*	Maximum number of products and product variants supported by this shop. Creating more items than this will cause an error.
Options	*obj*	*read*	A list of true/false options indicating what this shop supports.
APIUser	*obj*	*read*	Information about the settings for the user accessing the API.
SupportedLanguages	*array*	*read*	An array of "Language" elements containing the ISO-639-1 language code for each language supported by this shop.
ResourceList	*str*	*read*	An array of "Resource" objects.

Object description for "Options"

Name	Type	Mode	Description
MultiplePricelists	*bool*	*read*	Indicates whether the shop supports multiple price lists or not.
PriceQty	*bool*	*read*	Prices based on the quantity purchased. For example, 1 for 10:-/each, 5 for 7:-/each.
CampaignPrices	*bool*	*read*	This shop supports simple campaign prices.
AdvancedCampaigns	*bool*	*read*	"advanced" campaign handling. Allowing multiple campaigns. Currently not supported by the API.
DiscountCodes	*bool*	*read*	Campaign / Discount codes.
Stock	*bool*	*read*	Stock information for products and support for different stock profiles.

contd...

Name	Type	Mode	Description
MultipleWarehouses	*bool*	*read*	Multiple warehouses for stock information.
DiscountGroups	*bool*	*read*	True if the shop supports DiscountGroups.
MultipleCategories	*bool*	*read*	True if a product can be a member of multiple categories.
DownloadableProducts	*bool*	*read*	True if this shop supports downloadable products.
EuroShop	*bool*	*read*	True if EuroShop support is enabled in this shop.
Invoicing	*bool*	*read*	True if invoicing support is enabled in this shop.
Subscriptions	*bool*	*read*	True if subscriptions are supported in this shop.
VariantGrouping	*bool*	*read*	True if variant grouping is supported by this shop.
Symbols	*bool*	*read*	True if symbols are supported by this shop.
DeliveryAddresses	*bool*	*read*	True if shop supports multiple delivery addresses per customer.
OrderEditing	*bool*	*read*	True if shop supports creating/editing orders.
MultipleLanguageSupport	*bool*	*read*	True if shop supports multiple languages.
OSS	*bool*	*read*	True if shop has OneStopShop enabled.
Voec	*bool*	*read*	True if shop has Voec for Norway enabled.
PrimaryLanguage	*lang*	*read*	The primary language of the shop.

Object description for "APIUser"

Name	Type	Mode	Description
PrimaryLanguage	*lang*	*read*	The primary language configured for this API user.
MultipleLanguageSupport	*bool*	*read*	True if this API user is configured for multiple language support.
UpdateAllLanguages	*bool*	*read*	True if this API user is configured to update all languages for an element when no "lang" attribute is sent by the client.

Object description for "Resource"

Name	Type	Mode	Description
Name	*str*	*read*	The name of the resource.
Description	*str*	*read*	A description of the resource.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <APIVersion>27</APIVersion>
  <Shop>apitest</Shop>
  <ShopName>the apitest shop</ShopName>
  <ContactEmail>test@example.com</ContactEmail>
  <ServiceEmail>test@example.com</ServiceEmail>
  <DefaultCurrency>SEK</DefaultCurrency>
  <MaxNumberOfProductVariants>5000</MaxNumberOfProductVariants>
  <Options>
```

```
    <MultiplePricelists>true</MultiplePricelists>
    <PriceQty>true</PriceQty>
    <CampaignPrices>true</CampaignPrices>
    <AdvancedCampaigns>true</AdvancedCampaigns>
    <CampaignCodes>true</CampaignCodes>
    <ProductVariantPicture>true</ProductVariantPicture>
    <Stock>true</Stock>
    <MultipleWarehouses>true</MultipleWarehouses>
    <DiscountGroups>true</DiscountGroups>
    <MultipleCategories>true</MultipleCategories>
    <LinkedProducts>true</LinkedProducts>
    <Documents>true</Documents>
    <DownloadableProducts>true</DownloadableProducts>
    <EuroShop>true</EuroShop>
    <Invoicing>true</Invoicing>
    <Subscriptions>true</Subscriptions>
    <VariantGrouping>true</VariantGrouping>
    <Symbols>true</Symbols>
    <DeliveryAddresses>true</DeliveryAddresses>
    <OrderEditing>true</OrderEditing>
    <MultipleLanguageSupport>true</MultipleLanguageSupport>
    <PrimaryLanguage>sv</PrimaryLanguage>
  </Options>
  <APIUser>
    <PrimaryLanguage>en</PrimaryLanguage>
    <MultipleLanguageSupport>true</MultipleLanguageSupport>
    <UpdateAllLanguages>false</UpdateAllLanguages>
  </APIUser>
  <SupportedLanguages>
    <Language>sv</Language>
    <Language>en</Language>
  </SupportedLanguages>
  <ResourceList>
    <Resource href="/__API__/unit">
      <Name>Unit Collection</Name>
      <Description>All units defined in the system.</Description>
    </Resource>
    <Resource href="/__API__/order/state">
      <Name>OrderState collection</Name>
      <Description>A collection of all the states an order may
          take.</Description>
    </Resource>
    <Resource href="/__API__/warehouse">
      <Name>Warehouse Collection</Name>
      <Description>All warehouses defined in the system.</Description>
    </Resource>
    <Resource href="/__API__/invoice">
      <Name>Invoice Collection</Name>
      <Description>A collection of all the invoices in the
          shop.</Description>
    </Resource>
```

```
60      <Resource href="/__API__/discountcode">
61        <Name>Discountcode Collection</Name>
62        <Description>All discountcodes defined in the system.</Description>
63      </Resource>
64      <Resource href="/__API__/category">
65        <Name>Category Collection</Name>
66        <Description>All categories defined in the system.</Description>
67      </Resource>
68      <Resource href="/__API__/customer/authcheck">
69        <Name>Customer Authentication test</Name>
70        <Description>This resource is used by API users to verify the
              login and passwords of a customer, to facilitate single-signon
              systems.</Description>
71      </Resource>
72      <Resource href="/__API__/brand">
73        <Name>Brand Collection</Name>
74        <Description>All brands defined in the system.</Description>
75      </Resource>
76      <Resource href="/__API__/transporter">
77        <Name>Transporter Collection</Name>
78        <Description>All transporters supported by this shop.</Description>
79      </Resource>
80      <Resource href="/__API__/paymentmethod">
81        <Name>Paymentmethod Collection</Name>
82        <Description>All payment methods configured in this
              shop.</Description>
83      </Resource>
84      <Resource href="/__API__/customercategory">
85        <Name>Customercategory Collection</Name>
86        <Description>All customercategorys defined in the
              system.</Description>
87      </Resource>
88      <Resource href="/__API__/shippingmethod">
89        <Name>Shippingmethod Collection</Name>
90        <Description>All shipping methods configured in this
              shop.</Description>
91      </Resource>
92      <Resource href="/__API__/affiliateprogram">
93        <Name>Affiliate Program Collection</Name>
94        <Description>Lists all affiliate programs.</Description>
95      </Resource>
96      <Resource href="/__API__/discountgroup">
97        <Name>Discountgroup Collection</Name>
98        <Description>All discountgroups defined in the
              system.</Description>
99      </Resource>
100     <Resource href="/__API__/customattribute">
101       <Name>Custom attributes collection</Name>
102       <Description>A collection of all the custom attributes that may be
              set on products and other places..</Description>
103     </Resource>
```

```
104     <Resource href="/__API__/subscription">
105       <Name>Subscription collection</Name>
106       <Description>A collection of all the subscriptions in the
              shop.</Description>
107     </Resource>
108     <Resource href="/__API__/pricelist">
109       <Name>Pricelist Collection</Name>
110       <Description>All pricelists defined in the system.</Description>
111     </Resource>
112     <Resource href="/__API__/stockprofile">
113       <Name>Stockprofile Collection</Name>
114       <Description>All stockprofiles defined in the system.</Description>
115     </Resource>
116     <Resource href="/__API__/product/downloadable">
117       <Name>Downloadable Product Files</Name>
118       <Description>Product files for products used to sell downloadable
              items.</Description>
119     </Resource>
120     <Resource href="/__API__/campaign">
121       <Name>Campaign Collection</Name>
122       <Description>All campaigns defined in the system.</Description>
123     </Resource>
124     <Resource href="/__API__/product">
125       <Name>Product Collection</Name>
126       <Description>All products defined in the system.</Description>
127     </Resource>
128     <Resource href="/__API__/picture">
129       <Name>Picture Collection</Name>
130       <Description>All product/category/etc pictures in the picture
              subsystem</Description>
131     </Resource>
132     <Resource href="/__API__/symbol">
133       <Name>Symbol Collection</Name>
134       <Description>Symbols for use on products.</Description>
135     </Resource>
136     <Resource href="/__API__/order">
137       <Name>Order Collection</Name>
138       <Description>A collection of all the orders in the
              shop.</Description>
139     </Resource>
140     <Resource href="/__API__/document">
141       <Name>Document Collection</Name>
142       <Description>All  documents in the document subsystem</Description>
143     </Resource>
144     <Resource href="/__API__/customer">
145       <Name>Customer Collection</Name>
146       <Description>A collection of all the customers in the
              shop.</Description>
147     </Resource>
148   </ResourceList>
149 </Root>
```

3.2 Customers

The customer collection specified below contains all customers in the shop. New customers can be created by POSTing a customer to the collection URI.

The customer resources support the synchronization procedure described in section 2.11.

3.2.1 Customer Collection

URI:	`/__API__/customer`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, view, mark_as_synced, filter
POST Parameters:	mark_as_synced, welcome_email, allocate_customer_no

Query parameters

Name	Values	Description
`synced`	*yes,no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`view`	*short,long*	If specified as "long" then a complete "Customer" object is sent for each customer. This is useful when exporting many customers.
`welcome_email`	*yes,no*	If specified as "yes" the system sends out a welcome email for this customer. If specified as "no" the welcome email is suppressed. If not specified at all, system defaults are used.
`allocate_customer_no`	*yes,no*	If specified as "yes" the system will allocate a new customer number for this customer

Example URIs

```
https://apitest.testbutiken.se/__API__/customer/?view=long
https://apitest.testbutiken.se/__API__/customer/?synced=yes
https://apitest.testbutiken.se/__API__/customer/?filter=ChangedTime>%3D"2012-01-01T01:01:01Z"
```

The API is stringent regarding the format of the time parameter, also note that the welcome email is sent only when a customer is created or updated.

Object description

The returned object `<CustomerList>...</CustomerList>` is an array of `Customer` [1] objects. The objects are abbreviated to only contain information needed to determine if a customer has been updated and needs to be fetched again.

The `<Customer>` tag contains an `href="<uri>"` attribute that specifies the URI where this customer can be fetched.

When creating a new customer with POST the complete `Customer` object is sent.

The abbreviated `Customer` object:

Name	Type	Mode	Description
`CustomerNo`	*str*	*read, write*	The customer id / customer number.
`Mode`	*str*	*read, write*	One of “ACTIVE” or “LOCKEDOUT”
`CustomerOrders`	*empty*	*read*	The “href” attribute contains a reference to a collection containing all orders made by this customer. See 3.2.3.
`Subscriptions`	*empty*	*read*	The “href” attribute contains a reference to a collection containing all subscriptions owned by this customer. See 3.2.4.
`PreferredDeliveryAddress`	*empty*	*read*	If the customer has set a preferred delivery address then the “href” attribute will contain the URI for that address.
`DeliveryAddresses`	*empty*	*read*	The “href” attribute contains a reference to a collection containing all delivery addresses this customer has on their profile. See 3.2.5.
`CreatedTime`	*datetime*	*read*	The date and time when this customer was created.
`ChangedTime`	*datetime*	*read*	The date and time when this customer record was last changed.
`SyncedTime`	*datetime*	*read*	The date and time when this customer record was last synchronized.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <CustomerList>
3   <Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       customer_id="1" href="/__API__/customer/1">
4     <CustomerNo>1</CustomerNo>
5     <Mode>ACTIVE</Mode>
6     <CustomerOrders href="/__API__/customer/1/orders" xsi:nil="true" />
7     <Subscriptions href="/__API__/customer/1/subscriptions"
         xsi:nil="true" />
```

[1] See section 3.2.2

```
8     <DeliveryAddresses href="/__API__/customer/1/deliveryaddress"
          xsi:nil="true" />
9     <PreferredDeliveryAddress
          href="/__API__/customer/1/deliveryaddress/1" xsi:nil="true" />
10    <CreatedTime>2009-04-05T08:22:06Z</CreatedTime>
11    <ChangedTime>2016-08-30T09:33:16Z</ChangedTime>
12    <SyncedTime>2010-02-03T17:51:23Z</SyncedTime>
13  </Customer>
14  <Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        customer_id="2" href="/__API__/customer/2">
15    <CustomerNo>2</CustomerNo>
16    <Mode>ACTIVE</Mode>
17    <CustomerOrders href="/__API__/customer/2/orders" xsi:nil="true" />
18    <Subscriptions href="/__API__/customer/2/subscriptions"
          xsi:nil="true" />
19    <DeliveryAddresses href="/__API__/customer/2/deliveryaddress"
          xsi:nil="true" />
20    <PreferredDeliveryAddress xsi:nil="true" />
21    <CreatedTime>2011-05-25T09:10:18Z</CreatedTime>
22    <ChangedTime>2012-02-10T13:11:57Z</ChangedTime>
23    <SyncedTime xsi:nil="true" />
24  </Customer>
25  <Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        customer_id="29" href="/__API__/customer/29">
26    <CustomerNo>10042</CustomerNo>
27    <Mode>ACTIVE</Mode>
28    <CustomerOrders href="/__API__/customer/29/orders" xsi:nil="true" />
29    <Subscriptions href="/__API__/customer/29/subscriptions"
          xsi:nil="true" />
30    <DeliveryAddresses href="/__API__/customer/29/deliveryaddress"
          xsi:nil="true" />
31    <PreferredDeliveryAddress xsi:nil="true" />
32    <CreatedTime>2012-09-11T12:01:36Z</CreatedTime>
33    <ChangedTime>2012-09-11T12:01:36Z</ChangedTime>
34    <SyncedTime xsi:nil="true" />
35  </Customer>
36  <Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        customer_id="30" href="/__API__/customer/30">
37    <CustomerNo>10060</CustomerNo>
38    <Mode>ACTIVE</Mode>
39    <CustomerOrders href="/__API__/customer/30/orders" xsi:nil="true" />
40    <Subscriptions href="/__API__/customer/30/subscriptions"
          xsi:nil="true" />
41    <DeliveryAddresses href="/__API__/customer/30/deliveryaddress"
          xsi:nil="true" />
42    <PreferredDeliveryAddress xsi:nil="true" />
43    <CreatedTime>2012-09-11T12:10:47Z</CreatedTime>
44    <ChangedTime>2012-09-11T12:19:35Z</ChangedTime>
45    <SyncedTime xsi:nil="true" />
46  </Customer>
47 </CustomerList>
```

3.2.2 Customer

URI:	`/__API__/customer/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
GET Parameters:	mark_as_synced
PUT Parameters:	mark_as_synced, welcome_email, allocate_customer_no
Attributes:	href, customer_id

Query parameters

Name	Values	Description
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`welcome_email`	*yes,no*	If specified as "yes" the system sends out a welcome email for this customer. If specified as "no" the welcome email is suppressed. If not specified at all, system defaults are used.
`allocate_customer_no`	*yes,no*	If specified as "yes" the system will allocate a new customer number for this customer

Object description

The `Customer` element contains a "href" attribute specifying the complete URI for this customer. It also contains a "customer_id" attribute that is the customer_id parameter used in the shop front end. This might be used by the API user to create links that link into a specific customer in the shop.

Note that most, but not all, shops are configured to use the email address as the customer login. For these shops the system will set "Login" and "EMail" equal. The "Login" element takes precedence if both are present in the update.

Name	Type	Mode	Description
`CustomerNo`	*str*	*read, write, filter*	The customer id / customer number.
`Type`	*str*	*read, write, filter*	Type of customer. Allowed values are: "PERSON", "COMPANY", "GOVERNMENT"

contd...

Name	Type	Mode	Description
OneTimeCustomer	*bool*	*read, filter*	Marks that this is a one-time customer created by the shop. One-time customer records are created on the fly during the purchase process if the customer doesn't wish to create an account. One-time customers are read-only. A client MUST NOT try to update the customer, or create one-time customers.
Mode	*str*	*read, write, filter*	One of "ACTIVE" or "LOCKEDOUT"
OrgNo	*str*	*read, write, filter*	The social security number or organization number.
VatNo	*str*	*read, write, filter*	The VAT number.
FirstName	*str*	*read, write*	Customer first name.
LastName	*str*	*read, write*	Customer last name.
Company	*str*	*read, write*	Customer company name.
Address1	*str*	*read, write*	Address line 1.
Address2	*str*	*read, write*	Address line 2.
ZIP	*str*	*read, write*	Postal code.
City	*str*	*read, write*	City.
State	*str*	*read, write*	Province or State.
Country	*ccode*	*read, write*	2-letter country code according to ISO-3166-1 as CAPITALS.
Language	*lang*	*read, write*	2-letter language code according to ISO-639-1.
PhoneNo	*str*	*read, write*	Phone number.
CellPhoneNo	*str*	*read, write*	Cellphone number.
FaxNo	*str*	*read, write*	Fax number.
EMail	*emailaddr*	*read, write, filter*	Email address of the customer. Note that most shops are configured to use the email address as login. For these shops the system will set "Login" and "EMail" equal. The "Login" element takes precedence here.
DiscountPercent	*str*	*read, write*	Discount in percent (%)
CreditLimit	*str*	*read, write*	The credit limit for this customer.
CreditUsed	*str*	*read, write*	How much of the credit limit is used.
Currency	*str*	*read, write*	3-letter alphabetic currency code according to ISO-4217 as CAPITALS. If empty it defaults to the shop default currency.
Login	*str*	*read, write, filter*	Login name for user. Required to be set for new customers and omitted or non-empty for updates. Note that most shops are configured to use the email address as login. For these shops the system will set "Login" and "EMail" equal. The "Login" element takes precedence here.

contd...

Name	Type	Mode	Description
Password	*str*	*write*	Password for customer. Minimum 3 characters. This field is "write-only", passwords can not be retrieved, only changed.
Comment	*str*	*read, write*	A comment about this customer.
WantsNewsletter	*bool*	*read, write, filter*	If customer wants a newsletter.
HaveSentWelcomeEmail	*bool*	*read*	Indicates wether a welcome email has been sent.
Pricelist	*empty*	*read, write*	A "href" attribute contains the URI for the referenced price list. The tag itself is empty. Sending the tag without the "href" attribute removes the pricelist setting from the customer. This field is NOT sent for "view=long" customer listings.
Customercategory	*empty*	*read, write*	A "href" attribute contains the URI for the referenced customer category. The tag itself is empty. Sending the tag without the "href" attribute removes the customer category setting from the customer. This field is NOT sent for "view=long" customer listings.
Warehouse	*empty*	*read, write*	A "href" attribute contains the URI for the referenced warehouse. The tag itself is empty. Sending the tag without the "href" attribute removes the warehouse setting from the customer. This field is NOT sent for "view=long" customer listings.
GroupDiscount	*array*	*read, write*	An array of items containing the discount for each discount group.
CustomerOrders	*empty*	*read*	The "href" attribute contains a reference to a collection containing all orders made by this customer. See 3.2.3.
Subscriptions	*empty*	*read*	The "href" attribute contains a reference to a collection containing all subscriptions owned by this customer. See 3.2.4.
PreferredDeliveryAddress	*empty*	*read, write*	If the customer has set a preferred delivery address then the "href" attribute will contain the URI for that address.
DeliveryAddresses	*empty*	*read*	The "href" attribute contains a reference to a collection containing all delivery addresses this customer has on their profile. See 3.2.5.
ExtraText1..6	*str*	*read, write, filter*	Extra text parameters.
ExtraSelector1..10	*int*	*read, write, filter*	Extra select fields, defined in the shop. Can only take on values between 0 and 256, unlike most integers. If set the attribute *textvalue* will contain a textual value corresponding to the numeric index.

contd...

Name	Type	Mode	Description
CreateLoginToken	*empty*	*read*	The "href" attribute contains the URI to use for login token creation. See section 3.2.8.
ViaAffiliate	*str*	*read, write*	The "href" attribute contains the URI of the affiliate that this customer came via. The element data contains the affiliate ID/tag when reading, but is ignored on update/create.
CustomAttributes	*array*	*read, write*	A list of custom attributes and the values they are set to. See section 3.2.2 and 3.11 for details.
Keys	*array*	*read, write*	A list of keys assigned to this customer. See section 3.2.2 and 3.26 for details.
CreatedTime	*datetime, filter*	*read*	The date and time when this customer was created.
ChangedTime	*datetime, filter*	*read*	The date and time when this customer record was last changed.
SyncedTime	*datetime, filter*	*read*	The date and time when this customer record was last synchronized.

Object description for "GroupDiscount" items

Name	Type	Mode	Description
DiscountGroup	*empty*	*read, write*	The tag contains a "href" attribute identifying the discount group in question.
DiscountPercent	*float*	*read, write*	The discount in percent.

Object description for "CustomAttributes"

An array of CustomAttribute blocks containing a name and value. The CustomAttribute tag has a "href" attribute that specifies which CustomAttribute this refers to. Changing the "href" tag is the only way to change the attribute.

Please note that this array MUST be sent in its entirety if it is sent at all since exactly those attributes and values listed here will be set.

Object description for "Keys"

An array of Keys blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
Index	*int*	*read, write*	The index for the key. 0..63
Value	*bool*	*read, write*	If key is active then true, otherwise false

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    customer_id="1" href="/__API__/customer/1">
  <CustomerNo>1</CustomerNo>
  <Type>PERSON</Type>
  <OneTimeCustomer>false</OneTimeCustomer>
  <Mode>ACTIVE</Mode>
  <OrgNo xsi:nil="true" />
  <VatNo xsi:nil="true" />
  <FirstName>Tess T.</FirstName>
  <LastName>Persson</LastName>
  <Company xsi:nil="true" />
  <Address1>Testgatan 42</Address1>
  <Address2 xsi:nil="true" />
  <ZIP>123 45</ZIP>
  <City>Testcity</City>
  <State xsi:nil="true" />
  <Country>SE</Country>
  <Language>sv</Language>
  <PhoneNo>012-345678</PhoneNo>
  <CellPhoneNo xsi:nil="true" />
  <FaxNo xsi:nil="true" />
  <EMail>customer@example.com</EMail>
  <DiscountPercent xsi:nil="true" />
  <CreditLimit>25000</CreditLimit>
  <CreditUsed>0</CreditUsed>
  <Currency xsi:nil="true" />
  <Login>customer@example.com</Login>
  <Comment xsi:nil="true" />
  <WantsNewsletter>false</WantsNewsletter>
  <HaveSentWelcomeEmail>false</HaveSentWelcomeEmail>
  <CustomerCategory href="/__API__/customercategory/2" xsi:nil="true" />
  <Pricelist href="/__API__/pricelist/1" xsi:nil="true" />
  <Warehouse href="/__API__/warehouse/3" xsi:nil="true" />
  <GroupDiscount>
    <Item>
      <DiscountGroup href="/__API__/discountgroup/1" xsi:nil="true" />
      <DiscountPercent>5</DiscountPercent>
    </Item>
    <Item>
      <DiscountGroup href="/__API__/discountgroup/2" xsi:nil="true" />
      <DiscountPercent>2</DiscountPercent>
    </Item>
  </GroupDiscount>
  <CustomerOrders href="/__API__/customer/1/orders" xsi:nil="true" />
  <Subscriptions href="/__API__/customer/1/subscriptions" xsi:nil="true"
     />
  <ExtraText1>t1</ExtraText1>
  <ExtraText2>t2</ExtraText2>
```

```
48  <ExtraText3 xsi:nil="true" />
49  <ExtraText4 xsi:nil="true" />
50  <ExtraText5 xsi:nil="true" />
51  <ExtraText6 xsi:nil="true" />
52  <ExtraSelector1 textvalue="item 2">2</ExtraSelector1>
53  <ExtraSelector2>0</ExtraSelector2>
54  <ExtraSelector3>0</ExtraSelector3>
55  <ExtraSelector4>0</ExtraSelector4>
56  <ExtraSelector5>0</ExtraSelector5>
57  <ExtraSelector6>0</ExtraSelector6>
58  <ExtraSelector7>0</ExtraSelector7>
59  <ExtraSelector8>0</ExtraSelector8>
60  <ExtraSelector9>0</ExtraSelector9>
61  <ExtraSelector10>0</ExtraSelector10>
62  <CreateLoginToken href="/__API__/customer/1/createlogintoken"
        xsi:nil="true" />
63  <ViaAffiliate href="/__API__/affiliateprogram/1/member/1"
        xsi:nil="true" />
64  <DeliveryAddresses href="/__API__/customer/1/deliveryaddress"
        xsi:nil="true" />
65  <PreferredDeliveryAddress href="/__API__/customer/1/deliveryaddress/1"
        xsi:nil="true" />
66  <CustomAttributes>
67    <CustomAttribute href="/__API__/customattribute/1">
68      <ID xsi:nil="true" />
69      <ValueID xsi:nil="true" />
70      <Name>Size</Name>
71      <Value href="/__API__/customattribute/1/value/3">L</Value>
72    </CustomAttribute>
73    <CustomAttribute href="/__API__/customattribute/2">
74      <ID xsi:nil="true" />
75      <ValueID xsi:nil="true" />
76      <Name>Color</Name>
77      <Value href="/__API__/customattribute/2/value/4">Cyan</Value>
78      <Value href="/__API__/customattribute/2/value/5">Magenta</Value>
79      <Value href="/__API__/customattribute/2/value/6">Yellow</Value>
80      <Value href="/__API__/customattribute/2/value/7">Black</Value>
81    </CustomAttribute>
82  </CustomAttributes>
83  <Keys>
84    <Key>
85      <Index>0</Index>
86      <Value>true</Value>
87    </Key>
88    <Key>
89      <Index>7</Index>
90      <Value>false</Value>
91    </Key>
92  </Keys>
93  <CreatedTime>2009-04-05T08:22:06Z</CreatedTime>
94  <ChangedTime>2016-08-30T09:33:16Z</ChangedTime>
```

```
95    <SyncedTime>2010-02-03T17:51:23Z</SyncedTime>
96  </Customer>
```

3.2.3 Customer Order Collection

URI:	`/__API__/customer/.../orders` (from CustomerOrders href attribute)
Actions:	Retrieve(GET)
Content-Type:	text/xml
GET Parameters:	view, filter

This resource supports the client-based synchronization described in section 2.11.1.

Query parameters

Name	Values	Description
`view`	*short,long*	If specified as "long" then an almost complete "Order" object is sent for each order. Fields left out of a long listing are among others the order lines.
`filter`		See the Order section 3.3.2. for details about which fields can be filtered.

Object description

The collection provides a list of abbreviated `Order` objects:

Name	Type	Mode	Description
`OrderNo`	*int*	*read*	The order number.
`ErpOrderNo`	*str*	*read*	The order number in the shop owners ERP system, if such a system is used. Is not used in the shop but may be used to link orders in the shop to orders in the ERP.
`State`	*str*	*read*	The order state as defined in the shop. If this tag is empty then the order has not yet been completed and therefore does not yet have a state. Order states are described below.
`PaymentState`	*str*	*read*	The payment state of this order. Usually "UNPAID", "AUTHORIZED" or "PAID". Payment states are described below.
`CreatedTime`	*datetime*	*read*	The date and time when this order was created.
`ChangedTime`	*datetime*	*read*	The date and time when this order record was last changed.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<OrderList>
  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/1">
    <OrderNo>1</OrderNo>
    <ErpOrderNo xsi:nil="true" />
    <State>NEW</State>
    <PaymentState>UNPAID</PaymentState>
    <PaymentIsCaptured>false</PaymentIsCaptured>
    <CaptureTime xsi:nil="true" />
    <PaymentIsCancelled>false</PaymentIsCancelled>
    <CancelTime xsi:nil="true" />
    <CreatedTime>2009-04-05T08:24:26Z</CreatedTime>
    <ChangedTime>2013-02-05T09:08:42Z</ChangedTime>
  </Order>
  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/2">
    <OrderNo>2</OrderNo>
    <ErpOrderNo xsi:nil="true" />
    <State>ACCEPTED</State>
    <PaymentState>UNPAID</PaymentState>
    <PaymentIsCaptured>false</PaymentIsCaptured>
    <CaptureTime xsi:nil="true" />
    <PaymentIsCancelled>false</PaymentIsCancelled>
    <CancelTime xsi:nil="true" />
    <CreatedTime>2009-04-15T19:51:39Z</CreatedTime>
    <ChangedTime>2013-02-05T09:08:42Z</ChangedTime>
  </Order>
</OrderList>
```

3.2.4 Customer subscription Collection

URI:	`/__API__/customer/.../subscriptions` (from Subscriptions href attribute)
Actions:	Retrieve(GET)
Content-Type:	text/xml
GET Parameters:	view, filter

This resource supports the client-based synchronization described in section 2.11.1.

Query parameters

Name	Values	Description
`view`	*short,long*	If specified as “long” then an almost complete “Subscription” object is sent for each order.
`filter`		See Subscriptions (section 3.22.2 for details about which fields can be filtered.

Object description

The collection provides a list of abbreviated `Subscription` objects:

Name	Type	Mode	Description
`State`	*str*	*read, write*	One of "ACTIVE", "RENEWAL_NOTICE_SENT", "INACTIVE", "CANCELLED"
`ExpireDate`	*date*	*read, write*	The date when this subscription expires if it isn't renewed.
`SubscriptionItems`	*empty*	*read*	A collection of all items within this subscription. The "href" attribute contains the collection URI. See below for details.
`CreatedTime`	*datetime*	*read*	The date and time when this record was created.
`ChangedTime`	*datetime*	*read*	The date and time when this record was last changed.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SubscriptionList>
  <Subscription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/subscription/1">
    <State>ACTIVE</State>
    <ExpireDate>2012-01-02</ExpireDate>
    <NextRenewalDate>2012-01-02</NextRenewalDate>
    <SubscriptionItems href="/__API__/subscription/1/item"
        xsi:nil="true" />
    <CreatedTime>2010-01-25T15:04:54Z</CreatedTime>
    <ChangedTime>2012-09-05T14:02:21Z</ChangedTime>
  </Subscription>
</SubscriptionList>
```

3.2.5 Delivery Addresses collection

URI:	`/__API__/customer/.../deliveryaddress` (from DeliveryAddresses href attribute)
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

The collection provides a list of `DeliveryAddress` objects:

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DeliveryAddressList>
  <DeliveryAddress xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customer/1/deliveryaddress/1">
    <Name>Testdeliv</Name>
```

```
    <ContactPerson>Jonny testington</ContactPerson>
    <Address1>Testgatan 41</Address1>
    <Address2 xsi:nil="true" />
    <State xsi:nil="true" />
    <ZIP>12345</ZIP>
    <City>Testburg</City>
    <Country>SE</Country>
    <PhoneNo>123</PhoneNo>
    <CellPhoneNo>321</CellPhoneNo>
    <EMail>test@example.com</EMail>
    <ShippingInstructions>instructions!</ShippingInstructions>
  </DeliveryAddress>
</DeliveryAddressList>
```

3.2.6 Delivery Address

URI:	`/__API__/customer/.../deliveryaddress` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
Attributes:	href

Object description

Name	Type	Mode	Description
`Name`	*str*	*read, write*	The name of the recipient. Can be company name or person.
`ContactPerson`	*str*	*read, write*	The name of the contact person if this delivery is to a company address.
`Address1`	*str*	*read, write*	First row of address.
`Address2`	*str*	*read, write*	Second row of address.
`State`	*str*	*read, write*	State or province if applicable.
`ZIP`	*str*	*read, write*	Postal code.
`City`	*str*	*read, write*	Postal code.
`Country`	*ccode*	*read, write*	ISO Country code.
`PhoneNo`	*str*	*read, write*	Telephone number.
`CellPhoneNo`	*str*	*read, write*	Cellphone number.
`EMail`	*str*	*read, write*	Email address.
`ShippingInstructions`	*str*	*read, write*	Any shipping instructions the customer has entered.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DeliveryAddress xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/customer/1/deliveryaddress/1">
  <Name>Testdeliv</Name>
  <ContactPerson>Jonny testington</ContactPerson>
```

```
5   <Address1>Testgatan 41</Address1>
6   <Address2 xsi:nil="true" />
7   <State xsi:nil="true" />
8   <ZIP>12345</ZIP>
9   <City>Testburg</City>
10  <Country>SE</Country>
11  <PhoneNo>123</PhoneNo>
12  <CellPhoneNo>321</CellPhoneNo>
13  <EMail>test@example.com</EMail>
14  <ShippingInstructions>instructions!</ShippingInstructions>
15 </DeliveryAddress>
```

3.2.7 Customer Auth Check

URI:	`/__API__/customer/authcheck`
Actions:	Create(POST)
Content-Type:	text/xml
200-OK:	Indicates that the login/password is correct, and returns minimal Customer object
400-BAD:	Wrong login/password.

This resource is used by an external entity to check a customer login and password and retrieve a minimal `Customer` object.

This is useful for external systems (forums, ticket-tracking, etc) that wish to use the same login and passwords for the customers as the shop.

The system POSTs the request object described below, and the server answers either with a 200-OK and a minimal customer object or with a 400-BAD REQUEST and an error message.

Object description for the request object

Name	Type	Mode	Description
`Login`	*str*	*write*	The login name for the customer
`Password`	*str*	*write*	The password

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <CustomerAuthCheck>
3   <Login>customer@example.com</Login>
4   <Password>test</Password>
5 </CustomerAuthCheck>
```

Object description for a successful response

This object is a partial `Customer` object as specified in section 3.2.2.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <CustomerNo>1</CustomerNo>
  <FirstName>Tess T.</FirstName>
  <LastName>Persson</LastName>
  <Company xsi:nil="true" />
  <Address1>Testgatan 42</Address1>
  <Address2 xsi:nil="true" />
  <ZIP>123 45</ZIP>
  <City>Testcity</City>
  <Country>SE</Country>
  <Language>sv</Language>
  <PhoneNo>012-345678</PhoneNo>
  <CellPhoneNo xsi:nil="true" />
  <FaxNo xsi:nil="true" />
  <EMail>customer@example.com</EMail>
</Customer>
```

Example XML for failed request.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Error>
  <Message>Unknown customer login or wrong password</Message>
</Error>
```

3.2.8 Automatic/Remote customer login

URI:	*Customer specific, see example xml below*
Actions:	Create(POST)
Content-Type:	text/xml

External systems might need to remotely log in a customer into the web shop. Customers might already be logged in into an external system. Or maybe a more elaborate customer login procedure is required. This API request provides a facility to accommodate such systems.

First the system requests a login token from the web shop by posting a LoginTokenRequest as described below. The response to the request contains an URI to which the customer is then redirected.

The shop will cause a customer using this special URI to be logged in into the shop, and redirected to the page specified in the request.

For security all tokens are time limited. The API provides a means in the request to specify the desired timeout. A timeout of 60-120 seconds is recommended. If no timeout is specified a default value is used.

Object description for "LoginTokenRequest"

Name	Type	Mode	Description
RedirectTo	*str*	*write*	Where the customer should be redirected after the token login is done. If not specified the webshop index page "/" is used.
ValidityInSeconds	*str*	*write*	How long (in seconds) the token is valid. If not specified a default value will be used.

Example XML for token request.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<LoginTokenRequest>
  <RedirectTo>/cart.html</RedirectTo>
  <ValidityInSeconds>60</ValidityInSeconds>
</LoginTokenRequest>
```

Object description for "LoginToken"

Name	Type	Mode	Description
URI	*str*	*read*	The URI to which the customer must be redirected to log in to the shop.
RedirectTo	*str*	*read*	Where the customer should be redirected after the token login is done.
ValidityInSeconds	*str*	*write*	How long (in seconds) the token is valid.

Example XML for response.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<LoginToken xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <URI>https://apitest.setest2.bk.svbint.com/?CUSTOMER_TOKENLOGIN=btbo5ZB4Mrjb7Tq52V
  <ValidityInSeconds>60</ValidityInSeconds>
  <RedirectTo>/cart.html</RedirectTo>
</LoginToken>
```

Extrafields

The Customer resource supports extrafields, both text fields and selector fields. Extrafields are configurable at /__API__/customer/extrafields. See section 3.25 for details.

3.3 Orders

These resources are used to retrieve and manipulate orders in the shop.

For most orders only some parameters may be changed. These are marked as *read, write* in the tables below. Some fields may be marked with *read(editable)*, these fields are read-only for non-editable orders.

For editable orders (see below) most fields can be changed. Editable orders are orders created through the API or by the shop owner that have not yet been closed. Orders created by customers are not editable.

The order resources support the synchronization procedure described in section 2.11.

Editing orders

The order editing api is experimental! If you use it in a client you are expected to keep yourself updated regarding changes in this API

Only some orders are editable. These orders are marked with “IsEditable” set to “true”. An order marked as “IsEditable” is not completed. When the order gets completed “IsEditable” will change to false. An order may be marked as completed by the shop owner or via the API by setting “IsEditable” to false.

Editable orders can be seen as consisting of two parts: The order itself (identified by the “href” attribute in the order object) and the order items (identified by the “href” attributes in the *OrderItems* and *Item* elements respectively. A client CAN NOT change order items by a PUT request to the order URI. Items can only be changed by requests to each item URI.

When editing an order the client performs a PUT request to the order URI as specified in the “href” attribute. The request SHOULD contain only those elements that the client wishes to change. The client MUST NOT send the *OrderItems* block. All items that are directly editable are marked as *write(editable)* or just *write*.

At this time the API DOES NOT recalculate the sum, weight or volume of the order. If a client changes the price, weight or volume of any item in the order the client must also update the order totals.

Creating orders

> ⚠ The order creation api is experimental! If you use it in a client you are expected to keep yourself updated regarding changes in this API

Order creation is done with a POST request to the order collection resource. Order creation works like order editing above with one additional feature: When creating orders, unlike editing, the client MAY send all order items embedded in the Order object in the same was as orders are retrieved from the API.

This allows the client to send one POST request to create a complete order by sending all the order-data in the *Order* object and setting "IsEditable" to false.

3.3.1 Order Collection

URI:	`/__API__/order`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, mark_as_synced, view, filter
POST Parameters:	handle_stock, recalc_ordersum, recalc_payship

Query parameters

Name	Values	Description
`synced`	*yes,no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`view`	*short,long*	If specified as "long" then a complete "Order" object is sent for each order.
`handle_stock`	*yes,no*	If set to yes stock processing (subtraction) will take place on order creation or state change.
`recalc_ordersum`	*yes,no*	If set to yes the sum and tax for the entire order will be calculated from the supplied prices and the TotalSum / TotalTax fields are ignored.
`recalc_payship`	*yes,no*	If set to yes payment and shipping price for the order is recalculated from shop settings. NOTE: This implies recalc_ordersum!

Additional filter parameters

These parameters can be used in addition to the object parameters marked as filterable.

Name	Type	Description
ShipmentChangedTime	*datetime*	Match the changed time of any shipment for this order.

Object description

The collection provides a list of abbreviated `Order` objects:

Name	Type	Mode	Description
OrderNo	*str*	*read*	The order number.
ErpOrderNo	*str*	*read*	The order number in the shop owners ERP system, if such a system is used. Is not used in the shop but may be used to link orders in the shop to orders in the ERP.
Customer	*str*	*read*	A reference to the customer that has made this order. The "href" attribute references the customer object itself. The element contains the customer number (CustomerNo) for the customer as an extra aid.
State	*str*	*read*	The order state as defined in the shop. If this tag is empty then the order has not yet been completed and therefore does not yet have a state. Order states are described below.
PaymentIsCaptured	*bool*	*read*	True if the payment for this order has been "captured". This is only valid for payment method supporting 2-stage payments.
PaymentIsCancelled	*bool*	*read*	True if the payment for this order has been cancelled.
PaymentState	*str*	*read*	The payment state of this order. Usually "UNPAID", "AUTHORIZED" or "PAID". Payment states are described below.
CreatedTime	*datetime*	*read*	The date and time when this order was created.
ChangedTime	*datetime*	*read*	The date and time when this order record was last changed.
SyncedTime	*datetime*	*read*	The date and time when this order record was last synchronized.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OrderList>
  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/1" order_id="1">
    <OrderNo>1</OrderNo>
    <ErpOrderNo xsi:nil="true" />
    <Customer href="/__API__/customer/1">1</Customer>
    <State>NEW</State>
    <PaymentState>UNPAID</PaymentState>
```

```
9     <PaymentIsCaptured>false</PaymentIsCaptured>
10    <CaptureTime xsi:nil="true" />
11    <PaymentIsCancelled>false</PaymentIsCancelled>
12    <CancelTime xsi:nil="true" />
13    <CreatedTime>2009-04-05T08:24:26Z</CreatedTime>
14    <ChangedTime>2013-02-05T09:08:42Z</ChangedTime>
15    <SyncedTime xsi:nil="true" />
16  </Order>
17  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/order/2" order_id="2">
18    <OrderNo>2</OrderNo>
19    <ErpOrderNo xsi:nil="true" />
20    <Customer href="/__API__/customer/1">1</Customer>
21    <State>ACCEPTED</State>
22    <PaymentState>UNPAID</PaymentState>
23    <PaymentIsCaptured>false</PaymentIsCaptured>
24    <CaptureTime xsi:nil="true" />
25    <PaymentIsCancelled>false</PaymentIsCancelled>
26    <CancelTime xsi:nil="true" />
27    <CreatedTime>2009-04-15T19:51:39Z</CreatedTime>
28    <ChangedTime>2013-02-05T09:08:42Z</ChangedTime>
29    <SyncedTime xsi:nil="true" />
30  </Order>
31  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/order/3" order_id="3">
32    <OrderNo>999910000</OrderNo>
33    <ErpOrderNo xsi:nil="true" />
34    <Customer xsi:nil="true" />
35    <State xsi:nil="true" />
36    <PaymentState>UNPAID</PaymentState>
37    <PaymentIsCaptured>false</PaymentIsCaptured>
38    <CaptureTime xsi:nil="true" />
39    <PaymentIsCancelled>false</PaymentIsCancelled>
40    <CancelTime xsi:nil="true" />
41    <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
42    <ChangedTime>2016-01-27T10:35:40Z</ChangedTime>
43    <SyncedTime xsi:nil="true" />
44  </Order>
45  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/order/4" order_id="4">
46    <OrderNo>999910000</OrderNo>
47    <ErpOrderNo xsi:nil="true" />
48    <Customer xsi:nil="true" />
49    <State xsi:nil="true" />
50    <PaymentState>UNPAID</PaymentState>
51    <PaymentIsCaptured>false</PaymentIsCaptured>
52    <CaptureTime xsi:nil="true" />
53    <PaymentIsCancelled>false</PaymentIsCancelled>
54    <CancelTime xsi:nil="true" />
55    <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
56    <ChangedTime>2016-02-01T10:21:58Z</ChangedTime>
```

```
57      <SyncedTime xsi:nil="true" />
58    </Order>
59    <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/5" order_id="5">
60      <OrderNo>999910000</OrderNo>
61      <ErpOrderNo xsi:nil="true" />
62      <Customer xsi:nil="true" />
63      <State xsi:nil="true" />
64      <PaymentState>UNPAID</PaymentState>
65      <PaymentIsCaptured>false</PaymentIsCaptured>
66      <CaptureTime xsi:nil="true" />
67      <PaymentIsCancelled>false</PaymentIsCancelled>
68      <CancelTime xsi:nil="true" />
69      <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
70      <ChangedTime>2016-02-01T10:27:46Z</ChangedTime>
71      <SyncedTime xsi:nil="true" />
72    </Order>
73    <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/6" order_id="6">
74      <OrderNo>999910000</OrderNo>
75      <ErpOrderNo xsi:nil="true" />
76      <Customer xsi:nil="true" />
77      <State xsi:nil="true" />
78      <PaymentState>UNPAID</PaymentState>
79      <PaymentIsCaptured>false</PaymentIsCaptured>
80      <CaptureTime xsi:nil="true" />
81      <PaymentIsCancelled>false</PaymentIsCancelled>
82      <CancelTime xsi:nil="true" />
83      <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
84      <ChangedTime>2016-02-01T10:29:39Z</ChangedTime>
85      <SyncedTime xsi:nil="true" />
86    </Order>
87    <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/7" order_id="7">
88      <OrderNo>999910000</OrderNo>
89      <ErpOrderNo xsi:nil="true" />
90      <Customer xsi:nil="true" />
91      <State xsi:nil="true" />
92      <PaymentState>UNPAID</PaymentState>
93      <PaymentIsCaptured>false</PaymentIsCaptured>
94      <CaptureTime xsi:nil="true" />
95      <PaymentIsCancelled>false</PaymentIsCancelled>
96      <CancelTime xsi:nil="true" />
97      <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
98      <ChangedTime>2016-02-01T10:31:16Z</ChangedTime>
99      <SyncedTime xsi:nil="true" />
100   </Order>
101   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/8" order_id="8">
102     <OrderNo>999910000</OrderNo>
103     <ErpOrderNo xsi:nil="true" />
```

```
104     <Customer xsi:nil="true" />
105     <State xsi:nil="true" />
106     <PaymentState>UNPAID</PaymentState>
107     <PaymentIsCaptured>false</PaymentIsCaptured>
108     <CaptureTime xsi:nil="true" />
109     <PaymentIsCancelled>false</PaymentIsCancelled>
110     <CancelTime xsi:nil="true" />
111     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
112     <ChangedTime>2016-02-01T10:35:02Z</ChangedTime>
113     <SyncedTime xsi:nil="true" />
114   </Order>
115   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/9" order_id="9">
116     <OrderNo>999910000</OrderNo>
117     <ErpOrderNo xsi:nil="true" />
118     <Customer xsi:nil="true" />
119     <State xsi:nil="true" />
120     <PaymentState>UNPAID</PaymentState>
121     <PaymentIsCaptured>false</PaymentIsCaptured>
122     <CaptureTime xsi:nil="true" />
123     <PaymentIsCancelled>false</PaymentIsCancelled>
124     <CancelTime xsi:nil="true" />
125     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
126     <ChangedTime>2016-02-01T10:36:16Z</ChangedTime>
127     <SyncedTime xsi:nil="true" />
128   </Order>
129   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/10" order_id="10">
130     <OrderNo>999910000</OrderNo>
131     <ErpOrderNo xsi:nil="true" />
132     <Customer xsi:nil="true" />
133     <State xsi:nil="true" />
134     <PaymentState>UNPAID</PaymentState>
135     <PaymentIsCaptured>false</PaymentIsCaptured>
136     <CaptureTime xsi:nil="true" />
137     <PaymentIsCancelled>false</PaymentIsCancelled>
138     <CancelTime xsi:nil="true" />
139     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
140     <ChangedTime>2016-02-01T10:37:19Z</ChangedTime>
141     <SyncedTime xsi:nil="true" />
142   </Order>
143   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/11" order_id="11">
144     <OrderNo>999910000</OrderNo>
145     <ErpOrderNo xsi:nil="true" />
146     <Customer xsi:nil="true" />
147     <State xsi:nil="true" />
148     <PaymentState>UNPAID</PaymentState>
149     <PaymentIsCaptured>false</PaymentIsCaptured>
150     <CaptureTime xsi:nil="true" />
151     <PaymentIsCancelled>false</PaymentIsCancelled>
```

```
152     <CancelTime xsi:nil="true" />
153     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
154     <ChangedTime>2016-02-01T10:38:50Z</ChangedTime>
155     <SyncedTime xsi:nil="true" />
156   </Order>
157   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/12" order_id="12">
158     <OrderNo>999910000</OrderNo>
159     <ErpOrderNo xsi:nil="true" />
160     <Customer xsi:nil="true" />
161     <State xsi:nil="true" />
162     <PaymentState>UNPAID</PaymentState>
163     <PaymentIsCaptured>false</PaymentIsCaptured>
164     <CaptureTime xsi:nil="true" />
165     <PaymentIsCancelled>false</PaymentIsCancelled>
166     <CancelTime xsi:nil="true" />
167     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
168     <ChangedTime>2016-02-01T10:39:23Z</ChangedTime>
169     <SyncedTime xsi:nil="true" />
170   </Order>
171   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/13" order_id="13">
172     <OrderNo>999910000</OrderNo>
173     <ErpOrderNo xsi:nil="true" />
174     <Customer xsi:nil="true" />
175     <State xsi:nil="true" />
176     <PaymentState>UNPAID</PaymentState>
177     <PaymentIsCaptured>false</PaymentIsCaptured>
178     <CaptureTime xsi:nil="true" />
179     <PaymentIsCancelled>false</PaymentIsCancelled>
180     <CancelTime xsi:nil="true" />
181     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
182     <ChangedTime>2016-02-01T10:44:13Z</ChangedTime>
183     <SyncedTime xsi:nil="true" />
184   </Order>
185   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/14" order_id="14">
186     <OrderNo>999910000</OrderNo>
187     <ErpOrderNo xsi:nil="true" />
188     <Customer xsi:nil="true" />
189     <State xsi:nil="true" />
190     <PaymentState>UNPAID</PaymentState>
191     <PaymentIsCaptured>false</PaymentIsCaptured>
192     <CaptureTime xsi:nil="true" />
193     <PaymentIsCancelled>false</PaymentIsCancelled>
194     <CancelTime xsi:nil="true" />
195     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
196     <ChangedTime>2016-02-01T11:00:17Z</ChangedTime>
197     <SyncedTime xsi:nil="true" />
198   </Order>
```

```
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/order/15" order_id="15">
  <OrderNo>999910000</OrderNo>
  <ErpOrderNo xsi:nil="true" />
  <Customer xsi:nil="true" />
  <State xsi:nil="true" />
  <PaymentState>UNPAID</PaymentState>
  <PaymentIsCaptured>false</PaymentIsCaptured>
  <CaptureTime xsi:nil="true" />
  <PaymentIsCancelled>false</PaymentIsCancelled>
  <CancelTime xsi:nil="true" />
  <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
  <ChangedTime>2016-02-19T07:33:37Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</Order>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/order/16" order_id="16">
  <OrderNo>999910000</OrderNo>
  <ErpOrderNo xsi:nil="true" />
  <Customer xsi:nil="true" />
  <State xsi:nil="true" />
  <PaymentState>UNPAID</PaymentState>
  <PaymentIsCaptured>false</PaymentIsCaptured>
  <CaptureTime xsi:nil="true" />
  <PaymentIsCancelled>false</PaymentIsCancelled>
  <CancelTime xsi:nil="true" />
  <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
  <ChangedTime>2016-02-19T08:16:26Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</Order>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/order/17" order_id="17">
  <OrderNo>999910000</OrderNo>
  <ErpOrderNo xsi:nil="true" />
  <Customer xsi:nil="true" />
  <State xsi:nil="true" />
  <PaymentState>UNPAID</PaymentState>
  <PaymentIsCaptured>false</PaymentIsCaptured>
  <CaptureTime xsi:nil="true" />
  <PaymentIsCancelled>false</PaymentIsCancelled>
  <CancelTime xsi:nil="true" />
  <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
  <ChangedTime>2016-02-19T08:16:52Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</Order>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/order/18" order_id="18">
  <OrderNo>999910000</OrderNo>
  <ErpOrderNo xsi:nil="true" />
  <Customer xsi:nil="true" />
  <State xsi:nil="true" />
```

```
246     <PaymentState>UNPAID</PaymentState>
247     <PaymentIsCaptured>false</PaymentIsCaptured>
248     <CaptureTime xsi:nil="true" />
249     <PaymentIsCancelled>false</PaymentIsCancelled>
250     <CancelTime xsi:nil="true" />
251     <CreatedTime>2015-12-29T12:58:53Z</CreatedTime>
252     <ChangedTime>2016-02-19T08:18:03Z</ChangedTime>
253     <SyncedTime xsi:nil="true" />
254   </Order>
255 </OrderList>
```

3.3.2 Order

URI:	`/__API__/order/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml
GET Parameters:	mark_as_synced
PUT Parameters:	mark_as_synced, payment_action, handle_stock, recalc_ordersum, recalc_payship
Attributes:	href, order_id

Query parameters

Name	Values	Description
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`payment_action`	*capture,cancel*	This parameter asks the system to either "capture" or cancel the payment for this order. This is explained in section 3.3.4
`handle_stock`	*yes,no*	If set to yes stock processing (subtraction) will take place on order creation or state change.
`recalc_ordersum`	*yes,no*	If set to yes the sum and tax for the entire order will be calculated from the supplied prices and the TotalSum / TotalTax fields are ignored.
`recalc_payship`	*yes,no*	If set to yes payment and shipping price for the order is recalculated from shop settings. NOTE: This implies recalc_ordersum!

Object description

The `Order` element contains a "href" attribute specifying the complete URI for this order. It also contains a "order_id" attribute that is the order_id parameter used in the shop front end. This might be used by the API user to create links that link into a specific order in the shop.

Name	Type	Mode	Description
OrderNo	*str*	*read, write(editable), filter*	The order number.
ErpOrderNo	*str*	*read, write, filter*	The order number in the shop owners ERP system, if such a system is used. Is not used in the shop but may be used to link orders in the shop to orders in the ERP.
Customer	*str*	*read, write(editable), filter*	A reference to the customer that has made this order. The "href" attribute references the customer object itself.
State	*str*	*read, write, filter*	The order state as defined in the shop. If this tag is empty then the order has not yet been completed and therefore does not yet have a state. Order states are described in section 3.3.5
IsEditable	*bool*	*read, write(editable), filter*	Marks if this order is in an editable state. Editable orders are not completed yet and MAY be edited by the shop owner. Orders placed by customers are usually NOT editable.
ExportType	*str*	*read, write(editable)*	Specifies if this was an export or not, and what type of export. Allowed values are "NOEXPORT" — This was not an export, "TOEUWITHVATNO" export within EU and both shop and customer have a VAT number, "TOEUONESTOP" export within EU but customer (or shop) didn't have a VAT number but the shop have the opt for OSS (See 3.1) enabled, "TOEUNOVATNO" export within EU but customer (or shop) didn't have a VAT number, "OUTSIDEOFEUVOEC" export was to Norway and the opt for Voec (See 3.1) was enabled in the shop, "OUTSIDEOFEU" — export was to country outside of EU.
VatNo	*str*	*read, write(editable)*	VAT number customer had when order was placed. This field SHOULD only be used if "ExportType" above indicates that a VAT number was needed. This field MAY contain a VAT-number even if it was not required.
Pricelist	*str*	*read, write*	The price list used for this order. The "href" attribute specifies the price list. The element value is the pricelist ID, sent only as an aid to API users. It is ignored by the shop.
PaymentState	*str*	*read, write, filter*	The payment state of this order. Usually "UNPAID", "AUTHORIZED" or "PAID". Payment states are described below.
PaymentRefShop	*str*	*read, write(editable), filter*	A reference for the payment. Usually an ID identifying the payment in the payment processor.

contd...

Name	Type	Mode	Description
PaymentRefCustomer	*str*	*read, write(editable), filter*	A customer reference for the payment. Usually a transaction ID at the payment processor.
PaymentRefCapture	*str*	*read*	The reference for the payment that is used to capture the reserved amount.
PaymentTime	*datetime*	*read, write(editable), filter*	The time for the payment, if available.
PaymentIsCaptured	*bool*	*read, write(editable)*	True if the payment for this order has been "captured". This is only valid for payment method supporting 2-stage payments.
CaptureTime	*datetime*	*read, write(editable), filter*	Time when payment for this order was captured.
PaymentIsCancelled	*bool*	*read, write(editable)*	True if the payment for this order has been cancelled.
CancelTime	*datetime*	*read, write(editable), filter*	Time when payment for this order was cancelled.
OrderSource	*str*	*read, write(editable), filter*	Currently only "WEBSHOP".
TotalSum	*float*	*read, write(editable), filter*	Total order sum, including tax.
TotalTax	*float*	*read, write(editable)*	Total order tax.
PaymentType	*str*	*read, write(editable)*	The "tag" value of the payment method selected for this order. The *href* attribute contains the *PaymentMethod* URI.
PaymentPrice	*float*	*read, write(editable)*	The price for payment of this order.
PaymentTax	*float*	*read, write(editable)*	The tax for payment of this order.
PaymentTaxPercent	*float*	*read, write(editable)*	The tax in percent for the payment fee. If PaymentPrice and PaymentTax are both sent when updating an order, but PaymentTaxPercent IS NOT sent the system will estimate the tax.
ShippingType	*str*	*read, write(editable)*	The "tag" value of the shipping method selected for this order.The *href* attribute contains the *ShippingMethod* URI.
ShippingPrice	*float*	*read, write(editable)*	The price for shipment of this order.
ShippingTax	*float*	*read, write(editable)*	The tax for shipment of this order.
ShippingTaxPercent	*float*	*read, write(editable)*	The tax in percent for the shipping fee. If ShippingPrice and ShippingTax are both sent when updating an order, but ShippingTaxPercent IS NOT sent the system will estimate the tax.

contd...

Name	Type	Mode	Description
Currency	*str*	*read, write(editable), filter*	The currency for this order. The default currency is used if this field is empty.
CurrencyConversionRate	*float*	*read, write(editable)*	The rate used by the shop when converting this order to the base currency.(See 3.3.2)
CustomerIP	*str*	*read, write(editable), filter*	The IP address of the customer that performed this order.
MessageFromCustomer	*str*	*read, write(editable)*	A message from the customer.
MessageToCustomer	*str*	*read, write*	A message to the customer.
CustomerReferenceNumber	*str*	*read, write(editable)*	A reference number/ID for this order if entered by the customer.
CustomerOrgNo	*str*	*read, write(editable)*	The customer org. number.
CustomerName	*str*	*read, write(editable)*	The customer name.
CustomerFirstName	*str*	*read, write(editable)*	The customer first name if known.
CustomerLastName	*str*	*read, write(editable)*	The customer last name if known.
CustomerCompany	*str*	*read, write(editable)*	The customer company.
CustomerAddress1	*str*	*read, write(editable)*	The customer address, line 1.
CustomerAddress2	*str*	*read, write(editable)*	The customer address, line 2.
CustomerPostalCode	*str*	*read, write(editable)*	The customer postal/zip code.
CustomerCity	*str*	*read, write(editable)*	The customer city.
CustomerCountry	*ccode*	*read, write(editable)*	The customer country code.
Language	*lang*	*read, write, filter*	2-letter language code according to ISO-639-1.
ShippingName	*str*	*read, write(editable)*	The name on the shipping address. Usually a company name.
ShippingFirstName	*str*	*read, write(editable)*	The first name of the shipping contact if known.
ShippingLastName	*str*	*read, write(editable)*	The last name of the shipping contact if known.
ShippingContact	*str*	*read, write(editable)*	The name of the person for whom this shipment is destined.
ShippingAddress1	*str*	*read, write(editable)*	The shipping address, line 1.
ShippingAddress2	*str*	*read, write(editable)*	The shipping address, line 2.
ShippingPostalCode	*str*	*read, write(editable)*	The shipping postal code.

contd...

Name	Type	Mode	Description
ShippingCity	*str*	*read, write(editable)*	The shipping city.
ShippingCountry	*ccode*	*read, write(editable)*	The shipping country code.
ShippingPhone	*str*	*read, write(editable)*	The phone number of the contact person.
ShippingCellPhone	*str*	*read, write(editable)*	The cell phone / sms advice number of the recipient.
ShippingEMail	*emailaddr*	*read, write(editable)*	The email address of the contact person.
ShippingInstructions	*str*	*read, write(editable)*	Any additional shipping instructions.
RequestedDeliveryDate	*date*	*read, write(editable), filter*	The date the customer entered as requested delivery date.
PartialShipmentOK	*bool*	*read, write(editable)*	True if customer is willing to accept partial shipments of this order. False if customer requests that all items be shipped together.
ExtraText1..6	*str*	*read, write(editable)*	Extra text parameters.
ExtraSelect1..4	*int*	*read, write(editable)*	Extra select fields, defined in the shop. Can only take on values between 0 and 256, unlike most integers. If set the attribute *textvalue* will contain a textual value corresponding to the numeric index.
OrderItems	*array*	*read, write(createonly)*	An array of Item objects representing the order lines (items purchased). This object is described below. The "href" attribute references a collection of exactly these items. For editing / order creation purposes the items MUST be accessed using the collection.
Shipment	*empty*	*read*	The "href" attribute references the collection of shipments for this order. (See 3.3.3)
Weight	*float*	*read, write(editable)*	The total weight for this order
Volume	*float*	*read, write(editable)*	The total volume for this order
IsBulky	*bool*	*read, write(editable)*	An informational field that is true if this order contains any items marked as bulky
CreatedTime	*datetime*	*read, write(editable), filter*	The date and time when this order was created.
ChangedTime	*datetime*	*read, filter*	The date and time when this order record was last changed.
SyncedTime	*datetime*	*read, filter*	The date and time when this order record was last synchronized.

Payment states

The payment state is used to indicate payment progress of the order. The state transitions are mostly defined by the payment method in use. Simple methods may only have "UNPAID" / "PAID" while more advanced may have some intermediate states, some invoicing methods have the following state transition: "AUTHORIZED" -> "INVOICED" -> "PAID".

Please note that additional payment states MAY appear. Applications SHOULD ignore unknown payment states.

UNPAID The order has not yet been paid.
PAID The order is paid in full.
AUTHORIZED The payment processor has authorized payment for this order. Usually means the payment processor has accepted the customer and the payment information (credit card number for example).
INVOICED An invoice has been sent for this order.
DENIED The payment processor denied payment for this order.
CREDIT_OK Credit has been granted for this order.

Currency Conversion Rates

The CurrencyConversionRate is used by the shop when converting this order to the base currency. If you use pricelists for currency conversion in the shop, or your order is in the base currency, you can not use this value. If it is left empty and the currency on the order is NOT the base currency, the system will try to set a conversion rate based on the currency settings in the shop.

Object description for "OrderItems"

Name	Type	Mode	Description
`LineNumber`	*int*	*read, write(editable)*	The line number for this item.
`SKU`	*str*	*read, write(editable)*	Stock-Keeping Unit / article number.
`ManufacturerSKU`	*str*	*read, write(editable)*	The manufacturers Stock-keeping unit, if known at time of purchase.
`Variant`	*empty*	*read, write(editable)*	The "href" attribute contains the Variant URI for this item. This field is NOT required and may be blank or omitted if the item isn't based on a current Product Variant.
`Qty`	*float*	*read, write(editable)*	Quantity purchased. Note this is a float, not an integer.
`Price`	*float*	*read, write(editable)*	The per-unit price for this item.

contd...

Name	Type	Mode	Description
Tax	*float*	*read, write(editable)*	The per-unit tax for this item.
TaxPercent	*float*	*read, write(editable)*	The per-unit tax in percent for this item. If Price and Tax are sent but TaxPercent IS NOT sent when updating an order the system will estimate the tax percentage.
Discount	*float*	*read, write(editable)*	Discount in percent that was granted on this purchase.
Comment	*str*	*read, write(editable)*	Comment for this line. Mostly useful for comment-lines.
Type	*str*	*read, write(editable)*	NORMAL or COMMENT.
ShipmentStatus	*str*	*read, write(editable)*	The shipment status for this item if defined. Currently not used in the shop.
ProductName	*str*	*read, write(editable)*	The product name.
VariantName	*str*	*read, write(editable)*	The variant name.
Size	*str*	*read, write(editable)*	The "size" of the item purchased by the customer.
Qualifier1..2	*str*	*read, write(editable)*	Optional information sometimes used for special functionality.
ExtraText1..6	*str*	*read, write(editable)*	Extra text parameters.
Weight	*float*	*read, write(editable)*	The total weight for this row
Volume	*float*	*read, write(editable)*	The total volume for this row
Unit	*str*	*read, write(editable)*	The unit for this row
IsBulky	*bool*	*read, write(editable)*	An informational field that is true if this order line contains a bulky item
InputFields	*str*	*read*	List of input fields assigned to this order item. The fields are displayed one per line and with title and value separated with a comma sign (:). This field is currently not enabled for updates in edit mode.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/order/1" order_id="1">
  <OrderNo>1</OrderNo>
  <ErpOrderNo xsi:nil="true" />
  <Customer href="/__API__/customer/1">1</Customer>
  <State>NEW</State>
  <IsEditable>false</IsEditable>
  <PaymentState>UNPAID</PaymentState>
  <PaymentIsCaptured>false</PaymentIsCaptured>
  <CaptureTime xsi:nil="true" />
```

```
<PaymentIsCancelled>false</PaymentIsCancelled>
<CancelTime xsi:nil="true" />
<ExportType>NOEXPORT</ExportType>
<VatNo xsi:nil="true" />
<Pricelist href="/__API__/pricelist/1">PRIS1</Pricelist>
<PaymentRefShop xsi:nil="true" />
<PaymentRefCustomer xsi:nil="true" />
<PaymentRefCapture xsi:nil="true" />
<PaymentTime xsi:nil="true" />
<OrderSource>WEBSHOP</OrderSource>
<TotalSum>1152.5</TotalSum>
<TotalTax>230.5</TotalTax>
<PaymentType href="/__API__/paymentmethod/111">PF-FPAK</PaymentType>
<PaymentPrice>82</PaymentPrice>
<PaymentTax>20.5</PaymentTax>
<PaymentTaxPercent>25</PaymentTaxPercent>
<ShippingType href="/__API__/shippingmethod/117">F-PAK</ShippingType>
<ShippingPrice>96</ShippingPrice>
<ShippingTax>24</ShippingTax>
<ShippingTaxPercent>25</ShippingTaxPercent>
<Currency xsi:nil="true" />
<CustomerIP>fe80::219:d1ff:fe6f:cd3f</CustomerIP>
<Comments></Comments>
<MessageFromCustomer xsi:nil="true" />
<MessageToCustomer xsi:nil="true" />
<CustomerReferenceNumber xsi:nil="true" />
<CustomerOrgNo xsi:nil="true" />
<CustomerName>Tess T. Persson</CustomerName>
<CustomerFirstName xsi:nil="true" />
<CustomerLastName xsi:nil="true" />
<CustomerCompany xsi:nil="true" />
<CustomerAddress1>Testgatan 42</CustomerAddress1>
<CustomerAddress2 xsi:nil="true" />
<CustomerPostalCode>123 45</CustomerPostalCode>
<CustomerCity>Testcity</CustomerCity>
<CustomerCountry>SE</CustomerCountry>
<Language>sv</Language>
<ShippingName xsi:nil="true" />
<ShippingFirstName xsi:nil="true" />
<ShippingLastName xsi:nil="true" />
<ShippingContact>Tess T. Persson</ShippingContact>
<ShippingAddress1>Testgatan 42</ShippingAddress1>
<ShippingAddress2 xsi:nil="true" />
<ShippingPostalCode>123 45</ShippingPostalCode>
<ShippingCity>Testcity</ShippingCity>
<ShippingCountry>SE</ShippingCountry>
<ShippingPhone>012-345678</ShippingPhone>
<ShippingCellPhone xsi:nil="true" />
<ShippingEMail>customer@example.com</ShippingEMail>
<ShippingInstructions xsi:nil="true" />
<RequestedDeliveryDate>2011-03-30</RequestedDeliveryDate>
```

```
<PartialShipmentOK>true</PartialShipmentOK>
<ExtraText1 xsi:nil="true" />
<ExtraText2 xsi:nil="true" />
<ExtraText3 xsi:nil="true" />
<ExtraText4 xsi:nil="true" />
<ExtraText5 xsi:nil="true" />
<ExtraText6 xsi:nil="true" />
<ExtraSelect1>0</ExtraSelect1>
<ExtraSelect2>0</ExtraSelect2>
<ExtraSelect3>0</ExtraSelect3>
<ExtraSelect4>0</ExtraSelect4>
<Weight xsi:nil="true" />
<Volume xsi:nil="true" />
<IsBulky>false</IsBulky>
<OrderItems href="/__API__/order/1/item">
  <Item href="/__API__/order/1/item/1">
    <LineNumber>1</LineNumber>
    <Variant href="/__API__/product/8/variant/12" xsi:nil="true" />
    <SKU>1</SKU>
    <ManufacturerSKU xsi:nil="true" />
    <Qty>1</Qty>
    <Price>360</Price>
    <Tax>90</Tax>
    <TaxPercent>400</TaxPercent>
    <Discount>0</Discount>
    <Comment xsi:nil="true" />
    <Type>NORMAL</Type>
    <ShipmentStatus></ShipmentStatus>
    <ProductName>Produkt nummer ett</ProductName>
    <VariantName>Produkt nummer ett</VariantName>
    <Size></Size>
    <Qualifier1 xsi:nil="true" />
    <Qualifier2 xsi:nil="true" />
    <ExtraText1 xsi:nil="true" />
    <ExtraText2 xsi:nil="true" />
    <ExtraText3 xsi:nil="true" />
    <ExtraText4 xsi:nil="true" />
    <ExtraText5 xsi:nil="true" />
    <ExtraText6 xsi:nil="true" />
    <Weight xsi:nil="true" />
    <Volume xsi:nil="true" />
    <Unit>st</Unit>
    <IsBulky>false</IsBulky>
  </Item>
  <Item href="/__API__/order/1/item/2">
    <LineNumber>2</LineNumber>
    <Variant href="/__API__/product/10/variant/14" xsi:nil="true" />
    <SKU>3</SKU>
    <ManufacturerSKU xsi:nil="true" />
    <Qty>2</Qty>
    <Price>192</Price>
```

```
113       <Tax>48</Tax>
114       <TaxPercent>400</TaxPercent>
115       <Discount>0</Discount>
116       <Comment xsi:nil="true" />
117       <Type>NORMAL</Type>
118       <ShipmentStatus></ShipmentStatus>
119       <ProductName>Produkt nummer tre</ProductName>
120       <VariantName>Produkt nummer tre</VariantName>
121       <Size></Size>
122       <Qualifier1 xsi:nil="true" />
123       <Qualifier2 xsi:nil="true" />
124       <ExtraText1 xsi:nil="true" />
125       <ExtraText2 xsi:nil="true" />
126       <ExtraText3 xsi:nil="true" />
127       <ExtraText4 xsi:nil="true" />
128       <ExtraText5 xsi:nil="true" />
129       <ExtraText6 xsi:nil="true" />
130       <Weight xsi:nil="true" />
131       <Volume xsi:nil="true" />
132       <Unit>st</Unit>
133       <IsBulky>false</IsBulky>
134     </Item>
135   </OrderItems>
136   <Shipments href="/__API__/order/1/shipment" xsi:nil="true" />
137   <CreatedTime>2009-04-05T08:24:26Z</CreatedTime>
138   <ChangedTime>2013-02-05T09:08:42Z</ChangedTime>
139   <SyncedTime xsi:nil="true" />
140 </Order>
```

Order item collection

URI:	*Specified per order in "OrderItems" element*
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

The collection contains a list of OrderItems in the format specified above. New items MAY ONLY be created for editable orders. Make sure that you update the order totals as well when adding items!

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <ItemList>
3   <Item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/order/2/item/3">
4     <LineNumber>1</LineNumber>
5     <Variant href="/__API__/product/10/variant/14" xsi:nil="true" />
6     <SKU>3</SKU>
7     <ManufacturerSKU xsi:nil="true" />
```

```
8       <Qty>2</Qty>
9       <Price>192</Price>
10      <Tax>48</Tax>
11      <TaxPercent>400</TaxPercent>
12      <Discount>0</Discount>
13      <Comment xsi:nil="true" />
14      <Type>NORMAL</Type>
15      <ShipmentStatus></ShipmentStatus>
16      <ProductName>Produkt nummer tre</ProductName>
17      <VariantName>Produkt nummer tre</VariantName>
18      <Size></Size>
19      <Qualifier1 xsi:nil="true" />
20      <Qualifier2 xsi:nil="true" />
21      <ExtraText1 xsi:nil="true" />
22      <ExtraText2 xsi:nil="true" />
23      <ExtraText3 xsi:nil="true" />
24      <ExtraText4 xsi:nil="true" />
25      <ExtraText5 xsi:nil="true" />
26      <ExtraText6 xsi:nil="true" />
27      <Weight xsi:nil="true" />
28      <Volume xsi:nil="true" />
29      <Unit>st</Unit>
30      <IsBulky>false</IsBulky>
31      <InputFields>Height:197cm<br />Shoe size:48</InputFields>
32    </Item>
33    <Item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/2/item/4">
34      <LineNumber>2</LineNumber>
35      <Variant href="/__API__/product/17/variant/30" xsi:nil="true" />
36      <SKU>apiuser</SKU>
37      <ManufacturerSKU xsi:nil="true" />
38      <Qty>1</Qty>
39      <Price>-38.4</Price>
40      <Tax>-9.6</Tax>
41      <TaxPercent>400</TaxPercent>
42      <Discount>0</Discount>
43      <Comment xsi:nil="true" />
44      <Type>NORMAL</Type>
45      <ShipmentStatus></ShipmentStatus>
46      <ProductName>API tester campaign.</ProductName>
47      <VariantName>API tester campaign.</VariantName>
48      <Size></Size>
49      <Qualifier1 xsi:nil="true" />
50      <Qualifier2 xsi:nil="true" />
51      <ExtraText1 xsi:nil="true" />
52      <ExtraText2 xsi:nil="true" />
53      <ExtraText3 xsi:nil="true" />
54      <ExtraText4 xsi:nil="true" />
55      <ExtraText5 xsi:nil="true" />
56      <ExtraText6 xsi:nil="true" />
57      <Weight xsi:nil="true" />
```

```
    <Volume xsi:nil="true" />
    <Unit xsi:nil="true" />
    <IsBulky>false</IsBulky>
  </Item>
</ItemList>
```

Order item

URI:	*Specified per order in "Item" element*
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

The collection contains a list of Order items in the format specified above. An item MAY ONLY be updated or deleted for editable orders. Make sure that you update the order totals as well when changing items!

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <Item xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/2/item/4">
    <LineNumber>2</LineNumber>
    <Variant href="/__API__/product/17/variant/30" xsi:nil="true" />
    <SKU>apiuser</SKU>
    <ManufacturerSKU xsi:nil="true" />
    <Qty>1</Qty>
    <Price>-38.4</Price>
    <Tax>-9.6</Tax>
    <TaxPercent>400</TaxPercent>
    <Discount>0</Discount>
    <Comment xsi:nil="true" />
    <Type>NORMAL</Type>
    <ShipmentStatus></ShipmentStatus>
    <ProductName>API tester campaign.</ProductName>
    <VariantName>API tester campaign.</VariantName>
    <Size></Size>
    <Qualifier1 xsi:nil="true" />
    <Qualifier2 xsi:nil="true" />
    <ExtraText1 xsi:nil="true" />
    <ExtraText2 xsi:nil="true" />
    <ExtraText3 xsi:nil="true" />
    <ExtraText4 xsi:nil="true" />
    <ExtraText5 xsi:nil="true" />
    <ExtraText6 xsi:nil="true" />
    <Weight xsi:nil="true" />
    <Volume xsi:nil="true" />
    <Unit xsi:nil="true" />
    <IsBulky>false</IsBulky>
  </Item>
```

3.3.3 Shipments

Collection of shipments for an order

URI:	*Specified per order in "Shipments" element*
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

This resource is a collection of `Shipment` objects that represent all the shipments made for this order. Usually there is only one shipment per order, but there may be more for partially shipped orders or reshipment of defective goods.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ShipmentList>
  <Shipment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/1/shipment/1">
    <Transporter href="/__API__/transporter/2" xsi:nil="true" />
    <Courier>UPS</Courier>
    <ServiceCode xsi:nil="true" />
    <CourierShipmentId xsi:nil="true" />
    <TrackingCode>132465a</TrackingCode>
    <Comment>big box</Comment>
    <Weight xsi:nil="true" />
    <Volume xsi:nil="true" />
    <CreatedTime>2012-05-02T13:00:12Z</CreatedTime>
    <ChangedTime>2012-05-02T13:07:15Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
  </Shipment>
  <Shipment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/1/shipment/2">
    <Transporter href="/__API__/transporter/" xsi:nil="true" />
    <Courier>UPS</Courier>
    <ServiceCode xsi:nil="true" />
    <CourierShipmentId xsi:nil="true" />
    <TrackingCode>132465b</TrackingCode>
    <Comment>second box</Comment>
    <Weight xsi:nil="true" />
    <Volume xsi:nil="true" />
    <CreatedTime>2012-05-02T13:10:01Z</CreatedTime>
    <ChangedTime>2012-05-02T13:10:01Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
  </Shipment>
</ShipmentList>
```

Shipment

URI:	`/__API__/order/.../shipment` (from Shipment href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`Transporter`	*empty*	*read, write*	The "href" attribute references the transporter to be used for this shipment.
`Courier`	*string*	*read, write*	The name of the courier/transporter used for this shipment.
`ServiceCode`	*string*	*read, write*	The service code used for this shipment.
`TrackingCode`	*string*	*read, write*	The transporters tracking code if available. NOTE This is usually NOT a complete tracking link (except for "Generic" transporter method)
`Comment`	*string*	*read, write*	Comment entered for this shipment. MAY be written on parcel.
`Weight`	*string*	*read, write*	Weight of the shipment in KGs.
`Volume`	*string*	*read, write*	Volume of the shipment in litres.
`CreatedTime`	*datetime*	*read*	The date and time when this shipment was created.
`ChangedTime`	*datetime*	*read*	The date and time when this shipment record was last changed.
`SyncedTime`	*datetime*	*read*	The date and time when this shipment record was last synchronized.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Shipment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/order/1/shipment/1">
  <Transporter href="/__API__/transporter/2" xsi:nil="true" />
  <Courier>UPS</Courier>
  <ServiceCode xsi:nil="true" />
  <CourierShipmentId xsi:nil="true" />
  <TrackingCode>132465a</TrackingCode>
  <Comment>big box</Comment>
  <Weight xsi:nil="true" />
  <Volume xsi:nil="true" />
  <CreatedTime>2012-05-02T13:00:12Z</CreatedTime>
  <ChangedTime>2012-05-02T13:07:15Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</Shipment>
```

3.3.4 Capturing / Cancelling payment

Some payment methods have a 2-way model for acquiring payment. This is usually the case for credit card transactions.

In the first step information about the payment is sent to the processor who decides whether the payment will be authorized. This usually results in the order getting the payment state "AUTHORIZED". In the second step the shop informs the payment processor that the payment is to be captured/settled. These processors usually also allow a payment to be cancelled or refunded.

This API can be used to trigger a capture or cancel request to the payment processor by setting the query parameter `payment_action` during an UPDATE(`PUT`) request.

To capture payment for an order the `payment_action` query parameter is set to `capture` and to cancel/refund the payment the parameter is set to `cancel`.

Since this is done during a `PUT` request an XML object needs to be sent. All changes represented in the transmitted XML object will be performed BEFORE the capture/cancel is done. It is proper to send an empty order object if no other changes are to be made to the order.

When the capture/cancel is complete (this may take a few seconds since the payment processor needs to be contacted) an updated Order object is returned to the client. The order object will most likely have some fields (`PaymentState` at least) updated.

If the action did not succeed an error response will be sent. The changes to the order MAY have been processed already so the order MAY be updated.

Errors

There are three possible classes of errors that may be reported when trying to do a capture. They are signaled using different HTTP return codes.

In addition to the HTTP status code each error will also return an XML message detailing the error as described in section 2.9.

400 - Bad request When the client sends a request to the server that is ill formed for some reason the server will respond with status 400 - bad request. One example is if the client sends a erroneous value for the `payment_action` parameter. When the server sends this error it has ensured that no changes were made to the order and the order should be in the same state as before the request.

409 - Conflict This signals that the client is trying to perform an action that is in conflict with the current state of the order, for example if the client tries to perform a capture on an order that is already captured. When the server sends this error it has ensured that no changes were made to the order and the order should be in the same state as before the request.

500 - Server error This is usually only sent if there was an error in communicating with the payment processor or if the payment processor rejected the operation for some reason. When the server sends this error changes MAY have been made to the order. Parts of the update MAY have completed successfully. If the client sent an Order object containing changes then these changes have most likely been performed.

Example capture

Say you have just shipped the order `/__API__/order/42` to the customer and wish wish to change the order state to "FULLY_SHIPPED" and do a capture. To do this you would send this XML message

```
<?xml version=''1.0'' encoding=''ISO-8859-1''?>
<Order>
    <State>FULLY_SHIPPED</State>
</Order>
```

to this URI: `/__API__/order/42?payment_action=capture`. This will update the order state first, and then perform the capture operation on that order.

If you only wish to perform the capture without updating the order state you should send an empty Order object instead:

```
<?xml version=''1.0'' encoding=''ISO-8859-1''?>
<Order>
</Order>
```

3.3.5 Order-states

The order state is used mainly to inform the customer about their order. Some states may have actions associated with them, and customer specific functionality may also be triggered by them.

States may be added, removed or reconfigured by the user. Some states are system states and may not be removed. The system states are listed below:

The "null" state denotes orders that are not yet finished by the customer. Usually this state appears on orders that have not been fully processed by the payment processor.

NEW The order is a new order. The system sets the order to this state as soon as it is finished by the customer. Usually this is when payment processing has been completed by the customer. It does not mean that payment has been received yet.

ACCEPTED This state is not set by the system. It is used to mark an order as accepted by the store owner and usually awaiting further processing. This is a system state since some older synchronization facilities require this state to be present.

3.3.6 Collection of order-states

URI:	`/__API__/order/state`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

This resource is a collection of `OrderState` objects that represent all the order states configured in the shop. The `OrderState` object is defined below.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OrderStates>
  <OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/state/ACCEPTED">
    <State>ACCEPTED</State>
    <Ordering>2</Ordering>
    <SystemState>true</SystemState>
    <Active>true</Active>
    <Name lang="sv">Godkänd</Name>
    <Name lang="en" primary-language="true">Godkänd</Name>
    <Description lang="sv">Godkänd</Description>
    <Description lang="en" primary-language="true">Godkänd</Description>
    <ColorCode>#ffff00</ColorCode>
    <AffiliateState>OK</AffiliateState>
    <TypeInStatistics>INPROGRESS</TypeInStatistics>
    <StockAction>TAKE_STOCK</StockAction>
  </OrderState>
  <OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/state/DENIED">
    <State>DENIED</State>
    <Ordering>8</Ordering>
    <SystemState>false</SystemState>
    <Active>true</Active>
    <Name lang="sv">Vägrad</Name>
    <Name lang="en" primary-language="true">Vägrad</Name>
    <Description lang="sv">Vägrad</Description>
    <Description lang="en" primary-language="true">Vägrad</Description>
    <ColorCode>#ff0000</ColorCode>
    <AffiliateState>BAD</AffiliateState>
    <TypeInStatistics>BAD</TypeInStatistics>
    <StockAction>REVERT_STOCK</StockAction>
  </OrderState>
  <OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/order/state/DOUBLE">
    <State>DOUBLE</State>
    <Ordering>6</Ordering>
    <SystemState>false</SystemState>
    <Active>true</Active>
    <Name lang="sv">Dubbel</Name>
    <Name lang="en" primary-language="true">Dubbel</Name>
    <Description lang="sv">Dubbel</Description>
    <Description lang="en" primary-language="true">Dubbel</Description>
    <ColorCode>#ff0000</ColorCode>
    <AffiliateState>BAD</AffiliateState>
    <TypeInStatistics>BAD</TypeInStatistics>
```

```
43      <StockAction>REVERT_STOCK</StockAction>
44    </OrderState>
45    <OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/state/FULLY_SHIPPED">
46      <State>FULLY_SHIPPED</State>
47      <Ordering>4</Ordering>
48      <SystemState>false</SystemState>
49      <Active>true</Active>
50      <Name lang="sv">Full lev.</Name>
51      <Name lang="en" primary-language="true">Full lev.</Name>
52      <Description lang="sv">Full lev.</Description>
53      <Description lang="en" primary-language="true">Full
            lev.</Description>
54      <ColorCode>#00ff00</ColorCode>
55      <AffiliateState>OK</AffiliateState>
56      <TypeInStatistics>OK</TypeInStatistics>
57      <StockAction>TAKE_STOCK</StockAction>
58    </OrderState>
59    <OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/state/NEW">
60      <State>NEW</State>
61      <Ordering>1</Ordering>
62      <SystemState>true</SystemState>
63      <Active>true</Active>
64      <Name lang="sv">Ny</Name>
65      <Name lang="en" primary-language="true">Ny</Name>
66      <Description lang="sv">Ny</Description>
67      <Description lang="en" primary-language="true">Ny</Description>
68      <ColorCode>#ffff00</ColorCode>
69      <AffiliateState>BAD</AffiliateState>
70      <TypeInStatistics>INPROGRESS</TypeInStatistics>
71      <StockAction>TAKE_STOCK</StockAction>
72    </OrderState>
73    <OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/state/NOT_PICKED_UP">
74      <State>NOT_PICKED_UP</State>
75      <Ordering>7</Ordering>
76      <SystemState>false</SystemState>
77      <Active>true</Active>
78      <Name lang="sv">Outlöst</Name>
79      <Name lang="en" primary-language="true">Outlöst</Name>
80      <Description lang="sv">Outlöst</Description>
81      <Description lang="en" primary-language="true">Outlöst</Description>
82      <ColorCode>#ff0000</ColorCode>
83      <AffiliateState>BAD</AffiliateState>
84      <TypeInStatistics>BAD</TypeInStatistics>
85      <StockAction>REVERT_STOCK</StockAction>
86    </OrderState>
87    <OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/order/state/PARTIALLY_SHIPPED">
88      <State>PARTIALLY_SHIPPED</State>
```

```
89      <Ordering>3</Ordering>
90      <SystemState>false</SystemState>
91      <Active>true</Active>
92      <Name lang="sv">Del lev.</Name>
93      <Name lang="en" primary-language="true">Del lev.</Name>
94      <Description lang="sv">Del lev.</Description>
95      <Description lang="en" primary-language="true">Del lev.</Description>
96      <ColorCode>#ffff00</ColorCode>
97      <AffiliateState>OK</AffiliateState>
98      <TypeInStatistics>INPROGRESS</TypeInStatistics>
99      <StockAction>TAKE_STOCK</StockAction>
100   </OrderState>
101 </OrderStates>
```

3.3.7 OrderState

URI:	`/__API__/order/state/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`State`	*str*	*read, write*	The state ID/tag. This is used internally by the shop and is also the ID present in the "State" element in the Order object. This element MUST NOT be changed for system states. Changing this ID in a used state will invalidate the OrderState object URI, the new URI is sent in the response. Orders that have this state are NOT updated when the state changes!
`Ordering`	*int*	*read, write*	The order of states in all listings of order states. States are ordered in strictly non descending value of this field.
`SystemState`	*bool*	*read*	If "true" indicates that this is a system state.
`Active`	*bool*	*read, write*	Indicates whether this state is active or not. States that are in use by orders should never be removed, just made inactive.
`Name`	*str+lang*	*read, write*	The user/customer friendly name of this order state.
`Description+lang`	*str*	*read, write*	An optional description of this state that may be presented to the customer.
`ColorCode`	*str*	*read, write*	A RGB color code used to mark orders in the shop for easier access. Format is "#rrggbb" where "rr"/ "gg"/ "bb" are hexadecimal representations of the red, green and blue components.
`AffiliateState`	*str*	*read, write*	Which state this order should get in the affiliate-system when its quarantine period is over. Accepted values are "OK" or "BAD".

contd...

Name	Type	Mode	Description
TypeInStatistics	*str*	*read, write*	What type of order this is for statistics purposes. The following types are available: “OK” — this is an OK/completed order. “BAD” — this is a bad/denied order. “INPROGRESS” — this order is in progress. “STATE” — orders with this state are to be summarized separately.
StockAction	*str*	*read, write*	What to do with the stock for this order. The following values are possible: “NOTHING” — Stock processing is not affected at all when changing to this state. “TAKE_STOCK” — Available stock should be decreased for all items if it hasn’t already been done when switching to this state. “REVERT_STOCK” — Available stock should be restored as if these items were never purchased.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<OrderState xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/order/state/ACCEPTED">
  <State>ACCEPTED</State>
  <Ordering>2</Ordering>
  <SystemState>true</SystemState>
  <Active>true</Active>
  <Name lang="sv">Godkänd</Name>
  <Name lang="en" primary-language="true">Godkänd</Name>
  <Description lang="sv">Godkänd</Description>
  <Description lang="en" primary-language="true">Godkänd</Description>
  <ColorCode>#ffff00</ColorCode>
  <AffiliateState>OK</AffiliateState>
  <TypeInStatistics>INPROGRESS</TypeInStatistics>
  <StockAction>TAKE_STOCK</StockAction>
</OrderState>
```

Extrafields

The Order resource supports extrafields for the main order body and also for each order line.

For order data both text fields and selectors are supported, and configuration can be found at `/__API__/order/extrafields`.

For orderlines only text fields are supported. Configuration can be found at `/__API__/order/item/extrafields`.

See section 3.25 for details.

3.4 Payment Methods

The payment method resources can be used by a client to discover the payment methods that are configured in this shop, and to get more information about the payment method chosen for a particular order.

These resources are currently read-only.

3.4.1 Collection payment methods

URI:	`/__API__/paymentmethod`
Actions:	Retrieve(GET)
Content-Type:	text/xml

A collection of `PaymentMethod` objects. This resource lists all payment methods that have been configured in the shop. The `href` attribute contains an URI referencing the PaymentMethod.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PaymentMethodList>
  <PaymentMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/paymentmethod/111">
    <ID>PF-FPAK</ID>
    <Name>PF</Name>
    <FeeName>PF</FeeName>
    <Notes></Notes>
  </PaymentMethod>
  <PaymentMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/paymentmethod/255">
    <ID>PROFORMA</ID>
    <Name>Advance payment</Name>
    <FeeName>Advance payment fee</FeeName>
    <Notes>Pay to our bank account and maybe we'll send you the
       stuff.</Notes>
  </PaymentMethod>
  <PaymentMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/paymentmethod/830">
    <ID>PF-PPAK</ID>
    <Name>PF</Name>
    <FeeName>PF</FeeName>
    <Notes></Notes>
  </PaymentMethod>
  <PaymentMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/paymentmethod/831">
    <ID>FAKT</ID>
```

```
23      <Name>Invoice</Name>
24      <FeeName>Invoice Fee.</FeeName>
25      <Notes>pay within 30 days</Notes>
26    </PaymentMethod>
27  </PaymentMethodList>
```

3.4.2 PaymentMethod object

URI:	`/__API__/paymentmethod/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read*	The tag/ID as configured by the user. Multiple methods MAY have the same ID. This ID MAY be used as an external reference to a payment method or payment terms in an ERP system.
`Name`	*str+lang*	*read*	The name of this method, as displayed to the customer.
`FeeName`	*str+lang*	*read*	The name of the payment fee as displayed to the customer.
`Notes`	*str+lang*	*read*	Additional notes that are displayed to the customer.
`Locks`	*array*	*read, write*	A list of locks assigned to this payment method. See section 3.4.2 and 3.26 for details.
`AntiLocks`	*array*	*read, write*	A list of anti locks assigned to this payment method. See section 3.4.2 and 3.26 for details.

Object description for "Locks"

An array of `Locks` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the lock. 0..63
`Value`	*bool*	*read, write*	If lock is active then true, otherwise false

Object description for "AntiLocks"

An array of `AntiLocks` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered anti locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the anti lock. 0..63
`Value`	*bool*	*read, write*	If anti lock is active then true, otherwise false

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PaymentMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/paymentmethod/111">
  <ID>PF-FPAK</ID>
  <Name lang="sv">Postförskott Företagspaket</Name>
  <Name lang="en" primary-language="true">PF</Name>
  <FeeName lang="sv">PF Avgift</FeeName>
  <FeeName lang="en" primary-language="true">PF</FeeName>
  <Notes lang="sv">Betalas vid utkörning eller senare på postens
      serviceställe.</Notes>
  <Notes lang="en" primary-language="true"></Notes>
  <Locks>
      <Lock>
          <Index>0</Index>
          <Value>false</Value>
      </Lock>
      <Lock>
          <Index>7</Index>
          <Value>false</Value>
      </Lock>
  </Locks>
  <AntiLocks>
      <AntiLock>
          <Index>0</Index>
          <Value>false</Value>
      </AntiLock>
      <AntiLock>
          <Index>7</Index>
          <Value>false</Value>
      </AntiLock>
  </AntiLocks>
</PaymentMethod>
```

3.5 Shipping Methods

The shipping method resources can be used by a client to discover the shipping methods that are configured in this shop, and to get more information about the shipping method chosen for a particular order.

These resources are currently read-only.

3.5.1 Collection shipping methods

URI:	`/__API__/shippingmethod`
Actions:	Retrieve(GET)
Content-Type:	text/xml

A collection of `ShippingMethod` objects. This resource lists all shipping methods that have been configured in the shop. The `href` attribute contains an URI referencing the ShippingMethod.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <ShippingMethodList>
3   <ShippingMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/shippingmethod/117">
4     <ID>F-PAK</ID>
5     <Name>Parcel</Name>
6     <FeeName>Parcel</FeeName>
7     <Notes></Notes>
8   </ShippingMethod>
9   <ShippingMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/shippingmethod/257">
10    <ID>HÄMT</ID>
11    <Name>Pickup</Name>
12    <FeeName>Pickup</FeeName>
13    <Notes></Notes>
14  </ShippingMethod>
15  <ShippingMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/shippingmethod/844">
16    <ID>P-PAK</ID>
17    <Name>Another parcel</Name>
18    <FeeName>Another parcel</FeeName>
19    <Notes></Notes>
20  </ShippingMethod>
21 </ShippingMethodList>
```

3.5.2 ShippingMethod object

URI:	`/__API__/shippingmethod/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read*	The tag/ID as configured by the user. Multiple methods MAY have the same ID. This ID MAY be used as an external reference to a shipping method or shipping terms in an ERP system.
`Name`	*str+lang*	*read*	The name of this method, as displayed to the customer.
`FeeName`	*str+lang*	*read*	The name of the shipping fee as displayed to the customer.
`Notes`	*str+lang*	*read*	Additional notes that are displayed to the customer.
`Locks`	*array*	*read, write*	A list of locks assigned to this shipping method. See section 3.5.2 and 3.26 for details.
`AntiLocks`	*array*	*read, write*	A list of anti locks assigned to this shipping method. See section 3.5.2 and 3.26 for details.

Object description for "Locks"

An array of `Locks` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the lock. 0..63
`Value`	*bool*	*read, write*	If lock is active then true, otherwise false

Object description for "AntiLocks"

An array of `AntiLocks` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered anti locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the anti lock. 0..63
`Value`	*bool*	*read, write*	If anti lock is active then true, otherwise false

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <ShippingMethod xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/shippingmethod/117">
```

```
3   <ID>F-PAK</ID>
4   <Name lang="sv">Företagspaket</Name>
5   <Name lang="en" primary-language="true">Parcel</Name>
6   <FeeName lang="sv">Frakt</FeeName>
7   <FeeName lang="en" primary-language="true">Parcel</FeeName>
8   <Notes lang="sv">Utkörning till adressen innan kl. 16.00 nästa
        dag.</Notes>
9   <Notes lang="en" primary-language="true"></Notes>
10  <Locks>
11      <Lock>
12          <Index>0</Index>
13          <Value>false</Value>
14      </Lock>
15      <Lock>
16          <Index>7</Index>
17          <Value>false</Value>
18      </Lock>
19  </Locks>
20  <AntiLocks>
21      <AntiLock>
22          <Index>0</Index>
23          <Value>false</Value>
24      </AntiLock>
25      <AntiLock>
26          <Index>7</Index>
27          <Value>false</Value>
28      </AntiLock>
29  </AntiLocks>
30 </ShippingMethod>
```

3.6 Transporters

This resource lists the transporters configured for this shop. It can be used to discover which transporter to use when sending shipment data for an order.

These resources are currently read-only.

3.6.1 Collection of transporters

URI:	`/__API__/transporter`
Actions:	Retrieve(GET)
Content-Type:	text/xml

A collection of `Transporter` objects. This resource lists all transporters that have been configured in the shop. The `href` attribute contains an URI referencing the Transporter.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <TransporterList>
3   <Transporter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/transporter/1">
4     <Method>Posten</Method>
5     <Courier>Posten</Courier>
6     <Login xsi:nil="true" />
7   </Transporter>
8   <Transporter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/transporter/2">
9     <Method>UPS</Method>
10    <Courier>UPS</Courier>
11    <Login xsi:nil="true" />
12  </Transporter>
13 </TransporterList>
```

3.6.2 Transporter object

URI:	`/__API__/transporter/...` (from href attribute)
Actions:	Retrieve(GET)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
Method	*str*	*read*	The method ID within the shop system. This controls which implementation will be used when communicating with the transporter.
Courier	*str*	*read*	The courier that will be used for the delivery. Usually a user specified string.
Login	*str*	*read*	The login ID used when communicating with the transporter.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Transporter xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/transporter/2">
3   <Method>UPS</Method>
4   <Courier>UPS</Courier>
5   <Login xsi:nil="true" />
6 </Transporter>
```

3.7 Products and Variants

These resources are used to access products and product variants in the system.

Each product MUST have at least one variant since the product is basically just a container for the variants. Think of the product as being a "Category 6 network cable" and the variants as "Cat6 3m", "Cat6 5m", etc.

Because variants are so tightly coupled with their products each product has its own variant collection which is specified in the `Product` object.

The variants contain pricing information which might be quite complex. The pricing information is contained in the `PriceInformation` element in the `Variant` object. This object is an array of elements that contain pricing information relating to some specified price list[2]. Each element contains pricing information for different quantities if pricing for different quantities is supported.

The product resources support the synchronization procedure described in section 2.11 with some extensions so synchronization takes both products and product variants into account.

3.7.1 Product Collection

URI:	`/__API__/product`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, view, mark_as_synced, product_with_sku, filter
POST Parameters:	mark_as_synced, generate_seo_url

Query parameters

Name	Values	Description
`synced`	*yes, productnotsynced, no*	As defined in section 2.11.2 with the following extensions: This option checks the synchronized state of both the Product and its Variants. So the product itself may be in sync, but one of the variants may not be and the product will then be treated as not synchronized. The parameter may be sent as "productnotsynced" to only list those products where the *product* record is not in sync, ignoring the variant records.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.

[2] If the shop doesn't support multiple price lists then exactly one item will be present.

view	*short,long*	If specified as "long" then a complete "Product" object is sent for each product.
product_with_sku	*str*	If specified only a product that has a variant with a SKU matching the value of this parameter will be returned. Useful when you need to find out which product has a certain SKU. The object returned will contain all the usual elements and also a "Variant" element with a "href" attribute refering to the variant matched. NOTE: This conflicts with the sync parameters, and may not be used if syncing parameters are sent.
generate_seo_url	*0/1*	If specified and set to 1, the system will use the "Name" field to generate an SEO-URL. The "Name" element MUST be present. The "SEOURL" element MUST NOT be present.

Additional filter parameters

These parameters can be used in addition to the object parameters marked as filterable.

Name	Type	Description
ProductOrVariantChangedTime	*datetime*	The changed time of both product and all the variants for this product. *ChangedTime* in product object only matches the Product change time.

Object description

The returned object <ProductList>...</ProductList> is an array of abbreviated Product objects:

Name	Type	Mode	Description
ID	*str*	*read*	The ID of this category. Must be unique.
Type	*str*	*read*	Type of product.
Name	*str+lang*	*read*	The product name.
State	*str+lang*	*read*	One of "HIDDEN", "ACTIVE".
VariantList	*empty*	*read*	A reference to the Variant collection for this object. See section 3.7.3
Variant	*empty*	*read*	A reference to the Variant with the SKU given as a parameter (product_with_sku). This is only available when the parameter product_with_sku is used. See section 3.7.4
CreatedTime	*datetime*	*read*	The date and time when this record was created.
ChangedTime	*datetime*	*read*	The date and time when this record was last changed.
SyncedTime	*datetime*	*read*	The date and time when this record was last synchronized.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ProductList>
  <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      product_id="8" href="/__API__/product/8">
    <ID>1</ID>
    <Type>NORMAL</Type>
    <Name lang="sv">Produkt nummer ett</Name>
    <Name lang="en" primary-language="true">Produkt nummer ett</Name>
    <State lang="sv">ACTIVE</State>
    <State lang="en" primary-language="true">ACTIVE</State>
    <VariantList href="/__API__/product/8/variant" xsi:nil="true" />
    <CreatedTime>2007-11-22T07:00:12Z</CreatedTime>
    <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
    <SyncedTime>2009-04-15T09:25:47Z</SyncedTime>
  </Product>
  <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      product_id="9" href="/__API__/product/9">
    <ID xsi:nil="true" />
    <Type>NORMAL</Type>
    <Name lang="sv">Produkt nummer två</Name>
    <Name lang="en" primary-language="true">Produkt nummer två</Name>
    <State lang="sv">ACTIVE</State>
    <State lang="en" primary-language="true">ACTIVE</State>
    <VariantList href="/__API__/product/9/variant" xsi:nil="true" />
    <CreatedTime>2007-11-22T07:14:19Z</CreatedTime>
    <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
    <SyncedTime>2009-04-15T09:25:47Z</SyncedTime>
  </Product>
  <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      product_id="10" href="/__API__/product/10">
    <ID xsi:nil="true" />
    <Type>NORMAL</Type>
    <Name lang="sv">Produkt nummer tre</Name>
    <Name lang="en" primary-language="true">Produkt nummer tre</Name>
    <State lang="sv">ACTIVE</State>
    <State lang="en" primary-language="true">ACTIVE</State>
    <VariantList href="/__API__/product/10/variant" xsi:nil="true" />
    <CreatedTime>2007-11-22T07:25:14Z</CreatedTime>
    <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
    <SyncedTime>2009-04-15T09:25:47Z</SyncedTime>
  </Product>
  <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      product_id="11" href="/__API__/product/11">
    <ID xsi:nil="true" />
    <Type>NORMAL</Type>
    <Name lang="sv">Produkt nummer fyra</Name>
    <Name lang="en" primary-language="true">Produkt nummer fyra</Name>
    <State lang="sv">ACTIVE</State>
    <State lang="en" primary-language="true">ACTIVE</State>
```

```
46      <VariantList href="/__API__/product/11/variant" xsi:nil="true" />
47      <CreatedTime>2007-11-22T07:26:02Z</CreatedTime>
48      <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
49      <SyncedTime>2009-04-15T09:25:47Z</SyncedTime>
50    </Product>
51    <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          product_id="13" href="/__API__/product/13">
52      <ID xsi:nil="true" />
53      <Type>NORMAL</Type>
54      <Name lang="sv">Produkt nummer fem</Name>
55      <Name lang="en" primary-language="true">Produkt nummer fem</Name>
56      <State lang="sv">ACTIVE</State>
57      <State lang="en" primary-language="true">ACTIVE</State>
58      <VariantList href="/__API__/product/13/variant" xsi:nil="true" />
59      <CreatedTime>2007-11-22T07:30:58Z</CreatedTime>
60      <ChangedTime>2016-02-02T15:06:50Z</ChangedTime>
61      <SyncedTime>2009-04-15T09:25:47Z</SyncedTime>
62    </Product>
63    <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          product_id="14" href="/__API__/product/14">
64      <ID xsi:nil="true" />
65      <Type>NORMAL</Type>
66      <Name lang="sv">Produkt nummer sex</Name>
67      <Name lang="en" primary-language="true">Produkt nummer sex</Name>
68      <State lang="sv">ACTIVE</State>
69      <State lang="en" primary-language="true">ACTIVE</State>
70      <VariantList href="/__API__/product/14/variant" xsi:nil="true" />
71      <CreatedTime>2007-11-22T07:31:22Z</CreatedTime>
72      <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
73      <SyncedTime>2009-04-15T09:25:47Z</SyncedTime>
74    </Product>
75    <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          product_id="15" href="/__API__/product/15">
76      <ID xsi:nil="true" />
77      <Type>NORMAL</Type>
78      <Name lang="sv">Produkt nummer sju</Name>
79      <Name lang="en" primary-language="true">Produkt nummer sju</Name>
80      <State lang="sv">ACTIVE</State>
81      <State lang="en" primary-language="true">ACTIVE</State>
82      <VariantList href="/__API__/product/15/variant" xsi:nil="true" />
83      <CreatedTime>2007-11-22T07:31:41Z</CreatedTime>
84      <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
85      <SyncedTime>2009-04-15T09:34:25Z</SyncedTime>
86    </Product>
87    <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          product_id="16" href="/__API__/product/16">
88      <ID xsi:nil="true" />
89      <Type>NORMAL</Type>
90      <Name lang="sv">Produkt nummer åtta</Name>
91      <Name lang="en" primary-language="true">Produkt nummer åtta</Name>
92      <State lang="sv">ACTIVE</State>
```

```
93      <State lang="en" primary-language="true">ACTIVE</State>
94      <VariantList href="/__API__/product/16/variant" xsi:nil="true" />
95      <CreatedTime>2007-11-22T07:32:15Z</CreatedTime>
96      <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
97      <SyncedTime>2009-04-15T09:23:45Z</SyncedTime>
98    </Product>
99    <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          product_id="17" href="/__API__/product/17">
100     <ID xsi:nil="true" />
101     <Type>NORMAL</Type>
102     <Name lang="sv">API tester campaign.</Name>
103     <Name lang="en" primary-language="true">API tester campaign.</Name>
104     <State lang="sv">ACTIVE</State>
105     <State lang="en" primary-language="true">ACTIVE</State>
106     <VariantList href="/__API__/product/17/variant" xsi:nil="true" />
107     <CreatedTime>2012-08-23T14:22:21Z</CreatedTime>
108     <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
109     <SyncedTime xsi:nil="true" />
110   </Product>
111   <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          product_id="18" href="/__API__/product/18">
112     <ID xsi:nil="true" />
113     <Type>SUBSCRIPTION</Type>
114     <Name lang="sv">Subscription Product 1</Name>
115     <Name lang="en" primary-language="true">Subscription Product 1</Name>
116     <State lang="sv">ACTIVE</State>
117     <State lang="en" primary-language="true">ACTIVE</State>
118     <VariantList href="/__API__/product/18/variant" xsi:nil="true" />
119     <CreatedTime>2010-01-25T13:26:00Z</CreatedTime>
120     <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
121     <SyncedTime xsi:nil="true" />
122   </Product>
123   <Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          product_id="19" href="/__API__/product/19">
124     <ID xsi:nil="true" />
125     <Type>SUBSCRIPTION</Type>
126     <Name lang="sv">Subscription Product 2</Name>
127     <Name lang="en" primary-language="true">Subscription Product 2</Name>
128     <State lang="sv">ACTIVE</State>
129     <State lang="en" primary-language="true">ACTIVE</State>
130     <VariantList href="/__API__/product/19/variant" xsi:nil="true" />
131     <CreatedTime>2010-01-25T13:27:29Z</CreatedTime>
132     <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
133     <SyncedTime xsi:nil="true" />
134   </Product>
135 </ProductList>
```

3.7.2 Product

URI:	`/__API__/product/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
GET Parameters:	mark_as_synced
PUT Parameters:	mark_as_synced, generate_seo_url
Attributes:	href, product_id

Query parameters

Name	Values	Description
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`generate_seo_url`	*0/1*	If specified and set to 1, the system will use the "Name" field to generate an SEO-URL. The "Name" element MUST be present. The "SEOURL" element MUST NOT be present.

Object description

The `Product` element contains a "href" attribute specifying the complete URI for this product. It also contains a "product_id" attribute that is the product_id parameter used in the shop front end. This might be used by the API user to create links that link into a specific product in the shop.

The elements in this object are described below.

Name	Type	Mode	Description
`ID`	*str*	*read, write, filter*	The ID of this product. Must be unique.
`Type`	*str*	*read, filter*	Type of product. Possible values are: *NORMAL* — This is a normal product, *SUBSCRIPTION* — This product is a subscription product.
`QtyPriceMode`	*str*	*read*	Mode for calculating quantity-discounts in shop. Allowed values are "VARIANT" or "PRODUCT".
`Name`	*str+lang*	*read, write*	The name.
`ShortDescription`	*str+lang*	*read, write*	A short non-html description.
`Description`	*str+lang*	*read, write*	A description of this product provided by the shop. This field is displayed to the customer on the web and MAY contain HTML as long as the HTML is properly coded as an XML string.
`ExtraDescription`	*str+lang*	*read, write*	Extra description text.
`Promote`	*bool*	*read, write*	Indicates wether this is a promoted product.
`State`	*str+lang*	*read, write, filter*	One of "HIDDEN", "ACTIVE".

contd...

Name	Type	Mode	Description
ExtraText1..10	*str+lang*	*read, write, filter*	Extra text parameters.
ExtraSelector1..10	*int*	*read, write, filter*	Extra select fields, defined in the shop. Can only take on values between 0 and 256, unlike most integers. If set the attribute *textvalue* will contain a textual value corresponding to the numeric index.
ExtraHiddenText1..10	*str*	*read, write*	Extra text parameters that is not visible in the administration.
SEOTitle	*str+lang*	*read, write*	For SEO: The title displayed on the product page in the shop.
SEODescription	*str+lang*	*read, write*	For SEO: The meta-description displayed on the product page in the shop.
SEOKeywords	*str+lang*	*read, write*	For SEO: The meta-keywords displayed on the product page in the shop.
SEOURL	*str+lang*	*read, write*	The URL component for SEO-friendly URLs. These MUST be unique between products.
Campaign	*str*	*read, write*	The campaign this product belongs to. The "href" attribute contains the campaign URI. The element itself contains the campaign ID if it is specified. The ID is ignored by the shop. It is only sent to aid API users. NOTE: If a product is removed from a campaign all of its Variants MUST also have their *CampaignPrice* removed!
MainCategory	*empty*	*read, write*	A Category reference. The "href" attribute contains the URI for the category object. Sending the Category element without a "href" tag removes the product from its main category, which is probably a very bad idea.
ExtraCategories	*array*	*read, write*	An array containing Category references whose "href" attribute contains the URI for the category object.
Brand	*empty*	*read, write*	A Brand reference. The "href" attribute contains the URI for the brand object. Sending the Brand element without a "href" tag removes the brand.
VariantList	*empty*	*read*	A reference to the Variant collection for this product. The "href" attribute contains the URI for the collection.
ProductPictureList	*array+lang*	*read, write*	An array of ProductPicture objects referring to all the pictures of this product. For shops and API users supporting multiple languages this element will appear multiple times, once for each language. If this element is sent then all the product pictures for this product for that language will be changed to the references sent.

contd...

Name	Type	Mode	Description
LinkedProducts	*array*	*read, write*	An array of products or product variants that this product links to. If this element is sent from the client then ALL linked items MUST be sent since the system will remove any items not present. Omitting the array entirely DOES NOT remove any linked products. Also note that this array is unordered.
Documents	*array*	*read, write*	An array of "Document" elements containing a "href" attribute identifying the document.
Symbols	*array+lang*	*read, write*	An array of "Symbol" elements containing a "href" attribute identifying each symbol linked to this product. For shops and API users supporting multiple languages this element will appear multiple times, once for each language. If this element is sent then all the symbols linked to this product for that language will be changed to the references sent. See 3.7.2 for object description.
Tax	*float*	*read, write*	The tax rate for this product and all its variants. Only tax rates configured in the shop are allowed.
GroupVariantsBy	*array*	*read, write*	This element contains all fields by which variants of this product should be grouped. The values may be any combination of "ExtraText1" – "ExtraText20" as empty XML elements. The element may also contain a "CustomAttributes" block that lists all custom attributes to be used for grouping.
CustomAttributes	*array*	*read, write*	A list of custom attributes and the values they are set to for this Product. NOTE: Unlike Variants products MAY have multiple values for each attribute! See section 3.7.4 and 3.11 for details.
Locks	*array*	*read, write*	A list of locks assigned to this product. See section 3.7.2 and 3.26 for details.
AntiLocks	*array*	*read, write*	A list of anti locks assigned to this product. See section 3.7.2 and 3.26 for details.
CreatedTime	*datetime*	*read, filter*	The date and time when this record was created.
ChangedTime	*datetime*	*read, filter*	The date and time when this record was last changed.
SyncedTime	*datetime*	*read, filter*	The date and time when this record was last synchronized.

Object description for "ProductPicture"

References a picture in a particular view. Most shops have a default view called "DEFAULT" which is the main product picture, and three extra views called 2-4 for additional product pictures. This is, however, shop dependent and MAY be changed depending on the customer requirements, but it is unlikely that the "DEFAULT" view will be removed.

Name	Type	Mode	Description

contd...

Name	Type	Mode	Description
Picture	*empty*	*read, write*	A Picture reference. The "href" attribute contains the URI for the picture object.
ViewID	*str*	*read, write*	The ID of the view. All products SHOULD have a picture for at least the "DEFAULT" view.
ViewName	*str*	*read*	The name of the view.
Comment	*str*	*read, write*	A comment for the picture.
CreatedTime	*datetime*	*read*	The date and time when this record was created.
ChangedTime	*datetime*	*read*	The date and time when this record was last changed.

Object description for "LinkedProducts" items

Name	Type	Mode	Description
Product	*empty*	*read, write*	A Product tag with a "href" attribute identifying the linked product. The object MUST NOT contain both this tag and the "Variant" tag.
Variant	*empty*	*read, write*	A Variant tag with a "href" attribute identifying the linked variant. The object MUST NOT contain both this tag and the "Product" tag.
Label	*str*	*read, write*	The label for this reference.
SortOrder	*int*	*read, write*	The ordering of the references.
ExtraText1..4	*str*	*read, write*	Extra text parameters.
ExtraSelector1..4	*int*	*read, write*	Extra select fields, defined in the shop. Can only take on values between 0 and 256, unlike most integers. If set the attribute *textvalue* will contain a textual value corresponding to the numeric index.

Object description for "Symbol"

Name	Type	Mode	Description
URL	*str*	*read, write*	URL that this symbol should link to.
ToolTip	*str*	*read, write*	Tooltip shown for symbol.

Object description for "Locks"

An array of Locks blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
Index	*int*	*read, write*	The index for the lock. 0..63
Value	*bool*	*read, write*	If lock is active then true, otherwise false

Object description for “AntiLocks”

An array of `AntiLocks` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered anti locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the anti lock. 0..63
`Value`	*bool*	*read, write*	If anti lock is active then true, otherwise false

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Product xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    product_id="8" href="/__API__/product/8">
  <ID>1</ID>
  <Type>NORMAL</Type>
  <QtyPriceMode>VARIANT</QtyPriceMode>
  <Name lang="sv">Produkt nummer ett</Name>
  <Name lang="en" primary-language="true">Produkt nummer ett</Name>
  <ShortDescription lang="sv">En kortfattad beskrivning för produkten
      syns här</ShortDescription>
  <ShortDescription lang="en" primary-language="true">En kortfattad
      beskrivning för produkten syns här</ShortDescription>
  <Description lang="sv">&lt;p&gt;Lorem ipsum dolor sit amet,
      consectetuer adipiscing elit. Suspendisse enim nunc, elementum sit
      amet, tincidunt et, gravida non, justo. Vivamus pede. Vestibulum
      augue. Aliquam risus. Suspendisse nec est a lectus fermentum
      elementum. Pellentesque sollicitudin feugiat purus. Donec ut
      mauris. Curabitur molestie congue magna. In tempor elit id nulla.
      Phasellus sed tortor. Sed id urna in lectus bibendum venenatis.
      Aliquam ut mi. Aenean a mauris sed ligula feugiat blandit.
      Suspendisse metus neque, porta id, tincidunt pretium, euismod sed,
      massa. Nunc hendrerit elementum tortor. Quisque arcu sem, elementum
      nec, imperdiet sed, posuere lacinia, felis. Nam bibendum nulla at
      nisl. Nunc id nisi id lectus iaculis auctor. Quisque a
      arcu.&lt;/p&gt;</Description>
  <Description lang="en" primary-language="true">&lt;p&gt;Lorem ipsum
      dolor sit amet, consectetuer adipiscing elit. Suspendisse enim
      nunc, elementum sit amet, tincidunt et, gravida non, justo. Vivamus
      pede. Vestibulum augue. Aliquam risus. Suspendisse nec est a lectus
      fermentum elementum. Pellentesque sollicitudin feugiat purus. Donec
      ut mauris. Curabitur molestie congue magna. In tempor elit id
      nulla. Phasellus sed tortor. Sed id urna in lectus bibendum
      venenatis. Aliquam ut mi. Aenean a mauris sed ligula feugiat
      blandit. Suspendisse metus neque, porta id, tincidunt pretium,
      euismod sed, massa. Nunc hendrerit elementum tortor. Quisque arcu
      sem, elementum nec, imperdiet sed, posuere lacinia, felis. Nam
      bibendum nulla at nisl. Nunc id nisi id lectus iaculis auctor.
      Quisque a arcu.&lt;/p&gt;</Description>
```

```
<ExtraDescription lang="sv" xsi:nil="true" />
<ExtraDescription lang="en" primary-language="true" xsi:nil="true" />
<Promote>false</Promote>
<State lang="sv">ACTIVE</State>
<State lang="en" primary-language="true">ACTIVE</State>
<ExtraText1 lang="sv"></ExtraText1>
<ExtraText1 lang="en" primary-language="true"></ExtraText1>
<ExtraText2 lang="sv" xsi:nil="true" />
<ExtraText2 lang="en" primary-language="true" xsi:nil="true" />
<ExtraText3 lang="sv" xsi:nil="true" />
<ExtraText3 lang="en" primary-language="true" xsi:nil="true" />
<ExtraText4 lang="sv" xsi:nil="true" />
<ExtraText4 lang="en" primary-language="true" xsi:nil="true" />
<ExtraText5 lang="sv" xsi:nil="true" />
<ExtraText5 lang="en" primary-language="true" xsi:nil="true" />
<ExtraText6 lang="sv" xsi:nil="true" />
<ExtraText6 lang="en" primary-language="true" xsi:nil="true" />
<ExtraText7 lang="sv" xsi:nil="true" />
<ExtraText7 lang="en" primary-language="true" xsi:nil="true" />
<ExtraText8 lang="sv" xsi:nil="true" />
<ExtraText8 lang="en" primary-language="true" xsi:nil="true" />
<ExtraSelector1>0</ExtraSelector1>
<ExtraSelector2>0</ExtraSelector2>
<ExtraSelector3>0</ExtraSelector3>
<ExtraSelector4>0</ExtraSelector4>
<ExtraSelector5>0</ExtraSelector5>
<ExtraSelector6>0</ExtraSelector6>
<SEOTitle lang="sv" xsi:nil="true" />
<SEOTitle lang="en" primary-language="true" xsi:nil="true" />
<SEODescription lang="sv" xsi:nil="true" />
<SEODescription lang="en" primary-language="true" xsi:nil="true" />
<SEOKeywords lang="sv" xsi:nil="true" />
<SEOKeywords lang="en" primary-language="true" xsi:nil="true" />
<SEOURL lang="sv">produkt-nummer-ett</SEOURL>
<SEOURL lang="en" primary-language="true">produkt-nummer-ett</SEOURL>
<Campaign href="/__API__/campaign/1">c1</Campaign>
<MainCategory href="/__API__/category/6" xsi:nil="true" />
<ExtraCategories>
  <Category href="/__API__/category/7" />
  <Category href="/__API__/category/9" />
</ExtraCategories>
<Brand href="/__API__/brand/1" xsi:nil="true" />
<VariantList href="/__API__/product/8/variant" xsi:nil="true" />
<ProductPictureList lang="sv">
  <ProductPicture>
    <Picture href="/__API__/picture/75" xsi:nil="true" />
    <ViewID>DEFAULT</ViewID>
    <ViewName>Bild 1</ViewName>
    <Comment></Comment>
    <CreatedTime xsi:nil="true" />
    <ChangedTime xsi:nil="true" />
```

```
    </ProductPicture>
  </ProductPictureList>
  <ProductPictureList lang="en" primary-language="true">
    <ProductPicture>
      <Picture href="/__API__/picture/74" xsi:nil="true" />
      <ViewID>DEFAULT</ViewID>
      <ViewName>Bild 1</ViewName>
      <Comment></Comment>
      <CreatedTime xsi:nil="true" />
      <ChangedTime xsi:nil="true" />
    </ProductPicture>
  </ProductPictureList>
  <LinkedProducts>
    <Item>
      <Variant href="/__API__/product/9/variant/13" xsi:nil="true" />
      <Label></Label>
      <SortOrder>2</SortOrder>
      <ExtraText1 xsi:nil="true" />
      <ExtraText2 xsi:nil="true" />
      <ExtraText3 xsi:nil="true" />
      <ExtraText4 xsi:nil="true" />
      <ExtraSelector1>0</ExtraSelector1>
      <ExtraSelector2>0</ExtraSelector2>
      <ExtraSelector3>0</ExtraSelector3>
      <ExtraSelector4>0</ExtraSelector4>
    </Item>
    <Item>
      <Product href="/__API__/product/8" xsi:nil="true" />
      <Label></Label>
      <SortOrder>1</SortOrder>
      <ExtraText1 xsi:nil="true" />
      <ExtraText2 xsi:nil="true" />
      <ExtraText3 xsi:nil="true" />
      <ExtraText4 xsi:nil="true" />
      <ExtraSelector1>0</ExtraSelector1>
      <ExtraSelector2>0</ExtraSelector2>
      <ExtraSelector3>0</ExtraSelector3>
      <ExtraSelector4>0</ExtraSelector4>
    </Item>
  </LinkedProducts>
  <Documents>
    <Document href="/__API__/document/1" />
  </Documents>
  <GroupVariantsBy>
    <ExtraText1 xsi:nil="true" />
    <ExtraText2 xsi:nil="true" />
    <CustomAttributes>
      <CustomAttribute href="/__API__/customattribute/1" />
    </CustomAttributes>
  </GroupVariantsBy>
  <Tax>25</Tax>
```

```
114   <Symbols lang="sv">
115     <Symbol href="/__API__/symbol/1">
116       <URL xsi:nil="true" />
117       <ToolTip xsi:nil="true" />
118     </Symbol>
119   </Symbols>
120   <Symbols lang="en" primary-language="true">
121     <Symbol href="/__API__/symbol/1">
122       <URL xsi:nil="true" />
123       <ToolTip xsi:nil="true" />
124     </Symbol>
125   </Symbols>
126   <CustomAttributes>
127     <CustomAttribute href="/__API__/customattribute/1">
128       <ID xsi:nil="true" />
129       <ValueID xsi:nil="true" />
130       <Name>Size</Name>
131       <Value href="/__API__/customattribute/1/value/1">S</Value>
132     </CustomAttribute>
133     <CustomAttribute href="/__API__/customattribute/2">
134       <ID xsi:nil="true" />
135       <ValueID xsi:nil="true" />
136       <Name>Color</Name>
137       <Value href="/__API__/customattribute/2/value/5">Magenta</Value>
138       <Value href="/__API__/customattribute/2/value/4">Cyan</Value>
139     </CustomAttribute>
140   </CustomAttributes>
141   <Locks>
142       <Lock>
143           <Index>0</Index>
144           <Value>false</Value>
145       </Lock>
146       <Lock>
147           <Index>7</Index>
148           <Value>false</Value>
149       </Lock>
150   </Locks>
151   <AntiLocks>
152       <AntiLock>
153           <Index>0</Index>
154           <Value>false</Value>
155       </AntiLock>
156       <AntiLock>
157           <Index>7</Index>
158           <Value>false</Value>
159       </AntiLock>
160   </AntiLocks>
161   <CreatedTime>2007-11-22T07:00:12Z</CreatedTime>
162   <ChangedTime>2012-08-23T14:23:37Z</ChangedTime>
163   <SyncedTime>2009-04-15T09:25:47Z</SyncedTime>
164 </Product>
```

Extrafields

The Product resource supports extrafields, both text fields and selector fields. Extrafields are configurable at `/__API__/product/extrafields`. See section 3.25 for details.

3.7.3 Variant Collection

URI:	*Depends on product*
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, view, mark_as_synced
POST Parameters:	mark_as_synced

Query parameters

Name	Values	Description
`synced`	*yes, no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`view`	*short,long*	If specified as "long" then a complete object is sent.

Object description

The returned object `<VariantList>...</VariantList>` is an array of abbreviated `Variant` objects:

Name	Type	Mode	Description
`SKU`	*str*	*read, write*	The SKU for this variant.
`Name`	*str*	*read, write*	The name.
`State`	*str*	*read, write*	One of "HIDDEN", "ACTIVE".
`CreatedTime`	*datetime*	*read*	The date and time when this record was created.
`ChangedTime`	*datetime*	*read*	The date and time when this record was last changed.
`SyncedTime`	*datetime*	*read*	The date and time when this record was last synchronized.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <VariantList>
3   <Variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/product/8/variant/12" variant_id="12">
4     <SKU>1</SKU>
5     <Name lang="sv">Produkt nummer ett</Name>
```

```
    <Name lang="en" primary-language="true">Produkt nummer ett</Name>
    <State>ACTIVE</State>
    <CreatedTime>2007-11-22T07:00:12Z</CreatedTime>
    <ChangedTime>2012-09-06T09:14:40Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
  </Variant>
  <Variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/product/8/variant/27" variant_id="27">
    <SKU>1.1</SKU>
    <Name lang="sv">Produkt nummer ett A</Name>
    <Name lang="en" primary-language="true">Produkt nummer ett A</Name>
    <State>ACTIVE</State>
    <CreatedTime>2007-11-22T08:57:59Z</CreatedTime>
    <ChangedTime>2014-02-21T09:45:57Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
  </Variant>
  <Variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/product/8/variant/29" variant_id="29">
    <SKU>1.1b</SKU>
    <Name lang="sv">Produkt nummer ett B</Name>
    <Name lang="en" primary-language="true">Produkt nummer ett B</Name>
    <State>ACTIVE</State>
    <CreatedTime>2009-04-07T14:37:20Z</CreatedTime>
    <ChangedTime>2012-09-06T09:14:40Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
  </Variant>
</VariantList>
```

3.7.4 Variant

URI:	*Variant specific.*
URI:	`/__API__/product/.../variant` (from Variant href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
GET Parameters:	mark_as_synced
PUT Parameters:	mark_as_synced, keep_pricelists
Attributes:	href, variant_id

Query parameters

Name	Values	Description
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.

keep_pricelists	*yes,no*	Normally all prices for all pricelists MUST be sent when updating the price of the variant. The variant will be removed from all other price lists. If this parameter is set to "yes" then the system DOES NOT remove the variant from price lists for which price information was not sent.

Object description

The Variant element contains a "href" attribute specifying the complete URI for this variant. It also contains a "variant_id" attribute that is the variant_id parameter used in the shop front end. This might be used by the API user to create links that link into a specific variant in the shop.

Name	Type	Mode	Description
SKU	*str*	*read, write*	The SKU for this variant.
Name	*str+lang*	*read, write*	The name.
ManufacturerSKU	*str*	*read, write*	The Manufacturers SKU for this variant.
Barcode	*str*	*read, write*	A barcode (preferably EAN) for this variant.
Weight	*float*	*read, write*	The weight of this item.
Volume	*float*	*read, write*	The volume of this item.
IsBulky	*bool*	*read, write*	Marks this item as a "bulky" item. This is then used by the shop when calculating shipping costs.
BoxQty	*float*	*read, write*	The number of items per box from supplier.
State	*str*	*read, write*	One of "HIDDEN", "ACTIVE".
Unit	*empty*	*read, write*	A Unit reference. The "href" attribute contains the URI for the unit object. Every variant MUST have a Unit, so if this field is not provided on Create the default unit will be assigned.
StockProfile	*empty*	*read, write*	If present, contains an "href" attribute pointing to the stock profile selected for this Variant.
StockInformation	*array*	*read, write*	This is an array of Stock objects. If the shop supports multiple warehouses this element MAY contain multiple Stock object, one for each warehouse, if not exactly one(1) stock object must be provided. Note that the stock information is completely replaced by the information sent here, so any warehouse stock not present is deleted from the shop.
PriceInformation	*array*	*read, write*	This is an array of Prices in different price lists. If the shop supports multiple price lists this element may contain more than one "Item" element. If the shop doesn't support multiple price lists, exactly one such element MUST be present. See below for a description of the price items.

contd...

Name	Type	Mode	Description
CampaignPrice	*float*	*read, write*	The sale price for this item if it is part of a campaign. For some modes of operation a *Campaign* MUST be specified in the Product. If a product is no longer part of a campaign then all variants *CampaignPrice* MUST be removed and the Product *Campaign* MUST also be removed.
Picture	*str*	*read, write*	The image for this variant.
Downloadable	*empty+lang*	*read, write*	If this product is downloadable then this element will be present and have a "href" attribute referencing the downloadable.
SimpleSizes	*str*	*read, write*	A pipe "|"-separated list of simple variations of this product. For example sizes. These do not affect price or SKU in any way. The selected variant is sent as "Size" in the order.
SubscriptionPeriodUnit	*str*	*read, write*	For subscription products this is the time unit for SubscriptionPeriod. Valid values are *DAY*, *MONTH*, *YEAR*.
SubscriptionPeriod	*int*	*read, write*	The subscription period length in "SubscriptionPeriodUnit" units.
ExtraText1..20	*str+lang*	*read, write*	Extra text parameters.
CustomAttributes	*array*	*read, write*	A list of custom attributes and the values they are set to for this Variant. NOTE: Unlike Products variants MUST NOT have more than one value per attribute. This will change in the future so clients MUST be tolerant of multiple values. See section 3.7.4 and 3.11 for details.
ExtraSelector1..6	*int*	*read, write*	Extra select fields, defined in the shop. Can only take on values between 0 and 256, unlike most integers. If set the attribute *textvalue* will contain a textual value corresponding to the numeric index.
Product	*empty*	*read*	This element is used if the client needs to determine the Product-URI for this variant. The "href" attribute contains the URI.
Discountgroup	*str*	*read*	The discount group to which this variant belongs. The "href" field specifies the URI for the discount group. The contents of this element is the discount group tag if there is one. To change the discount group the "href" attribute MUST be modified. The tag is ignored on POST/PUT. To remove the variant from the discount group the client MUST send this element without a "href" attribute.
CreatedTime	*datetime*	*read*	The date and time when this record was created.
ChangedTime	*datetime*	*read*	The date and time when this record was last changed.

contd...

Name	Type	Mode	Description
SyncedTime	*datetime*	*read*	The date and time when this record was last synchronized.

Object description for "PriceInfo" Items

Name	Type	Mode	Description
PriceList	*empty*	*read, write*	If the shop supports multiple price lists this element will be present and contain a "href" attribute referencing the price list for this price item.
Prices	*array*	*read, write*	An array of Price objects. If the shop supports prices for different quantities then this element may contain up to six(6) Price objects. If the shop does NOT support prices for different quantities then exactly one Price object MUST be present. The order of the Price objects is significant. They MUST be ordered by increasing Qty.
RenewalPrice	*float*	*read, write*	The renewal price for this product if it is a subscription product.

Object description for "Price"

Name	Type	Mode	Description
Qty	*int*	*read, write*	The minimum quantity to be purchased.
Price	*float*	*read, write*	The per-item price.

Object description for "Stock"

Name	Type	Mode	Description
Warehouse	*str*	*read, write*	If the shop supports multiple warehouses then this element MUST be sent and indicate the warehouse that this information relates to. If the shop does not support multiple warehouses the client MUST NOT send this element.
StockAvailable	*float*	*read, write*	Amount of items available in stock.
StockComing	*float*	*read, write*	Amount of items on their way.

Object description for "CustomAttributes"

An array of CustomAttribute blocks containing a name and value. Currently only one value is allowed per attribute. The CustomAttribute tag has a "href" attribute that specifies which CustomAttribute this refers to. Changing the "href" tag is the only way to change the attribute.

Please note that this array MUST be sent in its entirety if it is sent at all since exactly those attributes and values listed here will be set on the Variant.

Name	Type	Mode	Description
ID	*str*	*read*	The attribute ID. This field CAN NOT be modified it is there only to simplify for clients that fetch product data.

contd...

Name	Type	Mode	Description
ValueID	*str*	*read*	The attribute value ID. This field CAN NOT be modified it is there only to simplify for clients that fetch product data.
Name	*str*	*read*	The name of this attribute. This field CAN NOT be modified it is there only to simplify for clients that fetch product data.
Value	*str*	*read, write*	The value of this attribute. The value itself is only to aid clients. To change the value the "href" attribute must be changed to another custom-attribute value.

Example XML for complete Variant object

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/product/8/variant/27" variant_id="27">
  <SKU>1.1</SKU>
  <Name lang="sv">Produkt nummer ett A</Name>
  <Name lang="en" primary-language="true">Produkt nummer ett A</Name>
  <ManufacturerSKU xsi:nil="true" />
  <Barcode>57123456789</Barcode>
  <Weight>0</Weight>
  <Volume>0</Volume>
  <IsBulky>false</IsBulky>
  <BoxQty>1</BoxQty>
  <State>ACTIVE</State>
  <Unit href="/__API__/unit/1" xsi:nil="true" />
  <StockProfile href="/__API__/stockprofile/3" xsi:nil="true" />
  <StockInformation>
    <Stock>
      <Warehouse href="/__API__/warehouse/1" xsi:nil="true" />
      <StockAvailable>23</StockAvailable>
      <StockComing>42</StockComing>
    </Stock>
    <Stock>
      <Warehouse href="/__API__/warehouse/3" xsi:nil="true" />
      <StockAvailable>230</StockAvailable>
      <StockComing>420</StockComing>
    </Stock>
  </StockInformation>
  <PriceInformation>
    <Item>
      <Pricelist href="/__API__/pricelist/1" xsi:nil="true" />
      <Prices>
        <Price>
          <Qty>1</Qty>
          <Price>441</Price>
        </Price>
        <Price>
          <Qty>2</Qty>
          <Price>341</Price>
        </Price>
```

```
        <Price>
          <Qty>3</Qty>
          <Price>241</Price>
        </Price>
      </Prices>
      <RenewalPrice xsi:nil="true" />
    </Item>
    <Item>
      <Pricelist href="/__API__/pricelist/2" xsi:nil="true" />
      <Prices>
        <Price>
          <Qty>1</Qty>
          <Price>442</Price>
        </Price>
        <Price>
          <Qty>2</Qty>
          <Price>342</Price>
        </Price>
      </Prices>
      <RenewalPrice xsi:nil="true" />
    </Item>
  </PriceInformation>
  <CampaignPrice xsi:nil="true" />
  <Downloadable href="/__API__/product/downloadable/1" lang="sv"
     xsi:nil="true" />
  <Downloadable href="/__API__/product/downloadable/1" lang="en"
     primary-language="true" xsi:nil="true" />
  <SimpleSizes lang="sv">S|M|L|XL|XXL</SimpleSizes>
  <SimpleSizes lang="en"
     primary-language="true">S|M|L|XL|XXL</SimpleSizes>
  <ExtraText1 lang="sv">Green</ExtraText1>
  <ExtraText1 lang="en" primary-language="true">Green</ExtraText1>
  <ExtraText2 lang="sv">M</ExtraText2>
  <ExtraText2 lang="en" primary-language="true">M</ExtraText2>
  <ExtraText3 lang="sv" xsi:nil="true" />
  <ExtraText3 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText4 lang="sv" xsi:nil="true" />
  <ExtraText4 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText5 lang="sv" xsi:nil="true" />
  <ExtraText5 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText6 lang="sv" xsi:nil="true" />
  <ExtraText6 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText7 lang="sv" xsi:nil="true" />
  <ExtraText7 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText8 lang="sv" xsi:nil="true" />
  <ExtraText8 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText9 lang="sv" xsi:nil="true" />
  <ExtraText9 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText10 lang="sv" xsi:nil="true" />
  <ExtraText10 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText11 lang="sv" xsi:nil="true" />
```

```
  <ExtraText11 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText12 lang="sv" xsi:nil="true" />
  <ExtraText12 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText13 lang="sv" xsi:nil="true" />
  <ExtraText13 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText14 lang="sv" xsi:nil="true" />
  <ExtraText14 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText15 lang="sv" xsi:nil="true" />
  <ExtraText15 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText16 lang="sv" xsi:nil="true" />
  <ExtraText16 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText17 lang="sv" xsi:nil="true" />
  <ExtraText17 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText18 lang="sv" xsi:nil="true" />
  <ExtraText18 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText19 lang="sv" xsi:nil="true" />
  <ExtraText19 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraText20 lang="sv" xsi:nil="true" />
  <ExtraText20 lang="en" primary-language="true" xsi:nil="true" />
  <ExtraSelector1>0</ExtraSelector1>
  <ExtraSelector2>0</ExtraSelector2>
  <ExtraSelector3>0</ExtraSelector3>
  <ExtraSelector4>0</ExtraSelector4>
  <ExtraSelector5>0</ExtraSelector5>
  <ExtraSelector6>0</ExtraSelector6>
  <CustomAttributes>
    <CustomAttribute href="/__API__/customattribute/1">
      <Name>Size</Name>
      <Value href="/__API__/customattribute/1/value/2">M</Value>
    </CustomAttribute>
  </CustomAttributes>
  <Product href="/__API__/product/8" xsi:nil="true" />
  <Discountgroup href="/__API__/discountgroup/1">TEST1</Discountgroup>
  <CreatedTime>2007-11-22T08:57:59Z</CreatedTime>
  <ChangedTime>2014-02-21T09:45:57Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</Variant>
```

Extrafields

The Variant resource supports extrafields, both text fields and selector fields. Extrafields are configurable at `/__API__/product/variant/extrafields`. See section 3.25 for details.

3.7.5 Variant Collection Direct access

This is a shortcut for Variant Collection (see 3.7.3) where the product id is omitted from the path

URI:	`/__API__/variant`
Actions:	Retrieve(GET)
Content-Type:	text/xml
GET Parameters:	page_size, page, synced, view, mark_as_synced, sku, variant_by_sku
POST Parameters:	mark_as_synced

Query parameters

Name	Values	Description
`page_size`	*1..250*	Number of variants per request. Defaults to 100.
`page`	*1..*	The page to view. Defaults to 1.
`synced`	*yes, no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`view`	*short,long*	If specified as "long" then a complete object is sent.
`variant_by_sku`	*string*	Search for a Variant based on the supplied SKU.
`sku`	*string*	Search for a Variant based on the supplied SKU. If the above, variant_by_sku, isn't supplied then this will be used.

This call returns a complete list of variants in the shop. Pagination is active for this list since it might become very large. Description of the meta data for the pagination is be found here 2.15.1

In other aspects this list is equal to variant collection 3.7.3.

For a complete documentation about this please see 3.7.3

3.7.6 Variant Direct access

This is a shortcut for Variant (see 3.7.4) where the product id is omitted from the path

URI:	`/__API__/variant/...`
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
GET Parameters:	mark_as_synced
PUT Parameters:	mark_as_synced, keep_pricelists
Attributes:	href, variant_id

Query parameters

Name	Values	Description
mark_as_synced	*yes,no*	As defined in section 2.11.2.
keep_pricelists	*yes,no*	Normally all prices for all pricelists MUST be sent when updating the price of the variant. The variant will be removed from all other price lists. If this parameter is set to "yes" then the system DOES NOT remove the variant from price lists for which price information was not sent.

For a complete documentation about this please see 3.7.4

3.7.7 Downloadable products

Products MAY consist of a file that is to be downloaded by the customer after purchase is complete. Such products have a `Downloadable` element with a "href" attribute pointing to the downloadable file. This resource describes that file.

The `Downloadable` element MAY be language dependent for those shops that support multiple languages. In that case the element will occur multiple times with the `lang` attribute specifying language. The Root-object (Section 3.1) contains an element specifying whether multiple languages are supported.

To remove a downloadable file reference the client MUST send a `Downloadable` element WITHOUT a `href` attribute. For shops supporting multiple languages a correct `lang` attribute MUST be sent.

3.7.8 Downloadables Collection

URI:	`/__API__/product/downloadable`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<DownloadableList>...</DownloadableList>` is an array of `Downloadable` objects as defined in section 3.7.9.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DownloadableList>
  <Downloadable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/product/downloadable/1">
    <FileName>bugform.pdf</FileName>
    <FileDate>2009-05-06T13:04:46Z</FileDate>
    <FileSize>23362</FileSize>
    <MimeType>application/pdf</MimeType>
    <MaxDownloads xsi:nil="true" />
    <EmailFile>false</EmailFile>
    <AllowDownloadBeforePayment>false</AllowDownloadBeforePayment>
    <FileData href="/__API__/product/downloadable/1/data" xsi:nil="true"
        />
  </Downloadable>
</DownloadableList>
```

3.7.9 Downloadable

URI:	`/__API__/product/downloadable/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`FileName`	*str*	*read, write*	The name of the file as seen by the customer.
`FileDate`	*datetime*	*read*	The last upload date for this file.
`FileSize`	*int*	*read*	The size of this file in bytes.
`MimeType`	*str*	*read, write*	The MIME-Type for this file.
`MaxDownloads`	*int*	*read, write*	Maximum number of times this file may be downloaded per purchase. If NULL or 0 then there is no limit.
`EmailFile`	*bool*	*read, write*	If "true" the file will be attached to the order confirmation email when the order is complete.
`AllowDownloadBeforePayment`	*bool*	*read, write*	Allow this file to be downloaded before payment has been received. The order will still need to be complete tho.
`FileData`	*empty*	*read*	The URI where the file data can be accessed is specified in an "href" attribute to this element. GETting from this URI retrieves the file, and PUTting to it replaces the file data. See 3.7.10 for more info.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<Downloadable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/product/downloadable/1">
  <FileName>bugform.pdf</FileName>
  <FileDate>2009-05-06T13:04:46Z</FileDate>
  <FileSize>23362</FileSize>
  <MimeType>application/pdf</MimeType>
  <MaxDownloads xsi:nil="true" />
  <EmailFile>false</EmailFile>
  <AllowDownloadBeforePayment>false</AllowDownloadBeforePayment>
  <FileData href="/__API__/product/downloadable/1/data" xsi:nil="true" />
</Downloadable>
```

3.7.10 Downloadable file data

This resource represents the actual file data.

URI:	`/__API__/product/downloadable/.../data` (from FileData href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	*Depends on file format*

The file data may be retrieved with a GET request to this URI, which is specified as part of the `FileData` element in the document object.

The data may also be changed by PUTting new data to this URI. It is required that the data is in the same format as the original file. If a format change is needed then the file object needs to be updated with a new MIME-Type first and then new file data uploaded.

3.8 Campaigns

To use campaigns the shop needs to have one of `CampaignPrices` or `AdvancedCampaigns` options. `AdvancedCampaigns` allow the shop owner to set many more parameters, like expiration date and such.

If the shop only has `CampaignPrices` then campaign settings can only be listed, not edited. The products still need to be tagged with a campaign they belong to in order to be shown as campaign items in the shop.

3.8.1 Campaign Collection

URI:	`/__API__/campaign`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<CampaignList>...</CampaignList>` is an array of `Campaign` objects as described below.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CampaignList>
  <Campaign xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/campaign/1">
    <ID>c1</ID>
    <Name lang="sv">Campaign 1</Name>
    <Name lang="en" primary-language="true">Campaign 1</Name>
    <ShortDescription lang="sv">Currently cheap!</ShortDescription>
    <ShortDescription lang="en" primary-language="true">Currently
        cheap!</ShortDescription>
    <Description lang="sv" xsi:nil="true" />
    <Description lang="en" primary-language="true" xsi:nil="true" />
    <StartDate>2007-01-01</StartDate>
    <EndDate>2099-12-31</EndDate>
    <State>ACTIVE</State>
  </Campaign>
  <Campaign xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/campaign/4">
    <ID>C2</ID>
    <Name lang="sv">Campaign 2</Name>
    <Name lang="en" primary-language="true">Campaign 2</Name>
```

```
19      <ShortDescription lang="sv">Waay cheap!</ShortDescription>
20      <ShortDescription lang="en" primary-language="true">Waay
            cheap!</ShortDescription>
21      <Description lang="sv" xsi:nil="true" />
22      <Description lang="en" primary-language="true" xsi:nil="true" />
23      <StartDate>2010-03-25</StartDate>
24      <EndDate>9876-01-23</EndDate>
25      <State>ACTIVE</State>
26    </Campaign>
27  </CampaignList>
```

3.8.2 Campaign

URI:	`/__API__/campaign/...` (from href attribute)
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID of this campaign. The ID values are not used by the shop, but MUST be unique among campaigns.
`Name`	*str+lang*	*read, write*	The name of the campaign. MAY be displayed in the shop depending on shop layout.
`ShortDescription`	*str+lang*	*read, write*	A short description of the campaign. MAY be (and usually is) displayed in the shop.
`Description`	*str+lang*	*read, write*	A description of the campaign. MAY be displayed in the shop.
`StartDate`	*date*	*read, write*	The start date for this campaign.
`EndDate`	*date*	*read, write*	The end date for this campaign.
`State`	*str*	*read*	The current state for this campaign. Possible values are ACTIVE or INACTIVE.

Example XML

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>
2  <Campaign xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/campaign/1">
3    <ID>c1</ID>
4    <Name lang="sv">Campaign 1</Name>
5    <Name lang="en" primary-language="true">Campaign 1</Name>
6    <ShortDescription lang="sv">Currently cheap!</ShortDescription>
7    <ShortDescription lang="en" primary-language="true">Currently
         cheap!</ShortDescription>
8    <Description lang="sv" xsi:nil="true" />
9    <Description lang="en" primary-language="true" xsi:nil="true" />
10   <StartDate>2007-01-01</StartDate>
```

```
11    <EndDate>2099-12-31</EndDate>
12    <State>ACTIVE</State>
13 </Campaign>
```

3.9 Pictures

Pictures are represented by two distinct resources: The XML picture metadata and the image data.

Picture metadata contains information about the type, size, filename and such parameters. It also contains a reference to an URI that represents the image data.

To create a new picture one needs to first create a new metadata object and then upload the image data to the URI specified in that object. Similarly, when changing an image one might need to update the metadata first in some cases and then upload new image data.

Pictures MUST be linked to from other objects!
Pictures that are not linked to MAY be removed by the system.
The client MUST NOT add pictures that it does not intend to use.

Pictures MAY be removed by the system when the resource that linked to them is removed!

The client MUST NOT link pictures to more than one object as this may cause problems when the objects are removed via the shop admin pages.

3.9.1 Picture Collection

URI:	`/__API__/picture`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<PictureList>...</PictureList>` is an array of complete `Picture` objects as defined below.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PictureList>
  <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/picture/8" picture_id="8">
    <Filename>logoDefault.gif</Filename>
    <Width>300</Width>
    <Height>67</Height>
    <MimeType>image/gif</MimeType>
    <ImageData href="/__API__/picture/8/data" xsi:nil="true" />
    <PictureURL>/PICTURE/logoDefault.gif</PictureURL>
    <CreatedTime xsi:nil="true" />
    <ChangedTime xsi:nil="true" />
  </Picture>
  <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/picture/20" picture_id="20">
    <Filename>ingen_bild4.jpg</Filename>
    <Width>499</Width>
    <Height>375</Height>
    <MimeType>image/jpeg</MimeType>
    <ImageData href="/__API__/picture/20/data" xsi:nil="true" />
    <PictureURL>/PICTURE/ingen_bild4.jpg</PictureURL>
    <CreatedTime xsi:nil="true" />
    <ChangedTime xsi:nil="true" />
  </Picture>
  <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/picture/60" picture_id="60">
    <Filename>Untitled-5.jpg</Filename>
    <Width>499</Width>
    <Height>375</Height>
    <MimeType>image/jpeg</MimeType>
    <ImageData href="/__API__/picture/60/data" xsi:nil="true" />
    <PictureURL>/PICTURE/Untitled-5.JPG</PictureURL>
    <CreatedTime xsi:nil="true" />
    <ChangedTime xsi:nil="true" />
  </Picture>
  <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/picture/69" picture_id="69">
    <Filename>sas-6gb-s.cmyk.jpg</Filename>
    <Width>133</Width>
    <Height>46</Height>
    <MimeType>image/jpeg</MimeType>
    <ImageData href="/__API__/picture/69/data" xsi:nil="true" />
```

```
39      <PictureURL>/PICTURE/sas-6gb-scmyk.jpg</PictureURL>
40      <CreatedTime xsi:nil="true" />
41      <ChangedTime xsi:nil="true" />
42    </Picture>
43    <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/70" picture_id="70">
44      <Filename>logoDefault.gif</Filename>
45      <Width>300</Width>
46      <Height>67</Height>
47      <MimeType>image/gif</MimeType>
48      <ImageData href="/__API__/picture/70/data" xsi:nil="true" />
49      <PictureURL>/PICTURE/logodefault.gif</PictureURL>
50      <CreatedTime xsi:nil="true" />
51      <ChangedTime xsi:nil="true" />
52    </Picture>
53    <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/71" picture_id="71">
54      <Filename>logoDefault.gif</Filename>
55      <Width>300</Width>
56      <Height>67</Height>
57      <MimeType>image/gif</MimeType>
58      <ImageData href="/__API__/picture/71/data" xsi:nil="true" />
59      <PictureURL>/PICTURE/logoDefault_bSlFwd.gif</PictureURL>
60      <CreatedTime xsi:nil="true" />
61      <ChangedTime xsi:nil="true" />
62    </Picture>
63    <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/72" picture_id="72">
64      <Filename>bannerDefault.jpg</Filename>
65      <Width>910</Width>
66      <Height>110</Height>
67      <MimeType>image/jpeg</MimeType>
68      <ImageData href="/__API__/picture/72/data" xsi:nil="true" />
69      <PictureURL>/PICTURE/bannerdefault.jpg</PictureURL>
70      <CreatedTime xsi:nil="true" />
71      <ChangedTime xsi:nil="true" />
72    </Picture>
73    <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/73" picture_id="73">
74      <Filename>bannerDefault.jpg</Filename>
75      <Width>910</Width>
76      <Height>110</Height>
77      <MimeType>image/jpeg</MimeType>
78      <ImageData href="/__API__/picture/73/data" xsi:nil="true" />
79      <PictureURL>/PICTURE/bannerDefault_GyXmWq.jpg</PictureURL>
80      <CreatedTime xsi:nil="true" />
81      <ChangedTime xsi:nil="true" />
82    </Picture>
83    <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/74" picture_id="74">
84      <Filename>ky.gif</Filename>
```

```
85      <Width>400</Width>
86      <Height>223</Height>
87      <MimeType>image/gif</MimeType>
88      <ImageData href="/__API__/picture/74/data" xsi:nil="true" />
89      <PictureURL>/PICTURE/ky.gif</PictureURL>
90      <CreatedTime xsi:nil="true" />
91      <ChangedTime xsi:nil="true" />
92    </Picture>
93    <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/75" picture_id="75">
94      <Filename>ky.gif</Filename>
95      <Width>400</Width>
96      <Height>223</Height>
97      <MimeType>image/gif</MimeType>
98      <ImageData href="/__API__/picture/75/data" xsi:nil="true" />
99      <PictureURL>/PICTURE/ky_eTWxRz.GIF</PictureURL>
100     <CreatedTime xsi:nil="true" />
101     <ChangedTime xsi:nil="true" />
102   </Picture>
103   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/76" picture_id="76">
104     <Filename>Untitled-5.jpg</Filename>
105     <Width>499</Width>
106     <Height>375</Height>
107     <MimeType>image/jpeg</MimeType>
108     <ImageData href="/__API__/picture/76/data" xsi:nil="true" />
109     <PictureURL>/PICTURE/untitled-5_oeeguc.jpg</PictureURL>
110     <CreatedTime xsi:nil="true" />
111     <ChangedTime xsi:nil="true" />
112   </Picture>
113   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/77" picture_id="77">
114     <Filename>Untitled-5.jpg</Filename>
115     <Width>499</Width>
116     <Height>375</Height>
117     <MimeType>image/jpeg</MimeType>
118     <ImageData href="/__API__/picture/77/data" xsi:nil="true" />
119     <PictureURL>/PICTURE/Untitled-5_oEegUC_LvVveS.JPG</PictureURL>
120     <CreatedTime xsi:nil="true" />
121     <ChangedTime xsi:nil="true" />
122   </Picture>
123   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/78" picture_id="78">
124     <Filename>Untitled-5.jpg</Filename>
125     <Width>499</Width>
126     <Height>375</Height>
127     <MimeType>image/jpeg</MimeType>
128     <ImageData href="/__API__/picture/78/data" xsi:nil="true" />
129     <PictureURL>/PICTURE/untitled-5_vgkypd.jpg</PictureURL>
130     <CreatedTime xsi:nil="true" />
131     <ChangedTime xsi:nil="true" />
```

```
132   </Picture>
133   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/79" picture_id="79">
134     <Filename>Untitled-5.jpg</Filename>
135     <Width>499</Width>
136     <Height>375</Height>
137     <MimeType>image/jpeg</MimeType>
138     <ImageData href="/__API__/picture/79/data" xsi:nil="true" />
139     <PictureURL>/PICTURE/Untitled-5_VGkypD_owSaGm.JPG</PictureURL>
140     <CreatedTime xsi:nil="true" />
141     <ChangedTime xsi:nil="true" />
142   </Picture>
143   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/80" picture_id="80">
144     <Filename>Untitled-5.jpg</Filename>
145     <Width>499</Width>
146     <Height>375</Height>
147     <MimeType>image/jpeg</MimeType>
148     <ImageData href="/__API__/picture/80/data" xsi:nil="true" />
149     <PictureURL>/PICTURE/untitled-5_iwpbbt.jpg</PictureURL>
150     <CreatedTime xsi:nil="true" />
151     <ChangedTime xsi:nil="true" />
152   </Picture>
153   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/81" picture_id="81">
154     <Filename>Untitled-5.jpg</Filename>
155     <Width>499</Width>
156     <Height>375</Height>
157     <MimeType>image/jpeg</MimeType>
158     <ImageData href="/__API__/picture/81/data" xsi:nil="true" />
159     <PictureURL>/PICTURE/Untitled-5_iWpbBT_AmFbhP.JPG</PictureURL>
160     <CreatedTime xsi:nil="true" />
161     <ChangedTime xsi:nil="true" />
162   </Picture>
163   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/82" picture_id="82">
164     <Filename>Untitled-5.jpg</Filename>
165     <Width>499</Width>
166     <Height>375</Height>
167     <MimeType>image/jpeg</MimeType>
168     <ImageData href="/__API__/picture/82/data" xsi:nil="true" />
169     <PictureURL>/PICTURE/untitled-5_ewajys.jpg</PictureURL>
170     <CreatedTime xsi:nil="true" />
171     <ChangedTime xsi:nil="true" />
172   </Picture>
173   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/83" picture_id="83">
174     <Filename>Untitled-5.jpg</Filename>
175     <Width>499</Width>
176     <Height>375</Height>
177     <MimeType>image/jpeg</MimeType>
```

```
178     <ImageData href="/__API__/picture/83/data" xsi:nil="true" />
179     <PictureURL>/PICTURE/Untitled-5_ewajyS_meSFGP.JPG</PictureURL>
180     <CreatedTime xsi:nil="true" />
181     <ChangedTime xsi:nil="true" />
182   </Picture>
183   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/84" picture_id="84">
184     <Filename>Untitled-5.jpg</Filename>
185     <Width>499</Width>
186     <Height>375</Height>
187     <MimeType>image/jpeg</MimeType>
188     <ImageData href="/__API__/picture/84/data" xsi:nil="true" />
189     <PictureURL>/PICTURE/untitled-5_riiewi.jpg</PictureURL>
190     <CreatedTime xsi:nil="true" />
191     <ChangedTime xsi:nil="true" />
192   </Picture>
193   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/85" picture_id="85">
194     <Filename>Untitled-5.jpg</Filename>
195     <Width>499</Width>
196     <Height>375</Height>
197     <MimeType>image/jpeg</MimeType>
198     <ImageData href="/__API__/picture/85/data" xsi:nil="true" />
199     <PictureURL>/PICTURE/Untitled-5_riiEwI_kRbLbH.JPG</PictureURL>
200     <CreatedTime xsi:nil="true" />
201     <ChangedTime xsi:nil="true" />
202   </Picture>
203   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/86" picture_id="86">
204     <Filename>Untitled-5.jpg</Filename>
205     <Width>499</Width>
206     <Height>375</Height>
207     <MimeType>image/jpeg</MimeType>
208     <ImageData href="/__API__/picture/86/data" xsi:nil="true" />
209     <PictureURL>/PICTURE/untitled-5_qnujso.jpg</PictureURL>
210     <CreatedTime xsi:nil="true" />
211     <ChangedTime xsi:nil="true" />
212   </Picture>
213   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/87" picture_id="87">
214     <Filename>Untitled-5.jpg</Filename>
215     <Width>499</Width>
216     <Height>375</Height>
217     <MimeType>image/jpeg</MimeType>
218     <ImageData href="/__API__/picture/87/data" xsi:nil="true" />
219     <PictureURL>/PICTURE/Untitled-5_QnUjSo_ZtpUky.JPG</PictureURL>
220     <CreatedTime xsi:nil="true" />
221     <ChangedTime xsi:nil="true" />
222   </Picture>
223   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          href="/__API__/picture/88" picture_id="88">
```

```
224     <Filename>Untitled-5.jpg</Filename>
225     <Width>499</Width>
226     <Height>375</Height>
227     <MimeType>image/jpeg</MimeType>
228     <ImageData href="/__API__/picture/88/data" xsi:nil="true" />
229     <PictureURL>/PICTURE/untitled-5_uojkcs.jpg</PictureURL>
230     <CreatedTime xsi:nil="true" />
231     <ChangedTime xsi:nil="true" />
232   </Picture>
233   <Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/picture/89" picture_id="89">
234     <Filename>Untitled-5.jpg</Filename>
235     <Width>499</Width>
236     <Height>375</Height>
237     <MimeType>image/jpeg</MimeType>
238     <ImageData href="/__API__/picture/89/data" xsi:nil="true" />
239     <PictureURL>/PICTURE/Untitled-5_uojKcs_tvslyn.JPG</PictureURL>
240     <CreatedTime xsi:nil="true" />
241     <ChangedTime xsi:nil="true" />
242   </Picture>
243 </PictureList>
```

3.9.2 Picture

URI:	`/__API__/picture/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`Filename`	*str*	*read, write*	The filename of the picture. Not necessarily the same as the filename on the server.
`Width`	*int*	*read*	The width in pixels.
`Height`	*int*	*read*	The height in pixels.
`MimeType`	*str*	*read, write*	The image type. Supported types are: “image/jpeg”, “image/png”, “image/gif”
`RemoteLink`	*str*	*read, write*	The remote bucket link.
`ImageData`	*empty*	*read*	The URI where the image data can be accessed is specified in an “href” attribute to this element. GETting from this URI retrieves the image, and PUTtting to it replaces the image data. See 3.9.3 for more info.
`PictureURL`	*empty*	*read*	The shop URL path for this image. Since a shop may have multiple domains only the URL path is sent. Add “http:// SHOP DOMAIN /” for a complete URL
`CreatedTime`	*datetime*	*read*	The date and time when this record was created.
`ChangedTime`	*datetime*	*read*	The date and time when this record was last changed.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Picture xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/picture/8" picture_id="8">
  <Filename>logoDefault.gif</Filename>
  <Width>300</Width>
  <Height>67</Height>
  <MimeType>image/gif</MimeType>
  <ImageData href="/__API__/picture/8/data" xsi:nil="true" />
  <PictureURL>/PICTURE/logoDefault.gif</PictureURL>
  <CreatedTime xsi:nil="true" />
  <ChangedTime xsi:nil="true" />
</Picture>
```

3.9.3 Picture Data

This resource represents the actual image data.

URI:	`/__API__/picture/.../data` (from ImageData href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	*Depends on image format*

The image data may be retrieved with a GET request to this URI, which is specified as part of the `ImageData` element in the picture object.

The data may also be changed by PUTting new image data to this URI. It is required that the new image is of the same type as the current image. If a format change is needed then the image object needs to be updated with a new MIME-Type first and then new image data uploaded.

3.9.4 Example Picture

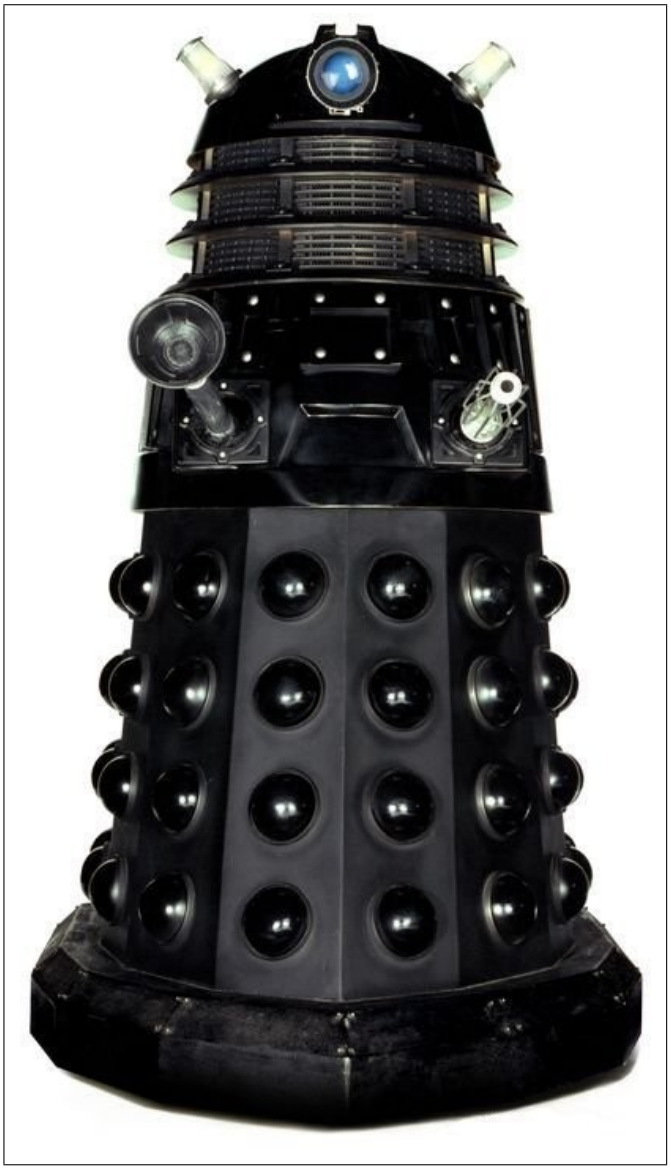

Figure 3.1: Product picture.

3.9.5 Picture Bucket Data

URI:	`/__API__/picture/.../bucket_data` (from ImageData href attribute)
Actions:	Retrieve(GET), Update(PUT)

The image bucket data may be retrieved with a GET request to this URI.

Name	Type	Mode	Description
`Kind`	*str*	*read, write*	Type of data. (storage#object)
`ID`	*str*	*read, write*	Id for file bucket.
`SelfLink`	*str*	*read, write*	Link to bucket.
`MediaLink`	*str*	*read, write*	Link to media.
`Name`	*str*	*read, write*	Filename.
`Bucket`	*str*	*read, write*	Name of bucket.
`Generation`	*str*	*read, write*	File generation.
`MetaGeneration`	*str*	*read, write*	Meta generation for file.
`ContentType`	*str*	*read, write*	Meta content type.
`StorageClass`	*str*	*read, write*	Storage class.
`Size`	*str*	*read, write*	File size.
`MD5Hash`	*str*	*read, write*	File MD5 hash sum.
`CRC32C`	*str*	*read, write*	CRC32 for file.
`Etag`	*str*	*read, write*	Remote Etag.
`CreatedTime`	*str*	*read, write*	Record created time.
`ChangedTime`	*str*	*read, write*	Record changed time.
`StorageChangedTime`	*str*	*read, write*	Storage change time.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<BucketData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/picture/123/bucket_data">
     <Kind>storage#object</Kind>
     <ID>unique-bucket-id/images/filename.jpg/1234567890123456</ID>
     <SelfLink>https://www.googleapis.com/storage/v1/b/unique-bucket-id/o/images%2Ffi
     <MediaLink>https://www.googleapis.com/download/storage/v1/b/unique-bucket-id/o/i
     <Name>images/filename.jpg</Name>
     <Bucket>unique-bucket-id</Bucket>
     <Generation>1234567890123456</Generation>
     <MetaGeneration>1</MetaGeneration>
     <ContentType>image/jpg</ContentType>
     <StorageClass>REGIONAL</StorageClass>
     <Size>2159</Size>
     <MD5Hash>Z9zXakAfvFfc2s5A2Yj98A==</MD5Hash>
     <CRC32C>ph2sqQ==</CRC32C>
     <Etag>vObZg44k2OsCQAG=</Etag>
     <CreatedTime>2021-02-09T12:16:06.939Z</CreatedTime>
     <ChangedTime>2021-02-09T12:16:06.939Z</ChangedTime>
     <StorageChangedTime>2021-02-09T12:16:06.939Z</StorageChangedTime>
```

```
20 </BucketData>
```

3.10 Product Categories

3.10.1 Category Collection

URI:	`/__API__/category`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, view, mark_as_synced, filter
POST Parameters:	mark_as_synced, generate_seo_url

Query parameters

Name	Values	Description
`synced`	*yes,no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`view`	*short,long*	If specified as "long" then an almost complete object is sent for each category. Fields left out of a long listing are fields referring to some external entities.
`generate_seo_url`	*0/1*	If specified and set to 1, the system will use the "Name" field to generate an SEO-URL. The "Name" element MUST be present. The "SEOURL" element MUST NOT be present.

Object description

The returned object `<CategoryList>...</CategoryList>` is an array of abbreviated `Category` objects:

Name	Type	Mode	Description
`ID`	*str*	*read, filter*	The ID of this category. Must be unique.
`Name`	*str*	*read*	The name of this category.
`ParentCategory`	*empty*	*read*	If present, contains an "href" attribute pointing to the parent category.
`CreatedTime, filter`	*datetime*	*read*	The date and time when this category was created.
`ChangedTime, filter`	*datetime*	*read*	The date and time when this record was last changed.
`SyncedTime, filter`	*datetime*	*read*	The date and time when this record was last synchronized.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<CategoryList>
  <Category xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      category_id="5" href="/__API__/category/5">
    <ID></ID>
    <Name lang="sv">Varugrupp ett</Name>
    <Name lang="en" primary-language="true">Varugrupp ett</Name>
    <CreatedTime>2007-11-22T06:53:09Z</CreatedTime>
    <ChangedTime>2009-06-01T14:10:33Z</ChangedTime>
    <SyncedTime>2010-02-03T18:06:30Z</SyncedTime>
  </Category>
  <Category xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      category_id="6" href="/__API__/category/6">
    <ID></ID>
    <Name lang="sv">Varugrupp två</Name>
    <Name lang="en" primary-language="true">Varugrupp två</Name>
    <CreatedTime>2007-11-22T07:29:08Z</CreatedTime>
    <ChangedTime>2009-06-01T14:10:33Z</ChangedTime>
    <SyncedTime>2010-02-03T18:06:30Z</SyncedTime>
  </Category>
  <Category xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      category_id="7" href="/__API__/category/7">
    <ID></ID>
    <Name lang="sv">Varugrupp tre</Name>
    <Name lang="en" primary-language="true">Varugrupp tre</Name>
    <CreatedTime>2007-11-22T07:29:19Z</CreatedTime>
    <ChangedTime>2009-06-01T14:10:33Z</ChangedTime>
    <SyncedTime>2010-02-03T18:06:30Z</SyncedTime>
  </Category>
  <Category xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      category_id="8" href="/__API__/category/8">
    <ID></ID>
    <Name lang="sv">Varugrupp fyra</Name>
    <Name lang="en" primary-language="true">Varugrupp fyra</Name>
    <CreatedTime>2007-11-22T07:29:32Z</CreatedTime>
    <ChangedTime>2009-06-01T14:10:33Z</ChangedTime>
    <SyncedTime>2010-02-03T18:06:30Z</SyncedTime>
  </Category>
  <Category xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      category_id="9" href="/__API__/category/9">
    <ID></ID>
    <Name lang="sv">Undergrupp ett</Name>
    <Name lang="en" primary-language="true">Undergrupp ett</Name>
    <ParentCategory href="/__API__/category/5" xsi:nil="true" />
    <CreatedTime>2007-11-22T11:40:03Z</CreatedTime>
    <ChangedTime>2009-06-01T14:10:33Z</ChangedTime>
    <SyncedTime>2010-02-03T18:06:30Z</SyncedTime>
  </Category>
</CategoryList>
```

3.10.2 Category

URI:	`/__API__/category/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
GET Parameters:	mark_as_synced
PUT Parameters:	mark_as_synced, generate_seo_url
Attributes:	href, category_id

Query parameters

Name	Values	Description
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`generate_seo_url`	*0/1*	If specified and set to 1, the system will use the "Name" field to generate an SEO-URL. The "Name" element MUST be present. The "SEOURL" element MUST NOT be present.

Object description

The `Category` element contains a "href" attribute specifying the complete URI for this category. It also contains a "category_id" attribute that is the category_id parameter used in the shop front end. This might be used by the API user to create links that link into a specific category in the shop.

The elements in this object are described below.

Name	Type	Mode	Description
`ID`	*str*	*read, write, filter*	The ID of this category. Must be unique.
`Name`	*str+lang*	*read, write*	The category name.
`Description`	*str*	*read, write*	The category description displayed to the customer. This field is displayed to the customer on the web and MAY contain HTML as long as the HTML is properly coded as an XML string.
`ParentCategory`	*empty*	*read, write*	If present, contains an "href" attribute pointing to the parent category. If this element is sent without the "href" attribute the category will be removed from its parent category and become a top-level category.
`Picture`	*empty*	*read, write*	A Picture reference for the main pictur for this category. The "href" attribute contains the URI for the picture object. Sending the Picture element without a "href" tag removes the picture.

contd...

Name	Type	Mode	Description
IconPicture	*empty*	*read, write*	A Picture for the category icon. The "href" attribute contains the URI for the picture object. Sending the Picture element without a "href" tag removes the picture.
Brand	*empty*	*read, write*	A Brand reference. The "href" attribute contains the URI for the brand object. Sending the Brand element without a "href" tag removes the brand.
ExtraText1..8	*str+lang*	*read, write, filter*	Extra text parameters.
ExtraSelector1..4	*int*	*read, write, filter*	Extra select fields, defined in the shop. Can only take on values between 0 and 256, unlike most integers. If set the attribute *textvalue* will contain a textual value corresponding to the numeric index.
State	*str+lang*	*read, write, filter*	One of "HIDDEN", "ACTIVE", "SEARCHABLE"
SEOTitle	*str+lang*	*read, write*	For SEO: The title displayed on the category page in the shop.
SEODescription	*str+lang*	*read, write*	For SEO: The meta-description displayed on the category page in the shop.
SEOKeywords	*str+lang*	*read, write*	For SEO: The meta-keywords displayed on the category page in the shop.
SEOURL	*str+lang*	*read, write*	The URL component for SEO-friendly URLs. These MUST be unique between categories.
SubCategories	*str*	*read*	An array of Category references listing all direct SubCategories of this category. This does NOT list the entire subtree.
Locks	*array*	*read, write*	A list of locks assigned to this category. See section 3.10.2 and 3.26 for details.
AntiLocks	*array*	*read, write*	A list of anti locks assigned to this category. See section 3.10.2 and 3.26 for details.
CreatedTime	*datetime*	*read, filter*	The date and time when this category was created.
ChangedTime	*datetime*	*read, filter*	The date and time when this record was last changed.
SyncedTime	*datetime*	*read, filter*	The date and time when this record was last synchronized.

Object description for "Locks"

An array of Locks blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
Index	*int*	*read, write*	The index for the lock. 0..63
Value	*bool*	*read, write*	If lock is active then true, otherwise false

Object description for "AntiLocks"

An array of AntiLocks blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered anti locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
Index	*int*	*read, write*	The index for the anti lock. 0..63
Value	*bool*	*read, write*	If anti lock is active then true, otherwise false

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Category xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    category_id="5" href="/__API__/category/5">
  <ID></ID>
  <Name lang="sv">Varugrupp ett</Name>
  <Name lang="en" primary-language="true">Varugrupp ett</Name>
  <Description lang="sv">&lt;h3&gt;Varugrupp ett&lt;/h3&gt;Lorem ipsum
      dolor sit amet, consectetuer adipiscing elit. Suspendisse enim
      nunc, elementum sit amet, tincidunt et, gravida non, justo. Vivamus
      pede. Vestibulum augue. Aliquam risus. Suspendisse nec est a lectus
      fermentum elementum. Pellentesque sollicitudin feugiat purus. Donec
      ut mauris. Curabitur molestie congue magna. In tempor elit id
      nulla. Phasellus sed tortor. Sed id urna in lectus bibendum
      venenatis. Aliquam ut mi.</Description>
  <Description lang="en" primary-language="true">&lt;h3&gt;Varugrupp
      ett&lt;/h3&gt;Lorem ipsum dolor sit amet, consectetuer adipiscing
      elit. Suspendisse enim nunc, elementum sit amet, tincidunt et,
      gravida non, justo. Vivamus pede. Vestibulum augue. Aliquam risus.
      Suspendisse nec est a lectus fermentum elementum. Pellentesque
      sollicitudin feugiat purus. Donec ut mauris. Curabitur molestie
      congue magna. In tempor elit id nulla. Phasellus sed tortor. Sed id
      urna in lectus bibendum venenatis. Aliquam ut mi.</Description>
  <Brand href="/__API__/brand/1" xsi:nil="true" />
  <ExtraText1 lang="sv"></ExtraText1>
  <ExtraText1 lang="en" primary-language="true"></ExtraText1>
  <ExtraText2 lang="sv"></ExtraText2>
  <ExtraText2 lang="en" primary-language="true"></ExtraText2>
  <ExtraText3 lang="sv"></ExtraText3>
  <ExtraText3 lang="en" primary-language="true"></ExtraText3>
  <ExtraText4 lang="sv"></ExtraText4>
  <ExtraText4 lang="en" primary-language="true"></ExtraText4>
  <ExtraText5 lang="sv"></ExtraText5>
  <ExtraText5 lang="en" primary-language="true"></ExtraText5>
  <ExtraText6 lang="sv"></ExtraText6>
  <ExtraText6 lang="en" primary-language="true"></ExtraText6>
  <ExtraText7 lang="sv"></ExtraText7>
  <ExtraText7 lang="en" primary-language="true"></ExtraText7>
  <ExtraText8 lang="sv"></ExtraText8>
  <ExtraText8 lang="en" primary-language="true"></ExtraText8>
  <ExtraSelector1>0</ExtraSelector1>
  <ExtraSelector2>0</ExtraSelector2>
  <ExtraSelector3>0</ExtraSelector3>
  <ExtraSelector4>0</ExtraSelector4>
  <State lang="sv">ACTIVE</State>
```

```
30   <State lang="en" primary-language="true">ACTIVE</State>
31   <SEOTitle lang="sv" xsi:nil="true" />
32   <SEOTitle lang="en" primary-language="true" xsi:nil="true" />
33   <SEODescription lang="sv" xsi:nil="true" />
34   <SEODescription lang="en" primary-language="true" xsi:nil="true" />
35   <SEOKeywords lang="sv" xsi:nil="true" />
36   <SEOKeywords lang="en" primary-language="true" xsi:nil="true" />
37   <SEOURL lang="sv">varugrupp-ett</SEOURL>
38   <SEOURL lang="en" primary-language="true">varugrupp-ett</SEOURL>
39   <Locks>
40       <Lock>
41           <Index>0</Index>
42           <Value>false</Value>
43       </Lock>
44       <Lock>
45           <Index>7</Index>
46           <Value>false</Value>
47       </Lock>
48   </Locks>
49   <AntiLocks>
50       <AntiLock>
51           <Index>0</Index>
52           <Value>false</Value>
53       </AntiLock>
54       <AntiLock>
55           <Index>7</Index>
56           <Value>false</Value>
57       </AntiLock>
58   </AntiLocks>
59   <CreatedTime>2007-11-22T06:53:09Z</CreatedTime>
60   <ChangedTime>2009-06-01T14:10:33Z</ChangedTime>
61   <SyncedTime>2010-02-03T18:06:30Z</SyncedTime>
62   <SubCategories>
63     <Category href="/__API__/category/9">
64       <ID></ID>
65       <Name lang="sv">Undergrupp ett</Name>
66       <Name lang="en" primary-language="true">Undergrupp ett</Name>
67       <ParentCategory href="/__API__/category/5" xsi:nil="true" />
68       <CreatedTime>2007-11-22T11:40:03Z</CreatedTime>
69       <ChangedTime>2009-06-01T14:10:33Z</ChangedTime>
70       <SyncedTime>2010-02-03T18:06:30Z</SyncedTime>
71     </Category>
72   </SubCategories>
73 </Category>
```

Extrafields

The Category resource supports extrafields, both text fields and selector fields. Extrafields are configurable at `/__API__/category/extrafields`. See section 3.25 for details.

3.11 Custom Attributes

Custom attributes are attributes defined within the shop that have a given name and a list of values.

The attributes are selectable by the shop administrator for Variants (Section 3.7.4) and in the future other objects.

The values within a custom attribute are ordered and the API provides functionality for specifying this ordering if the client wishes to do so.

3.11.1 Custom Attributes Collection

URI:	`/__API__/customattribute`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Name	Values	Description
`view`	*short,long*	If specified as "long" then complete "Attribute" object is sent for each attribute. Useful for downloading entire attribute set with values.

Object description

This resource represents a list of abbreviated custom-attribute objects. The objects contain a "href" attribute that specifies the URI where a specific custom attribute may be accessed.

Name	Type	Mode	Description
`ID`	*str*	*read*	The attribute ID
`Name`	*str*	*read*	The name of this attribute in the primary language for this API user.
`Active`	*bool*	*read*	True if this attribute is active in the shop.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CustomAttributeList>
  <CustomAttribute xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customattribute/1">
    <ID xsi:nil="true" />
    <Name>Size</Name>
    <Active>true</Active>
  </CustomAttribute>
  <CustomAttribute xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customattribute/2">
```

```
    <ID xsi:nil="true" />
    <Name>Color</Name>
    <Active>true</Active>
  </CustomAttribute>
</CustomAttributeList>
```

3.11.2 Custom Attribute

URI:	`/__API__/customattribute/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read,write*	The attribute ID
`Name`	*str+lang*	*read, write*	The name of this attribute for all languages the shop supports (if this API user is configured for multiple language support)
`Active`	*bool*	*read, write*	True if this attribute is active in the shop.
`Values`	*array*	*read, write*	An ordered array of value references used to determine the order in which values are listed in the shop. The client wishing to change the order may do so by sending a *Values* block with *AttributeValue* elements for all values in the desired order. When reading these items contain all attribute value data aswell. However modifying attribute values is not allowed here. Use the attribute value resource for that.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CustomAttribute xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/customattribute/1">
  <ID xsi:nil="true" />
  <Name lang="sv">Storlek</Name>
  <Name lang="en" primary-language="true">Size</Name>
  <Active>true</Active>
  <Values href="/__API__/customattribute/1/value/">
    <AttributeValue href="/__API__/customattribute/1/value/1">
      <ID xsi:nil="true" />
      <Value lang="sv">S</Value>
      <Value lang="en" primary-language="true">S</Value>
      <Active>true</Active>
    </AttributeValue>
    <AttributeValue href="/__API__/customattribute/1/value/2">
      <ID xsi:nil="true" />
      <Value lang="sv">M</Value>
      <Value lang="en" primary-language="true">M</Value>
      <Active>true</Active>
```

```
    </AttributeValue>
    <AttributeValue href="/__API__/customattribute/1/value/3">
      <ID xsi:nil="true" />
      <Value lang="sv">L</Value>
      <Value lang="en" primary-language="true">L</Value>
      <Active>true</Active>
    </AttributeValue>
  </Values>
</CustomAttribute>
```

3.11.3 Custom Attribute Values Collection

URI:	`/__API__/customattribute/.../value` (from href attribute)
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

This resource represents a list of full attribute value objects. The objects contain a “href” attribute that specifies the URI where a specific value may be accessed.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ValueList>
  <AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customattribute/1/value/1">
    <ID xsi:nil="true" />
    <Value lang="sv">S</Value>
    <Value lang="en" primary-language="true">S</Value>
    <Active>true</Active>
  </AttributeValue>
  <AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customattribute/1/value/2">
    <ID xsi:nil="true" />
    <Value lang="sv">M</Value>
    <Value lang="en" primary-language="true">M</Value>
    <Active>true</Active>
  </AttributeValue>
  <AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customattribute/1/value/3">
    <ID xsi:nil="true" />
    <Value lang="sv">L</Value>
    <Value lang="en" primary-language="true">L</Value>
    <Active>true</Active>
  </AttributeValue>
</ValueList>
```

3.11.4 Custom Attributes Value

Object description

URI:	`/__API__/customattribute/.../value/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Name	Type	Mode	Description
`ID`	*str*	*read*	The attribute value ID
`Value`	*str+lang*	*read, write*	The name of this attribute value in all languages.
`Active`	*bool*	*read, write*	True if this value is active in the shop.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/customattribute/1/value/1">
  <ID xsi:nil="true" />
  <Value lang="sv">S</Value>
  <Value lang="en" primary-language="true">S</Value>
  <Active>true</Active>
</AttributeValue>
```

3.12 Customer Categories

The customer categories are used to set defaults for customers that belong to this category. These settings can be overridden by settings in the customer objects.

3.12.1 Customer Category Collection

URI:	`/__API__/customercategory`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<CustomercategoryList>...</CustomercategoryList>` is an array of `Customercategory` objects as defined in section 3.12.2.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CustomercategoryList>
  <Customercategory
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customercategory/1">
    <ID>GOOD</ID>
    <Name>Very Good Customers</Name>
    <Currency xsi:nil="true" />
    <DiscountPercent>0</DiscountPercent>
  </Customercategory>
  <Customercategory
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/customercategory/2">
    <ID>BAD</ID>
    <Name>Bad customers</Name>
    <Currency xsi:nil="true" />
    <DiscountPercent>0</DiscountPercent>
  </Customercategory>
</CustomercategoryList>
```

3.12.2 Customer Category

URI:	*Customer Category specific.*
URI:	`/__API__/customercategory/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID of this customer category. Must be unique.
`Name`	*str*	*read, write*	The customer category name.
`Currency`	*str*	*read, write*	The default currency according to ISO-4217.
`DiscountPercent`	*str*	*read, write*	Default discount in percent (%)
`Pricelist`	*empty*	*read, write*	The default price list. A “href” attribute contains the URI for the referenced price list. The tag itself is empty. Sending the tag without the “href” attribute removes the pricelist setting for this category.
`Warehouse`	*empty*	*read, write*	The default price list. A “href” attribute contains the URI for the referenced price list. The tag itself is empty. Sending the tag without the “href” attribute removes the pricelist setting for this category.
`Keys`	*array*	*read, write*	A list of keys assigned to this customer category. See section 3.12.2 and 3.26 for details.

Object description for “Keys”

An array of `Keys` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the key. 0..63
`Value`	*bool*	*read, write*	If key is active then true, otherwise false

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customercategory xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/customercategory/1">
  <ID>GOOD</ID>
  <Name>Very Good Customers</Name>
  <Currency xsi:nil="true" />
  <DiscountPercent>0</DiscountPercent>
  <Pricelist href="/__API__/pricelist/2" xsi:nil="true" />
  <Warehouse href="/__API__/warehouse/1" xsi:nil="true" />
  <Keys>
    <Key>
      <Index>0</Index>
      <Value>false</Value>
    </Key>
    <Key>
      <Index>7</Index>
      <Value>false</Value>
    </Key>
  </Keys>
</Customercategory>
```

3.13 Pricelist

Price lists are only available if the shop supports extended prices information.

3.13.1 Pricelist Collection

URI:	`/__API__/pricelist`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description for GET

The returned object `<PricelistList>...</PricelistList>` is an array of `Pricelist` [3] objects.

The `<Pricelist>` tag contains an `href="<uri>"` attribute that specifies the URI identifying this pricelist.

Object description for POST

When creating a new Pricelist with POST the complete `Pricelist` object specified in section 3.13.2 is posted.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PricelistList>
  <Pricelist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/pricelist/1">
    <ID>PRIS1</ID>
    <Name>Customer price</Name>
    <Currency>SEK</Currency>
    <IsCustomerSelectable>false</IsCustomerSelectable>
  </Pricelist>
  <Pricelist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/pricelist/2">
    <ID>PRIS2</ID>
    <Name>Testlist</Name>
    <Currency>SEK</Currency>
    <IsCustomerSelectable>false</IsCustomerSelectable>
  </Pricelist>
```

[3] See section 3.13.2

```
</PricelistList>
```

3.13.2 Pricelist

URI:	*Pricelist specific.*
URI:	`/__API__/pricelist` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	An ID for this list. Must be unique or empty.
`Name`	*str+lang*	*read, write*	List name.
`Currency`	*str*	*read, write*	3-letter alphabetic currency code according to ISO-4217 as CAPITALS.
`IsCustomerSelectable`	*bool*	*read, write*	Marks if a customer may select this pricelist or if it is set by the shop owner.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Pricelist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/pricelist/1">
  <ID>PRIS1</ID>
  <Name lang="sv">Kundpris</Name>
  <Name lang="en" primary-language="true">Customer price</Name>
  <Currency>SEK</Currency>
  <IsCustomerSelectable>false</IsCustomerSelectable>
</Pricelist>
```

3.14 Brands / Manufacturers

3.14.1 Brand Collection

URI:	`/__API__/brand`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
POST-parameters:	generate_seo_url

Query parameters

Name	Values	Description
`generate_seo_url`	*0/1*	If specified and set to 1, the system will use the "Name" field to generate an SEO-URL. The "Name" element MUST be present. The "SEOURL" element MUST NOT be present.

Object description

The returned object `<BrandList>...</BrandList>` is an array of `Brand` objects as defined in section 3.14.2.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <BrandList>
3   <Brand xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       brand_id="1" href="/__API__/brand/1">
4     <ID>8</ID>
5     <Name lang="sv">API test brand</Name>
6     <Name lang="en" primary-language="true">API test brand</Name>
7     <Description lang="sv">A brand for testing the api</Description>
8     <Description lang="en" primary-language="true">A brand for testing
         the api</Description>
9   </Brand>
10 </BrandList>
```

3.14.2 Brand

URI:	`/__API__/brand` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
PUT-parameters:	generate_seo_url
Attributes:	href, brand_id

Object description

The `Brand` element contains a "href" attribute specifying the complete URI for this brand. It also contains a "brand_id" attribute that is the brand_id parameter used in the shop front end. This might be used by the API user to create links that link into a specific brand in the shop.

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID of this brand. Must be unique.
`Name`	*str+lang*	*read, write*	The brand name.
`Description`	*str+lang*	*read, write*	The brand description shown in the shop. This field is displayed to the customer on the web and MAY contain HTML as long as the HTML is properly coded as an XML string.
`WebLink`	*str+lang*	*read, write*	A link to the manufacturers homepage.
`SortOrder`	*int*	*read, write*	Sorting order field. Lower is better.
`Picture`	*empty*	*read, write*	A Picture reference. The "href" attribute contains the URI for the picture object. Sending the Picture element without a "href" tag removes the picture from the brand.
`Documents`	*array*	*read, write*	An array of "Document" elements containing a "href" attribute identifying the document.
`ExtraText1..6`	*str+lang*	*read, write*	Six (6) extra text fields for extra information that the shop might want.
`SEOTitle`	*str+lang*	*read, write*	For SEO: The title displayed on the category page in the shop.
`SEODescription`	*str+lang*	*read, write*	For SEO: The meta-description displayed on the category page in the shop.
`SEOKeywords`	*str*	*read, write*	For SEO: The meta-keywords displayed on the category page in the shop.
`SEOURL`	*str+lang*	*read, write*	The URL component for SEO-friendly URLs. These MUST be unique between brands.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Brand xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     brand_id="1" href="/__API__/brand/1">
3   <ID>8</ID>
4   <Name lang="sv">API test brand</Name>
5   <Name lang="en" primary-language="true">API test brand</Name>
6   <Description lang="sv">A brand for testing the api</Description>
```

```
7   <Description lang="en" primary-language="true">A brand for testing the
       api</Description>
8   <WebLink lang="sv">http://www.example.com</WebLink>
9   <WebLink lang="en"
       primary-language="true">http://www.example.com</WebLink>
10  <SortOrder xsi:nil="true" />
11  <Documents>
12    <Document href="/__API__/document/1" />
13  </Documents>
14  <ExtraText1 lang="sv" xsi:nil="true" />
15  <ExtraText1 lang="en" primary-language="true" xsi:nil="true" />
16  <ExtraText2 lang="sv" xsi:nil="true" />
17  <ExtraText2 lang="en" primary-language="true" xsi:nil="true" />
18  <ExtraText3 lang="sv" xsi:nil="true" />
19  <ExtraText3 lang="en" primary-language="true" xsi:nil="true" />
20  <ExtraText4 lang="sv" xsi:nil="true" />
21  <ExtraText4 lang="en" primary-language="true" xsi:nil="true" />
22  <ExtraText5 lang="sv" xsi:nil="true" />
23  <ExtraText5 lang="en" primary-language="true" xsi:nil="true" />
24  <ExtraText6 lang="sv" xsi:nil="true" />
25  <ExtraText6 lang="en" primary-language="true" xsi:nil="true" />
26  <SEOTitle lang="sv" xsi:nil="true" />
27  <SEOTitle lang="en" primary-language="true" xsi:nil="true" />
28  <SEODescription lang="sv" xsi:nil="true" />
29  <SEODescription lang="en" primary-language="true" xsi:nil="true" />
30  <SEOKeywords lang="sv" xsi:nil="true" />
31  <SEOKeywords lang="en" primary-language="true" xsi:nil="true" />
32  <SEOURL lang="sv">api-test-brand-2</SEOURL>
33  <SEOURL lang="en" primary-language="true">api-test-brand-2</SEOURL>
34 </Brand>
```

Extrafields

The Brand resource supports extrafields, both text fields and selector fields. Extrafields are configurable at `/__API__/brand/extrafields`. See section 3.25 for details.

3.15 Units

3.15.1 Unit Collection

URI:	/__API__/unit
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object <UnitList>...</UnitList> is an array of Unit objects as defined in section 3.15.2.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<UnitList>
  <Unit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/unit/1">
    <ID xsi:nil="true" />
    <Unit lang="sv">st</Unit>
    <Unit lang="en" primary-language="true">st</Unit>
    <NumberOfDecimals>0</NumberOfDecimals>
  </Unit>
  <Unit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/unit/2">
    <ID xsi:nil="true" />
    <Unit lang="sv">m</Unit>
    <Unit lang="en" primary-language="true">m</Unit>
    <NumberOfDecimals>1</NumberOfDecimals>
  </Unit>
  <Unit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/unit/3">
    <ID xsi:nil="true" />
    <Unit lang="sv">kg</Unit>
    <Unit lang="en" primary-language="true">kg</Unit>
    <NumberOfDecimals>1</NumberOfDecimals>
  </Unit>
  <Unit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/unit/4">
    <ID xsi:nil="true" />
    <Unit lang="sv">sats</Unit>
    <Unit lang="en" primary-language="true">sats</Unit>
    <NumberOfDecimals>0</NumberOfDecimals>
  </Unit>
```

```
27   <Unit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/unit/5">
28     <ID xsi:nil="true" />
29     <Unit lang="sv">förp</Unit>
30     <Unit lang="en" primary-language="true">förp</Unit>
31     <NumberOfDecimals>0</NumberOfDecimals>
32   </Unit>
33   <Unit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/unit/6">
34     <ID xsi:nil="true" />
35     <Unit lang="sv">förp</Unit>
36     <Unit lang="en" primary-language="true">förp</Unit>
37     <NumberOfDecimals>0</NumberOfDecimals>
38   </Unit>
39 </UnitList>
```

3.15.2 Unit

URI:	`/__API__/unit` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID of this unit. Must be unique.
`Unit`	*str+lang*	*read, write*	The unit...
`NumberOfDecimals`	*int*	*read, write*	The number of decimals for quantities of this unit.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Unit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     href="/__API__/unit/1">
3   <ID xsi:nil="true" />
4   <Unit lang="sv">st</Unit>
5   <Unit lang="en" primary-language="true">st</Unit>
6   <NumberOfDecimals>0</NumberOfDecimals>
7 </Unit>
```

3.16 Stockprofiles

3.16.1 Stockprofile Collection

URI:	/__API__/stockprofile
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object <StockprofileList>...</StockprofileList> is an array of Stockprofile objects as defined in section 3.16.2.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StockprofileList>
  <Stockprofile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/stockprofile/1">
    <ID>STOCK1</ID>
    <Name>Standard</Name>
    <Mode>NONE</Mode>
    <DenyOrderIfNotInStock>false</DenyOrderIfNotInStock>
    <DisplayCode></DisplayCode>
  </Stockprofile>
  <Stockprofile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/stockprofile/3">
    <ID>STOCK2</ID>
    <Name>Other</Name>
    <Mode>NONE</Mode>
    <DenyOrderIfNotInStock>false</DenyOrderIfNotInStock>
    <DisplayCode></DisplayCode>
  </Stockprofile>
</StockprofileList>
```

3.16.2 Stockprofile

URI:	/__API__/stockprofile (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
ID	*str*	*read, write*	The ID of this stockprofile. Must be unique.
Name	*str+lang*	*read, write*	The stockprofile name.
Mode	*bool*	*read, write*	Stock control mode. Allowed values are *NONE* — Don't do anything, *DECREASE* — Decrease stock on purchase.
DenyOrderIfNotInStock	*bool*	*read, write*	Deny placing an order if item is not in stock.
DisplayCode	*str+lang*	*read, write*	Script code used to display stock information.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Stockprofile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/stockprofile/1">
  <ID>STOCK1</ID>
  <Name lang="sv">Standard</Name>
  <Name lang="en" primary-language="true">Standard</Name>
  <Mode>NONE</Mode>
  <DenyOrderIfNotInStock>false</DenyOrderIfNotInStock>
  <DisplayCode lang="sv"></DisplayCode>
  <DisplayCode lang="en" primary-language="true"></DisplayCode>
</Stockprofile>
```

3.17 Warehouses

3.17.1 Warehouse Collection

URI:	`/__API__/warehouse`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<WarehouseList>...</WarehouseList>` is an array of `Warehouse` objects as defined in section 3.17.2.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<WarehouseList>
  <Warehouse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/warehouse/1">
    <ID>1</ID>
    <Name>Lager 1</Name>
  </Warehouse>
  <Warehouse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/warehouse/3">
    <ID>2</ID>
    <Name>Lager 2</Name>
  </Warehouse>
</WarehouseList>
```

3.17.2 Warehouse

URI:	`/__API__/warehouse/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID of this warehouse. Must be unique.
`Name`	*str*	*read, write*	The warehouse name.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Warehouse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/warehouse/1">
3   <ID>1</ID>
4   <Name>Lager 1</Name>
5 </Warehouse>
```

3.18 Discountgroups

3.18.1 Discountgroup Collection

URI:	`/__API__/discountgroup`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<DiscountgroupList>...</DiscountgroupList>` is an array of `Discountgroup` objects as defined in section 3.18.2.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DiscountgroupList>
  <Discountgroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/discountgroup/1">
    <ID>TEST1</ID>
    <Name>Test 1</Name>
    <Description>First test group</Description>
  </Discountgroup>
  <Discountgroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/discountgroup/2">
    <ID>TEST2</ID>
    <Name>Test 2</Name>
    <Description xsi:nil="true" />
  </Discountgroup>
</DiscountgroupList>
```

3.18.2 Discountgroup

URI:	`/__API__/discountgroup/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID of this discountgroup. Must be unique.
`Name`	*str*	*read, write*	The discountgroup name.

contd...

Name	Type	Mode	Description
Description	*str*	*read, write*	The discountgroup description.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Discountgroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/discountgroup/1">
  <ID>TEST1</ID>
  <Name>Test 1</Name>
  <Description>First test group</Description>
</Discountgroup>
```

3.19 Documents

Like pictures, Documents are represented by two distinct resources: The XML document metadata and the file data. These entities are accessed using different but related URIs.

Document metadata contains information about the type, size, filename and such parameters. It also contains a reference to an URI that represents the file data.

To create a new document one needs to first create a new metadata object and then upload the file data to the URI specified in that object. Similarly, when changing a document one might need to update the metadata first in some cases and then upload new file data.

3.19.1 Document Collection

URI:	`/__API__/document`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<DocumentList>...</DocumentList>` is an array of `Document` objects as defined in section 3.19.2.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <DocumentList>
3   <Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/document/1">
4     <ID>XMAS1</ID>
5     <Name>Christmas Card</Name>
6     <Description>A christmas card.</Description>
7     <FileName>ft041201.gif</FileName>
8     <FileSize>12292</FileSize>
9     <MimeType>image/gif</MimeType>
10    <DocumentData href="/__API__/document/1/data" xsi:nil="true" />
11  </Document>
12 </DocumentList>
```

3.19.2 Document

URI:	`/__API__/document/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The document ID.
`Name`	*str*	*read, write*	The document name.
`Description`	*str*	*read, write*	The document description.
`FileName`	*str*	*read, write*	The filename of the document. This is exactly the filename used on the server, so different documents MUST have different filenames.
`FileSize`	*int*	*read*	The size of the file.
`MimeType`	*str*	*read, write*	The document type.
`Locks`	*array*	*read, write*	A list of locks assigned to this document. See section 3.19.2 and 3.26 for details.
`AntiLocks`	*array*	*read, write*	A list of anti locks assigned to this document. See section 3.19.2 and 3.26 for details.
`DocumentData`	*empty*	*read*	The URI where the file data can be accessed is specified in an "href" attribute to this element. GETting from this URI retrieves the file, and PUTting to it replaces the file data. See 3.19.3 for more info.

Object description for "Locks"

An array of `Locks` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the lock. 0..63
`Value`	*bool*	*read, write*	If lock is active then true, otherwise false

Object description for "AntiLocks"

An array of `AntiLocks` blocks containing an index and value (true or false). If omitted the previous value will be kept. Only registered anti locks will be used. See section 3.26 for managing locks.

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the anti lock. 0..63
`Value`	*bool*	*read, write*	If anti lock is active then true, otherwise false

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/document/1">
3   <ID>XMAS1</ID>
4   <Name>Christmas Card</Name>
```

```
  <Description>A christmas card.</Description>
  <FileName>ft041201.gif</FileName>
  <FileSize>12292</FileSize>
  <MimeType>image/gif</MimeType>
  <DocumentData href="/__API__/document/1/data" xsi:nil="true" />
</Document>
```

3.19.3 Document Data

This resource represents the actual file data.

URI:	`/__API__/document/.../data` (from href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	*Depends on file format*

The file data may be retrieved with a GET request to this URI, which is specified as part of the `DocumentData` element in the document object.

The data may also be changed by PUTting new data to this URI. It is required that the data is in the same format as the original file. If a format change is needed then the file object needs to be updated with a new MIME-Type first and then new file data uploaded.

3.20 Discountcode

3.20.1 Discountcode Collection

URI:	`/__API__/discountcode`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<DiscountcodeList>...</DiscountcodeList>` is an array of `Discountcode` [4] objects.

The `<Discountcode>` tag contains an `href="<uri>"` attribute that specifies the URI identifying this discountcode.

When creating a new Discountcode with POST the complete `Discountcode` object specified in section 3.20.2 is posted.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DiscountcodeList>
  <Discountcode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/discountcode/1">
    <Code>apiuser</Code>
    <Name>API tester campaign.</Name>
    <State>ACTIVE</State>
    <Mode>OMNI</Mode>
    <StartDate>2009-04-14T22:00:00Z</StartDate>
    <EndDate>2009-07-24T22:00:00Z</EndDate>
    <DiscountPercent>10</DiscountPercent>
    <DiscountSum xsi:nil="true" />
    <MinimumOrderSum xsi:nil="true" />
    <MaximumOrderSum xsi:nil="true" />
    <LimitToOrderSum>false</LimitToOrderSum>
    <FixedDiscountIncludesTax>false</FixedDiscountIncludesTax>
    <IncludePaymentAndShippingCosts>false</IncludePaymentAndShippingCosts>
    <ValidForAllProducts>false</ValidForAllProducts>
    <DisallowForProductsInCampaign>false</DisallowForProductsInCampaign>
    <CannotBeCombinedWithOtherDiscountCodes>false</CannotBeCombinedWithOtherDiscount
    <CannotBeCombinedWithNForM>false</CannotBeCombinedWithNForM>
    <Variant href="/__API__/product/17/variant/30" xsi:nil="true" />
    <OrderList href="/__API__/discountcode/1/orders" xsi:nil="true" />
```

[4] See section 3.20.2

```
23   </Discountcode>
24 </DiscountcodeList>
```

3.20.2 Discountcode

URI:	`/__API__/discountcode/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`Code`	*str*	*read, write*	The discount code to be entered by the customer.
`Name`	*str*	*read, write*	Campaign name.
`State`	*str*	*read, write*	One of "ACTIVE" or "INACTIVE"
`Mode`	*str*	*read, write*	The code mode. See mode description below.
`StartDate`	*datetime*	*read, write*	The date and time at which this campaign starts. The code is not valid before this date.
`EndDate`	*datetime*	*read, write*	The date and time at which this campaign ends. The code is not valid after this date.
`DiscountPercent`	*float*	*read, write*	A discount in percent that is to be applied to the order. Can be combined with "DiscountSum".
`DiscountSum`	*float*	*read, write*	A discount to be applied to the order. Can be combined with "DiscountPercent".
`MinimumOrderSum`	*float*	*read, write*	This code only applies to orders of at least this total.
`MaximumOrderSum`	*float*	*read, write*	This code only applies to orders of at most this total.
`LimitToOrderSum`	*bool*	*read, write*	The discount can never exceed the order sum total. If this is false and the "DiscountSum" is larger than the order total then the customer order total will become negative.

contd...

Name	Type	Mode	Description
FixedDiscountIncludesTax	*bool*	*read, write*	The fixed discount sum includes tax and is to be calculated on the price with tax applied. This is used for gift certificates mostly.
IncludePaymentAndShippingCosts	*bool*	*read, write*	If true then payment and shipping costs will be considered when calculating the discounts.
ValidForAllProducts	*bool*	*read, write*	If true then this discount code will consider all products when calculating the discounts. If false only specific products are considered. The logic is different for percentage and fixed sum codes. Consult the main shop documentation for details.
DisallowForProductsInCampaign	*bool*	*read, write*	If true then this code will NOT be applied to products that belong to a campaign.
CannotBeCombinedWithOtherDiscountCodes	*bool*	*read, write*	If true then this code cannot be combined with other discount codes.
CannotBeCombinedWithNForM	*bool*	*read, write*	If true then this code cannot be used if an "Buy N pay for M" campaign is active in the cart.
Variant	*empty*	*read, write*	A reference to the variant to be used for this code. Each code MUST have a product and variant associated with it, to be placed in the cart. The variant MUST be created before creating the code. Many codes MAY share variants, for example if different codes belong to the same campaign.
OrderList	*empty*	*read*	A reference to a collection containing the orders where this code has been used.
AllowBrands	*array*	*read,write*	An array of *Brand* elements whose "href" attribute specifies an allowed Brand.
AllowCategories	*array*	*read,write*	An array of *Category* elements whose "href" attribute specifies an allowed Category.

contd...

Name	Type	Mode	Description
AllowDiscountGroups	*array*	*read,write*	An array of *DiscountGroup* elements whose "href" attribute specifies an allowed Discount group.
AllowProducts	*array*	*read,write*	An array of *Product* elements whose "href" attribute specifies an allowed product.
AllowVariants	*array*	*read,write*	An array of *Variant* elements whose "href" attribute specifies an allowed variant.

Modes

Discount codes support a couple different modes that determine how the code can be used. The following modes are supported:

OMNI This code is "omnipresent" and may be used many times by customers during its validity period.

ONCEPERCUST This code can only be used once per customer. But different customers may use the same code.

ONCEONLY This is a one-time-only code.

FIXEDSUMCOUNTDOWN Codes with this mode can be used multiple times, and for each time they are used the "DiscountSum" is decreased by the order amount. This mode may be used to implement gift certificates.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Discountcode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/discountcode/1">
  <Code>apiuser</Code>
  <Name>API tester campaign.</Name>
  <State>ACTIVE</State>
  <Mode>OMNI</Mode>
  <StartDate>2009-04-14T22:00:00Z</StartDate>
  <EndDate>2009-07-24T22:00:00Z</EndDate>
  <DiscountPercent>10</DiscountPercent>
  <DiscountSum xsi:nil="true" />
  <MinimumOrderSum xsi:nil="true" />
  <MaximumOrderSum xsi:nil="true" />
  <LimitToOrderSum>false</LimitToOrderSum>
  <FixedDiscountIncludesTax>false</FixedDiscountIncludesTax>
  <IncludePaymentAndShippingCosts>false</IncludePaymentAndShippingCosts>
  <ValidForAllProducts>false</ValidForAllProducts>
  <DisallowForProductsInCampaign>false</DisallowForProductsInCampaign>
```

```
18   <CannotBeCombinedWithOtherDiscountCodes>false</CannotBeCombinedWithOtherDiscountCo
19   <CannotBeCombinedWithNForM>false</CannotBeCombinedWithNForM>
20   <Variant href="/__API__/product/17/variant/30" xsi:nil="true" />
21   <OrderList href="/__API__/discountcode/1/orders" xsi:nil="true" />
22   <AllowBrands>
23     <Brand href="/__API__/brand/1" />
24   </AllowBrands>
25   <AllowCategories>
26     <Category href="/__API__/category/5" />
27   </AllowCategories>
28   <AllowDiscountGroups>
29     <DiscountGroup href="/__API__/discountgroup/1" />
30   </AllowDiscountGroups>
31   <AllowProducts>
32     <Product href="/__API__/product/10" />
33   </AllowProducts>
34   <AllowVariants>
35     <Variant href="/__API__/product/11/variant/15" />
36   </AllowVariants>
37 </Discountcode>
```

3.20.3 Orders where codes were used

URI:	`/__API__/discountcode/.../orders` (from OrderList href attribute)
Actions:	Retrieve(GET)
Content-Type:	text/xml
GET Parameters:	synced, mark_as_synced, filter

This collection contains a list of orders where the customer has used this particular discount code. The object syntax is the same as for the order collection described in section 3.3.1.

This resource supports the synchronization procedure described in section 2.11.

Query parameters

Name	Values	Description
`synced`	*yes,no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`filter`		See the Order section 3.3.1. for details about which fields can be filtered.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <OrderList>
```

```
3   <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/order/2">
4     <OrderNo>2</OrderNo>
5     <ErpOrderNo xsi:nil="true" />
6     <State>ACCEPTED</State>
7     <PaymentState>UNPAID</PaymentState>
8     <PaymentIsCaptured>false</PaymentIsCaptured>
9     <CaptureTime xsi:nil="true" />
10    <PaymentIsCancelled>false</PaymentIsCancelled>
11    <CancelTime xsi:nil="true" />
12    <CreatedTime>2009-04-15T19:51:39Z</CreatedTime>
13    <ChangedTime>2009-04-15T19:51:39Z</ChangedTime>
14    <SyncedTime>2009-04-17T06:57:41Z</SyncedTime>
15  </Order>
16 </OrderList>
```

3.21 Invoice

3.21.1 Invoice Collection

URI:	/__API__/invoice
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, mark_as_synced, filter

Object description

The returned object <InvoiceList>...</InvoiceList> is an array of Invoice [5] objects.

The <Invoice> tag contains an href="<uri>" attribute that specifies the URI identifying this invoice.

When creating a new Invoice with POST the complete Invoice object specified in section 3.21.2 is posted.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <InvoiceList>
3   <Invoice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/invoice/1">
4     <InvoiceNo>10000</InvoiceNo>
5     <Customer href="/__API__/customer/1">1</Customer>
6     <Source href="/__API__/order/2">ORDER</Source>
7     <Type>INVOICE</Type>
8     <State>PENDING</State>
9     <CreatedTime xsi:nil="true" />
10    <ChangedTime>2014-02-18T09:17:31Z</ChangedTime>
11    <SyncedTime xsi:nil="true" />
12  </Invoice>
13  <Invoice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       href="/__API__/invoice/2">
14    <InvoiceNo>10001</InvoiceNo>
15    <Customer href="/__API__/customer/1">1</Customer>
16    <Source href="/__API__/invoice/1">INVOICE</Source>
17    <Type>CREDIT</Type>
18    <State>PENDING</State>
19    <CreatedTime xsi:nil="true" />
20    <ChangedTime>2014-02-18T09:17:31Z</ChangedTime>
21    <SyncedTime xsi:nil="true" />
```

[5] See section 3.21.2

```
22   </Invoice>
23 </InvoiceList>
```

3.21.2 Invoice

URI:	`/__API__/invoice/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`InvoiceNo`	*str*	*read, filter*	The invoice number. This is read only and is created by the system when an invoice is created.
`Customer`	*str*	*read, write*	A reference to the customer that this invoice is for. The "href" attribute references the customer object itself.
`Source`	*str*	*read, write, filter*	The source of this invoice. Type is specified within, and for some types a "href" attribute refers to the source object. See below for supported sources.
`PaymentState`	*str*	*read, write, filter*	The payment state of this invoice. See section 3.3.2.
`Type`	*str*	*read, write, filter*	The invoice type. See below for types.
`State`	*str*	*read, write, filter*	The invoice state. See below for states.
`Comment`	*str*	*read, write*	Invoice comment.
`OurVatNo`	*str*	*read, write*	The shop VAT number.
`TheirVatNo`	*str*	*read, write, filter*	The customer VAT number.
`Currency`	*str*	*read*	The currency for this invoice. The default currency is used if this field is empty.
`CurrencyConversionRate`	*float*	*read*	The rate used by the shop when converting this invoice to the base currency.
`Sum`	*float*	*read*	The total invoice sum, including tax.
`Tax`	*float*	*read*	The total invoice tax.
`InvoiceDate`	*date*	*read, write, filter*	Date when invoice was created. This is not the same as the "CreatedTime" entry below. This refers to the date that will be written on the invoice.
`DueDate`	*date*	*read, write, filter*	Date when this invoice is due.
`CustomerName`	*str*	*read*	The customer name.
`CustomerCompany`	*str*	*read*	The customer company.
`CustomerAddress1`	*str*	*read*	The customer address, line 1.
`CustomerAddress2`	*str*	*read*	The customer address, line 2.
`CustomerPostalCode`	*str*	*read*	The customer postal/zip code.
`CustomerCity`	*str*	*read*	The customer city.
`CustomerCountry`	*ccode*	*read*	The customer country code.
`Language`	*lang*	*read*	The language this invoice should be in.

contd...

Name	Type	Mode	Description
InvoiceItems	*array*	*read, write*	An array of Item objects representing the items purchased. This object is described below.
CreatedTime	*datetime*	*read, filter*	The date and time when this category was created.
ChangedTime	*datetime*	*read, filter*	The date and time when this record was last changed.
SyncedTime	*datetime*	*read, filter*	The date and time when this record was last synchronized.

Object description for "InvoiceItem"

Name	Type	Mode	Description
LineNo	*int*	*read, write*	The line number.
Type	*str*	*read, write*	Type of item. Allowed types are "NORMAL", "DISCOUNT", "COMMENT".
SKU	*str*	*read, write*	The SKU for this item, as written on invoice.
Descr	*str*	*read, write*	The description for this item, as written on invoice.
Unit	*str*	*read, write*	The unit, as written on invoice
Price	*float*	*read, write*	Per-unit price.
Tax	*float*	*read, write*	Per-unit tax.
TaxPercent	*float*	*read, write(editable)*	The per-unit tax in percent for this item. If Price and Tax are sent but TaxPercent IS NOT sent when updating an order the system will estimate the tax percentage.
Qty	*float*	*read, write*	Quantity.

Sources

The following are the sources that can be set on an invoice, and the requirements pertaining to each source type.

ORDER The source of this invoice is an order. The "href" attribute MUST be present and reference the order in question.

INVOICE The source of this invoice is another invoice. The "href" attribute MUST be present and reference the invoice in question.

MULTIPLEORDERS This invoice is a compound invoice containing items from multiple orders. The "href" attribute MUST NOT be present.

OTHER The source of this invoice is something else. The "href" attribute MUST NOT be present.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Invoice xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/invoice/1">
```

```
<InvoiceNo>10000</InvoiceNo>
<Customer href="/__API__/customer/1">1</Customer>
<Source href="/__API__/order/2">ORDER</Source>
<PaymentState>UNPAID</PaymentState>
<Type>INVOICE</Type>
<State>PENDING</State>
<Comment>Faktura för order: 2</Comment>
<Currency xsi:nil="true" />
<CurrencyConversionRate>1</CurrencyConversionRate>
<OurVatNo xsi:nil="true" />
<TheirVatNo xsi:nil="true" />
<Sum>614.5</Sum>
<Tax>122.9</Tax>
<InvoiceDate>2009-10-21</InvoiceDate>
<DueDate>2009-11-20</DueDate>
<CustomerName>Tess T. Persson</CustomerName>
<CustomerCompany xsi:nil="true" />
<CustomerAddress1>Testgatan 42</CustomerAddress1>
<CustomerAddress2 xsi:nil="true" />
<CustomerPostalCode>123 45</CustomerPostalCode>
<CustomerCity>Testcity</CustomerCity>
<CustomerCountry>SE</CustomerCountry>
<Language>SV</Language>
<InvoiceItems>
  <Item>
    <LineNo>1</LineNo>
    <Type>NORMAL</Type>
    <SKU>3</SKU>
    <Descr>Produkt nummer tre</Descr>
    <Unit>st</Unit>
    <Price>192</Price>
    <Tax>48</Tax>
    <TaxPercent>25</TaxPercent>
    <Qty>2</Qty>
  </Item>
  <Item>
    <LineNo>2</LineNo>
    <Type>NORMAL</Type>
    <SKU>apiuser</SKU>
    <Descr>API tester campaign.</Descr>
    <Unit xsi:nil="true" />
    <Price>-38.4</Price>
    <Tax>-9.6</Tax>
    <TaxPercent>25</TaxPercent>
    <Qty>1</Qty>
  </Item>
  <Item>
    <LineNo>3</LineNo>
    <Type>NORMAL</Type>
    <SKU>Exp. avg</SKU>
    <Descr>Faktura</Descr>
```

```
54        <Unit xsi:nil="true" />
55        <Price>50</Price>
56        <Tax>12.5</Tax>
57        <TaxPercent>25</TaxPercent>
58        <Qty>1</Qty>
59      </Item>
60      <Item>
61        <LineNo>4</LineNo>
62        <Type>NORMAL</Type>
63        <SKU>Frakt</SKU>
64        <Descr>Företagspaket</Descr>
65        <Unit xsi:nil="true" />
66        <Price>96</Price>
67        <Tax>24</Tax>
68        <TaxPercent>25</TaxPercent>
69        <Qty>1</Qty>
70      </Item>
71    </InvoiceItems>
72    <CreatedTime xsi:nil="true" />
73    <ChangedTime>2014-02-18T09:17:31Z</ChangedTime>
74    <SyncedTime xsi:nil="true" />
75  </Invoice>
```

3.22 Subscriptions

3.22.1 Subscription Collection

URI:	`/__API__/subscription`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, view, mark_as_synced, filter
POST Parameters:	mark_as_synced

Query parameters

Name	Values	Description
`synced`	*yes,no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.
`view`	*short,long*	If specified as "long" then an almost complete "Subscription" object is sent for each subscription. Fields left out of a long listing are fields referring to some external entities.

Object description

Name	Type	Mode	Description
`Customer`	*str*	*read*	A reference to the customer that has made this order. The "href" attribute references the customer object itself. The element contains the customer number (CustomerNo) for the customer as an extra aid.
`State`	*str*	*read, write*	One of "ACTIVE", "RENEWAL_NOTICE_SENT", "INACTIVE", "CANCELLED"
`ExpireDate`	*date*	*read, write, filter*	The date when this subscription expires if it isn't renewed.
`NextRenewalDate`	*date*	*read, write, filter*	The date when the next renewal is scheduled.
`AnnualWorth`	*int*	*read*	The total annual income from this subscription.
`Currency`	*str*	*read, write*	The currency of the subscription.
`SubscriptionItems`	*empty*	*read*	A collection of all items within this subscription. The "href" attribute contains the collection URI. See below for details.
`CreatedTime`	*datetime*	*read, filter*	The date and time when this record was created.
`ChangedTime`	*datetime*	*read, filter*	The date and time when this record was last changed.
`SyncedTime`	*datetime*	*read, filter*	The date and time when this record was last synchronized.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SubscriptionList>
  <Subscription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/subscription/1">
    <Customer href="/__API__/customer/1" xsi:nil="true" />
    <State>ACTIVE</State>
    <ExpireDate>2012-01-02</ExpireDate>
    <NextRenewalDate>2012-01-02</NextRenewalDate>
    <AnnualWorth>5504</AnnualWorth>
    <Currency>SEK</Currency>
    <SubscriptionItems href="/__API__/subscription/1/item"
       xsi:nil="true" />
    <CreatedTime>2010-01-25T15:04:54Z</CreatedTime>
    <ChangedTime>2012-09-05T14:02:21Z</ChangedTime>
    <SyncedTime xsi:nil="true" />
  </Subscription>
</SubscriptionList>
```

3.22.2 Subscription

URI:	*As specified in href attribute.*
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml
GET Parameters:	mark_as_synced
PUT Parameters:	mark_as_synced

Query parameters

Name	Values	Description
mark_as_synced	*yes,no*	As defined in section 2.11.2.

Object description

Name	Type	Mode	Description
Customer	*str*	*read*	A reference to the customer that has made this order. The "href" attribute references the customer object itself. The element contains the customer number (CustomerNo) for the customer as an extra aid.
State	*str*	*read, write*	One of "ACTIVE", "RENEWAL_NOTICE_SENT", "INACTIVE", "CANCELLED"

contd...

Name	Type	Mode	Description
PaymentMethod	*empty*	*read, write*	The "href" attribute specifies the URI of the payment method to be used for automatic renewal of this subscription
ShippingMethod	*empty*	*read, write*	The "href" attribute specifies the URI of the shipping method to be used for automatic renewal of this subscription
AutomaticRenewals	*bool*	*read, write*	True if the system is to try to automatically renew this subscription before it expires. If set to false, the API user is expected to handle this.
AnnualWorth	*int*	*read*	The total annual income from this subscription.
Currency	*str*	*read, write*	The currency of the subscription.
FreeEveryNRenewals	*int*	*read, write*	If this element is not NULL then it specifies how often the customer is to receive a free renewal of this subscription. Useful for giving customers "every 4th month free" kind of deals.
SubscriptionStartedAt	*datetime*	*read, write, filter*	The date and time when this subscription was set up.
ExpireDate	*date*	*read, write, filter*	The date when this subscription expires if it isn't renewed.
ReorderDate	*date*	*read, write, filter*	If the shop is to handle renewals (AutomaticRenewals = true) then this is the date when the shop will place a renewal order for this subscription. This is usually a few weeks or a month before expiration.
DaysFromReorderToRenewal	*int*	*read, write*	This is used to calculate the "ReorderDate" for the next renewal when a subscription is renewed. The system takes the expiration date and subtracts this number of days from it.
LastRenewalOrderDate	*date*	*read, write, filter*	Date when the last renewal order for this subscription was created.
LastRenewalDate	*date*	*read, write, filter*	Date when this subscription was last renewed.
NumberOfRenewals	*int*	*read, write*	Specifies how many times this subscription has been renewed.
SubscriptionItems	*empty*	*read*	A collection of all items within this subscription. The "href" attribute contains the collection URI. See below for details.
CreatedTime	*datetime*	*read, filter*	The date and time when this record was created.
ChangedTime	*datetime*	*read, filter*	The date and time when this record was last changed.
SyncedTime	*datetime*	*read, filter*	The date and time when this record was last synchronized.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Subscription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/subscription/1">
3   <Customer href="/__API__/customer/1">1</Customer>
4   <State>ACTIVE</State>
```

```
5   <PaymentMethod href="/__API__/paymentmethod/831" xsi:nil="true" />
6   <ShippingMethod href="/__API__/shippingmethod/117" xsi:nil="true" />
7   <AutomaticRenewals>true</AutomaticRenewals>
8   <AnnualWorth>5504</AnnualWorth>
9   <Currency>SEK</Currency>
10  <FreeEveryNRenewals xsi:nil="true" />
11  <SubscriptionStartedAt>2010-01-25T13:24:57Z</SubscriptionStartedAt>
12  <ExpireDate>2012-01-02</ExpireDate>
13  <NextRenewalDate>2012-01-02</NextRenewalDate>
14  <ReorderDate>2012-12-02</ReorderDate>
15  <DaysFromReorderToRenewal>32</DaysFromReorderToRenewal>
16  <LastRenewalOrderDate xsi:nil="true" />
17  <LastRenewalDate xsi:nil="true" />
18  <NumberOfRenewals xsi:nil="true" />
19  <SubscriptionItems href="/__API__/subscription/1/item" xsi:nil="true"
        />
20  <CreatedTime>2010-01-25T15:04:54Z</CreatedTime>
21  <ChangedTime>2012-09-05T14:02:21Z</ChangedTime>
22  <SyncedTime xsi:nil="true" />
23 </Subscription>
```

3.22.3 SubscriptionItem Collection

URI:	*As specified in Subscription Items element*
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml
GET Parameters:	synced, mark_as_synced, filter
POST Parameters:	mark_as_synced

A list of `SubscriptionItem` objects.

Query parameters

Name	Values	Description
`synced`	*yes,no*	As defined in section 2.11.2.
`mark_as_synced`	*yes,no*	As defined in section 2.11.2.

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <SubscriptionItemList>
3   <SubscriptionItem
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/subscription/1/item/1">
4     <Product href="/__API__/product/18" xsi:nil="true" />
5     <Variant href="/__API__/product/18/variant/31" xsi:nil="true" />
```

```
6     <InitialPrice xsi:nil="true" />
7     <InitialTax xsi:nil="true" />
8     <RenewalPrice>321</RenewalPrice>
9     <RenewalTax>76</RenewalTax>
10    <ExtraText1 xsi:nil="true" />
11    <ExtraText2 xsi:nil="true" />
12    <CreatedTime>2010-01-25T15:06:02Z</CreatedTime>
13    <ChangedTime>2010-01-26T10:26:51Z</ChangedTime>
14    <SyncedTime xsi:nil="true" />
15  </SubscriptionItem>
16  <SubscriptionItem
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/subscription/1/item/3">
17    <Product href="/__API__/product/19" xsi:nil="true" />
18    <Variant href="/__API__/product/19/variant/33" xsi:nil="true" />
19    <InitialPrice xsi:nil="true" />
20    <InitialTax>0</InitialTax>
21    <RenewalPrice>500</RenewalPrice>
22    <RenewalTax>125</RenewalTax>
23    <ExtraText1 xsi:nil="true" />
24    <ExtraText2 xsi:nil="true" />
25    <CreatedTime>2010-01-26T10:03:10Z</CreatedTime>
26    <ChangedTime>2010-01-26T10:03:10Z</ChangedTime>
27    <SyncedTime xsi:nil="true" />
28  </SubscriptionItem>
29  <SubscriptionItem
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/subscription/1/item/6">
30    <Product href="/__API__/product/19" xsi:nil="true" />
31    <Variant href="/__API__/product/19/variant/33" xsi:nil="true" />
32    <InitialPrice>42</InitialPrice>
33    <InitialTax xsi:nil="true" />
34    <RenewalPrice>555</RenewalPrice>
35    <RenewalTax>125</RenewalTax>
36    <ExtraText1 xsi:nil="true" />
37    <ExtraText2 xsi:nil="true" />
38    <CreatedTime>2010-01-26T12:38:50Z</CreatedTime>
39    <ChangedTime>2010-01-26T12:38:50Z</ChangedTime>
40    <SyncedTime xsi:nil="true" />
41  </SubscriptionItem>
42 </SubscriptionItemList>
```

3.22.4 SubscriptionItem

URI:	*As specified in href attribute.*
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml
GET Parameters:	synced, mark_as_synced
PUT Parameters:	mark_as_synced

Query parameters

Name	Values	Description
synced	*yes,no*	As defined in section 2.11.2.
mark_as_synced	*yes,no*	As defined in section 2.11.2.

Object description

Name	Type	Mode	Description
Product	*empty*	*read*	A reference to the product containing the Variant specified in this item. This is just a convenience-element, and cannot be changed.
Variant	*empty*	*read, write*	The variant the customer has subscribed to. The "href" attribute contains the URI to a *Variant* object.
InitialPrice	*float*	*read, write*	The price of the first purchase of this item. Is only used for documentation. Doesn't need to be the same as the Variant price.
InitialTax	*float*	*read, write*	The tax of the first purchase of this item. Is only used for documentation. Doesn't need to be the same as the Variant tax.
RenewalPrice	*float*	*read, write*	The renewal price of this item. Doesn't need to be the same as the Variant price.
RenewalTax	*float*	*read, write*	The renewal tax of this item. Doesn't need to be the same as the Variant tax.
ExtraText1..2	*str*	*read, write, filter*	Two (2) extra textfields for extra information that the shop might want.
CreatedTime	*datetime*	*read, filter*	The date and time when this record was created.
ChangedTime	*datetime*	*read, filter*	The date and time when this record was last changed.
SyncedTime	*datetime*	*read, filter*	The date and time when this record was last synchronized.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SubscriptionItem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/subscription/1/item/1">
  <Product href="/__API__/product/18" xsi:nil="true" />
  <Variant href="/__API__/product/18/variant/31" xsi:nil="true" />
  <InitialPrice xsi:nil="true" />
  <InitialTax xsi:nil="true" />
  <RenewalPrice>321</RenewalPrice>
  <RenewalTax>76</RenewalTax>
  <ExtraText1 xsi:nil="true" />
  <ExtraText2 xsi:nil="true" />
  <CreatedTime>2010-01-25T15:06:02Z</CreatedTime>
  <ChangedTime>2010-01-26T10:26:51Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</SubscriptionItem>
```

3.23 Affiliates

3.23.1 Affiliate Program Collection

URI:	`/__API__/affiliateprogram`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<AffiliateProgramList>...</AffiliateProgramList>` is an array of abbreviated `AffiliateProgram` objects.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AffiliateProgramList>
  <AffiliateProgram
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/affiliateprogram/1">
    <ID>Tag</ID>
    <Name>Testaffiliat</Name>
    <MemberList href="/__API__/affiliateprogram/1/member" xsi:nil="true"
       />
  </AffiliateProgram>
</AffiliateProgramList>
```

3.23.2 Affiliate Program

URI:	*As specified in href attribute.*
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID/tag set for this affiliate program.
`Name`	*str*	*read, write*	The name of the program.
`Description`	*str*	*read, write*	The program description.

contd...

Name	Type	Mode	Description
Instructions	*str*	*read, write*	Instructions for this program that are displayed to the customer. This field is displayed to the customer on the web and MAY contain HTML as long as the HTML is properly coded as an XML string.
OrderWithinDays	*int*	*read, write*	When a customer comes in via an affiliate member of this program they must place an order within this amount of days for the affiliate to receive kickback.
NewCustomersOnly	*bool*	*read, write*	Kickback is awarded only for new customers.
FirstOrderOnly	*bool*	*read, write*	Kickback is awarded only for the first order the customer purchases.
FixedReward	*float*	*read, write*	A fixed reward of this amount is awarded per order generated.
RewardInPercent	*float*	*read, write*	This percentage of the order is awarded as kickback.
QuarantineDays	*int*	*read, write*	A new order is in quarantine for this amount of days before any kickback is awarded.
ExtraText1..5	*str*	*read, write*	Extra textfields for extra information that the shop might want.
MemberList	*empty*	*read*	A reference to the affiliate program members collection. The "href" attribute contains the URI for the collection.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AffiliateProgram xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/affiliateprogram/1">
  <ID>Tag</ID>
  <Name>Testaffiliat</Name>
  <Description>Description</Description>
  <Instructions>&lt;p&gt;Customer instructions!&lt;/p&gt;</Instructions>
  <OrderWithinDays>365</OrderWithinDays>
  <NewCustomersOnly>true</NewCustomersOnly>
  <FirstOrderOnly>true</FirstOrderOnly>
  <FixedReward>10</FixedReward>
  <RewardInPercent>5</RewardInPercent>
  <QuarantineDays>60</QuarantineDays>
  <ExtraText1 xsi:nil="true" />
  <ExtraText2 xsi:nil="true" />
  <ExtraText3 xsi:nil="true" />
  <ExtraText4 xsi:nil="true" />
  <ExtraText5 xsi:nil="true" />
  <RedirectTo>/product.html/produkt-nummer-ett</RedirectTo>
  <MemberList href="/__API__/affiliateprogram/1/member" xsi:nil="true" />
</AffiliateProgram>
```

3.23.3 Member Collection

URI:	*As specified in MemberList element*
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<AffiliateList>...</AffiliateList>` is an array of complete `Affiliate` objects.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AffiliateList>
  <Affiliate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/affiliateprogram/1/member/1">
    <ID xsi:nil="true" />
    <Customer href="/__API__/customer/2">2</Customer>
    <State>ACTIVE</State>
    <Activator>TXKuet2tXDeAZgMShC</Activator>
  </Affiliate>
</AffiliateList>
```

3.23.4 Affiliate

URI:	*As specified in href attribute.*
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml

Object description

Name	Type	Mode	Description
`ID`	*str*	*read, write*	The ID/tag set for this affiliate program.
`Customer`	*str*	*read, write*	The Customer object representing this affiliate is referenced using the "href" attribute. To aid clients the customer number is sent as the element data, but only the "href" attribute is used during update/create.
`State`	*str*	*read, write*	One of "ACTIVE", "HIDDEN".
`Activator`	*str(20)*	*read, write*	An unique "activator" string that when prepended to the correct URL (usually *http://<SHOP DOMAIN>/AFF/*) marks the new customer as having arrived via this affiliate. MUST be at most 20 characters.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Affiliate xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    href="/__API__/affiliateprogram/1/member/1">
  <ID xsi:nil="true" />
  <Customer href="/__API__/customer/2">2</Customer>
  <State>ACTIVE</State>
  <Activator>TXKuet2tXDeAZgMShC</Activator>
</Affiliate>
```

3.24 Symbols

3.24.1 Symbol Collection

URI:	`/__API__/symbol`
Actions:	Retrieve(GET), Create(POST)
Content-Type:	text/xml

Object description

The returned object `<SymbolList>...</SymbolList>` is an array of abbreviated `Symbol` objects.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SymbolList>
  <Symbol feature_id="1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      href="/__API__/symbol/1">
    <ID xsi:nil="true" />
    <Name>Symbol 1</Name>
    <ShortDescription>This is the 1st symbol</ShortDescription>
    <Description xsi:nil="true" />
    <WebLink>http://symbol.example.com</WebLink>
  </Symbol>
</SymbolList>
```

3.24.2 Symbol

URI:	*As specified in href attribute.*
Actions:	Retrieve(GET), Update(PUT), Delete(DELETE)
Content-Type:	text/xml
Attributes:	href, feature_id

Object description

The `Symbol` element contains a "href" attribute specifying the complete URI for this symbol. It also contains a "feature_id" attribute that is the feature_id parameter used in the shop front end. This might be used by the API user to create links that link into a specific symbol in the shop.

Name	Type	Mode	Description

contd...

Name	Type	Mode	Description
ID	*str*	*read, write*	The ID/tag of this symbol.
Name	*str*	*read, write*	The symbol name.
ShortDescription	*str*	*read, write*	A short non-html description.
Description	*str*	*read, write*	The description.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Symbol feature_id="1"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     href="/__API__/symbol/1">
   <ID xsi:nil="true" />
   <Name>Symbol 1</Name>
   <ShortDescription>This is the 1st symbol</ShortDescription>
   <Description xsi:nil="true" />
   <WebLink>http://symbol.example.com</WebLink>
   <Picture href="/__API__/picture/69" xsi:nil="true" />
</Symbol>
```

3.25 Extrafields

Extra fields are custom shop-owner defined fields connected to a particular resource.

This section describes the configuration resources for extrafields. Each resource that has extrafields will have its own subresource for configuration. All these resources work as described in this section.

3.25.1 Extrafield resource

URI:	/__API__/<resource>/extrafields
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml

Object description

The Extrafields object contains configuration data for the extrafields that are legal for a particular resource.

Extrafields can be of two types: Text fields and selectors. Not all resources support both types.

Text fields are fields that contain some text string for their value. The shop usually does not enforce any semantics or syntax of this value. For text fields only their label can be configured.

Selectors contain a numeric value that is the index into a list of values. The list is indexed from 1, and if a selector has a value of '0' that means that no value was chosen. För selector fields both their label and the list of values can be configured.

Name	Type	Mode	Description
ExtraTextFields	*list*	*read, write*	List of textfields available for this resource. The data within each element is the field label. If this element is omitted by the server that menas the resource doesn't support text fields.
ExtraSelectorFields	*list*	*read, write*	List of selector fields available for this resource. Each item contains a label and a list of values. If this element is omitted by the server that menas the resource doesn't support selector fields.

Selector field data

Name	Type	Mode	Description
Label	*str*	*read, write*	The label for this selector field
Values	*array of str*	*read, write*	Contains an array of <*Value*> elements that contain the values this selector may take in order. That is the first item in the list is the value with index 1, the second item has index 2 and so on.

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Extrafields xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ExtraTextFields>
    <ExtraText1>test1</ExtraText1>
    <ExtraText2>test2</ExtraText2>
    <ExtraText3 xsi:nil="true" />
    <ExtraText4 xsi:nil="true" />
    <ExtraText5 xsi:nil="true" />
    <ExtraText6 xsi:nil="true" />
  </ExtraTextFields>
  <ExtraSelectorFields>
    <ExtraSelector1>
      <Label>sel1</Label>
      <Values>
        <Value>item 1</Value>
        <Value>item 2</Value>
        <Value>item 3</Value>
      </Values>
    </ExtraSelector1>
    <ExtraSelector2>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector2>
```

```
    <ExtraSelector3>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector3>
    <ExtraSelector4>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector4>
    <ExtraSelector5>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector5>
    <ExtraSelector6>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector6>
    <ExtraSelector7>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector7>
    <ExtraSelector8>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector8>
    <ExtraSelector9>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector9>
    <ExtraSelector10>
      <Label xsi:nil="true" />
      <Values>
      </Values>
    </ExtraSelector10>
  </ExtraSelectorFields>
</Extrafields>
```

3.26 Locks and Keys

Locks and keys are a feature to allow or block customers and customer categories access to products, categories, pay methods, shipping methods, documents, pages and news feeds.

Customers and customer categories are assigned keys whilst products, categories, pay methods,

shipping methods, documents, pages and news feeds are assigned either locks to allow access if there is a matching key or anti locks to block access if there is a matching key.

Pages and news feeds are currently not accessible via the API.

3.26.1 Locks and keys Collection

URI:	`/__API__/lockkey`
Actions:	Retrieve(GET)
Content-Type:	text/xml

Object description

The returned object `<LockKeyList>...</LockKeyList>` is an array of `LockKey` objects:

Name	Type	Mode	Description
`Index`	*int*	*read, write*	The index for the lock
`Name`	*str*	*read, write*	The name for the lock

Example XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<LockKeyList>
    <LockKey href="/__API__/lockkey/0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
         <Index>0</Index>
         <Name>Test - Lock 1</Name>
    </LockKey>
    <LockKey xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        href="/__API__/lockkey/7">
         <Index>7</Index>
         <Name>Test - Lock 7</Name>
    </LockKey>
</LockKeyList>
```

3.26.2 Locks and keys

URI:	`/__API__/lockkey/...` (from href attribute)
Actions:	Retrieve(GET), Update(PUT)
Content-Type:	text/xml
Attributes:	href

To remove a lock, update the index with an empty string.

Only locks with a defined `Name` will be visible in the list.

Name	Type	Mode	Description
Index	*int*	*read, write*	The index for the lock
Name	*str*	*read, write*	The name for the lock

The Index is an integer from 0 to 63. Values outside this range will generate an error 1017.

If Index is supplied in an PUT request and it does not match the number supplied in the URI an mismatch error 1018 will be returned. (Section A)

Example XML

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <LockKey href="/__API__/lockkey/7"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3      <Index>7</Index>
4      <Name>Test - Lock 7</Name>
5 </LockKey>
```

CHAPTER 4

EXAMPLE SCENARIOS

This chapter contains some example scenarios for interaction with the API. Please note that the XML dumps in this chapter may be out of date. Please refer to earlier chapters for details.

4.1 Adding / Updating / Deleting Customers

4.1.1 Adding two new customers

Say we want to add three new customers to the system. These are the customers we wish to add:

1. Customer Number: 10042
   Name: Arthur Dent
   Login: arthurdent@example.com
   Password: dent
   Email: arthurdent@example.com
2. Customer Number: 10060
   Name: Niels Bohr
   Login: bohr@example.com
   Password: bohr
   Email: bohr@example.com Discount: 5%

To add these customers a `POST` request needs to be made to `/__API__/customer` for each customer. The `POST` request body contains the XML data describing the customer according to section 3.2.2. Only the fields we are interested in changing need to be sent. The server will reply with the complete customer object and a `Location` header containing the URI for this new customer[1]. This URI is needed when updating or deleting the customer.

[1]The URI is also available in the "href" attribute of the Customer object.

Adding customer 1 (Mr Dent)

The XML message below is `POST`ed to `/__API__/customer`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer>
  <CustomerNo>10042</CustomerNo>
  <Type>PERSON</Type>
  <FirstName>Arthur</FirstName>
  <LastName>Dent</LastName>
  <EMail>arthurdent@example.com</EMail>
  <Login>arthurdent@example.com</Login>
  <Password>dent</Password>
</Customer>
```

The server replies with a `Location` header containing the URI for the customer (for example `/__API__/customer/29`) and the complete customer object. The URI for this customer is saved by the client for future reference.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" customer_id="29" href=
  <CustomerNo>10042</CustomerNo>
  <Type>PERSON</Type>
  <OneTimeCustomer>false</OneTimeCustomer>
  <Mode>ACTIVE</Mode>
  <OrgNo xsi:nil="true" />
  <VatNo xsi:nil="true" />
  <FirstName>Arthur</FirstName>
  <LastName>Dent</LastName>
  <Company xsi:nil="true" />
  <Address1 xsi:nil="true" />
  <Address2 xsi:nil="true" />
  <ZIP xsi:nil="true" />
  <City xsi:nil="true" />
  <State xsi:nil="true" />
  <Country>SE</Country>
  <Language>en</Language>
  <PhoneNo xsi:nil="true" />
  <CellPhoneNo xsi:nil="true" />
  <FaxNo xsi:nil="true" />
  <EMail>arthurdent@example.com</EMail>
  <DiscountPercent xsi:nil="true" />
```

```
  <CreditLimit xsi:nil="true" />
  <CreditUsed xsi:nil="true" />
  <Currency xsi:nil="true" />
  <Login>arthurdent@example.com</Login>
  <Comment></Comment>
  <WantsNewsletter>true</WantsNewsletter>
  <HaveSentWelcomeEmail>false</HaveSentWelcomeEmail>
  <GroupDiscount>
  </GroupDiscount>
  <CustomerOrders href="/__API__/customer/29/orders"
  xsi:nil="true" />
  <Subscriptions href="/__API__/customer/29/subscriptions"
  xsi:nil="true" />
  <ExtraText1 xsi:nil="true" />
  <ExtraText2 xsi:nil="true" />
  <ExtraText3 xsi:nil="true" />
  <ExtraText4 xsi:nil="true" />
  <ExtraText5 xsi:nil="true" />
  <ExtraText6 xsi:nil="true" />
  <ExtraSelector1>0</ExtraSelector1>
  <ExtraSelector2>0</ExtraSelector2>
  <ExtraSelector3>0</ExtraSelector3>
  <ExtraSelector4>0</ExtraSelector4>
  <ExtraSelector5>0</ExtraSelector5>
  <ExtraSelector6>0</ExtraSelector6>
  <ExtraSelector7>0</ExtraSelector7>
  <ExtraSelector8>0</ExtraSelector8>
  <ExtraSelector9>0</ExtraSelector9>
  <ExtraSelector10>0</ExtraSelector10>
  <CreateLoginToken href="/__API__/customer/29/createlogintoken"
  xsi:nil="true" />
  <DeliveryAddresses href="/__API__/customer/29/deliveryaddress"
  xsi:nil="true" />
  <PreferredDeliveryAddress xsi:nil="true" />
  <CreatedTime>2012-09-11T12:01:36Z</CreatedTime>
  <ChangedTime>2012-09-11T12:01:36Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</Customer>
```

Adding customer 2 (Mr Bohr)

Adding customer 2 is done exactly like customer 1. Customer 2 has a discount set so we send the `Discount` element in the XML message as well. A `POST` request is made to `/__API__/customer`:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer>
  <CustomerNo>10060</CustomerNo>
  <Type>PERSON</Type>
  <FirstName>Niels</FirstName>
  <LastName>Bohr</LastName>
  <EMail>bohr@example.com</EMail>
  <Login>bohr@example.com</Login>
  <Password>bohr</Password>
  <DiscountPercent>5</DiscountPercent>
</Customer>
```

And the server replies with:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" customer_id="30" href=
  <CustomerNo>10060</CustomerNo>
  <Type>PERSON</Type>
  <OneTimeCustomer>false</OneTimeCustomer>
  <Mode>ACTIVE</Mode>
  <OrgNo xsi:nil="true" />
  <VatNo xsi:nil="true" />
  <FirstName>Niels</FirstName>
  <LastName>Bohr</LastName>
  <Company xsi:nil="true" />
  <Address1 xsi:nil="true" />
  <Address2 xsi:nil="true" />
  <ZIP xsi:nil="true" />
  <City xsi:nil="true" />
  <State xsi:nil="true" />
  <Country>SE</Country>
  <Language>en</Language>
  <PhoneNo xsi:nil="true" />
  <CellPhoneNo xsi:nil="true" />
  <FaxNo xsi:nil="true" />
  <EMail>bohr@example.com</EMail>
  <DiscountPercent>5</DiscountPercent>
```

```
  <CreditLimit xsi:nil="true" />
  <CreditUsed xsi:nil="true" />
  <Currency xsi:nil="true" />
  <Login>bohr@example.com</Login>
  <Comment></Comment>
  <WantsNewsletter>true</WantsNewsletter>
  <HaveSentWelcomeEmail>false</HaveSentWelcomeEmail>
  <GroupDiscount>
  </GroupDiscount>
  <CustomerOrders href="/__API__/customer/30/orders" xsi:nil="true" />
  <Subscriptions href="/__API__/customer/30/subscriptions"
xsi:nil="true" />
  <ExtraText1 xsi:nil="true" />
  <ExtraText2 xsi:nil="true" />
  <ExtraText3 xsi:nil="true" />
  <ExtraText4 xsi:nil="true" />
  <ExtraText5 xsi:nil="true" />
  <ExtraText6 xsi:nil="true" />
  <ExtraSelector1>0</ExtraSelector1>
  <ExtraSelector2>0</ExtraSelector2>
  <ExtraSelector3>0</ExtraSelector3>
  <ExtraSelector4>0</ExtraSelector4>
  <ExtraSelector5>0</ExtraSelector5>
  <ExtraSelector6>0</ExtraSelector6>
  <ExtraSelector7>0</ExtraSelector7>
  <ExtraSelector8>0</ExtraSelector8>
  <ExtraSelector9>0</ExtraSelector9>
  <ExtraSelector10>0</ExtraSelector10>
  <CreateLoginToken href="/__API__/customer/30/createlogintoken"
xsi:nil="true" />
  <DeliveryAddresses href="/__API__/customer/30/deliveryaddress"
xsi:nil="true" />
  <PreferredDeliveryAddress xsi:nil="true" />
  <CreatedTime>2012-09-11T12:10:47Z</CreatedTime>
  <ChangedTime>2012-09-11T12:10:47Z</ChangedTime>
  <SyncedTime xsi:nil="true" />
</Customer>
```

The client saves the customer URI (`/__API__/customer/30`) for future reference, this URI is sent by the server both in a `Location` HTTP header and in the *href* attribute.

Note that the client only needs to send the fields it wishes to change. All other fields will be set to defaults.

4.1.2 Updating a customer record

So it happens that Mr Bohr invents some new thing that earns us a lot of money so management decides to increase his discount in our store to 15%.

Thus, the client sends a `PUT` request to `/__API__/customer/30` [2] containing an XML message with the required changes:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer>
  <DiscountPercent>15</DiscountPercent>
</Customer>
```

The server will reply with a complete Customer object. No `Location` header is sent this time since the customer URI doesn't change when doing updates. Although the server sends the entire customer object only the relevant parts are shown below:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" customer_id="30" href=
  ...
  <DiscountPercent>15</DiscountPercent>
  ...
</Customer>
```

4.1.3 Deleting a customer

Unfortunately it so happens that Mr Bohr has passed away, and thus needs to be removed from the web shop. This is done by sending a `DELETE` request to the customer URI (`/__API__/customer/30` for Mr Bohr. The client MUST NOT send any data in the request body. The server will reply with a 204 status code to indicate success.

At this point the client MUST forget the customer URI for Mr Bohr as it is no longer valid for any operations.

4.2 First time import of products

When new products have been added to a back end system (or this is the first time the system communicates with the webshop) all these new products need to be sent to the web shop.

[2] The client has recorded the URI when it created the customer.

The client must then send a `POST` request to create each product in the web shop. This works more or less exactly as the Customer examples in the previous section.

For each product created in the shop it is vital that the client records the product URI that was returned from the API so that the products may be modified or deleted later on.

4.3 Retrieving order data for all new purchases

A client should periodically query the web shop for any new purchases that customers have made. The querying interval can be anything from a few minutes to once a day (it is rarely a good idea to check less often than daily).

To be able to retrieve only new purchases the client keeps track of the changed date of the last order it has fetched. For a more detailed description of the various sync procedures available see section 2.11.

The first time the client must fetch the complete order list by doing a `GET` request to `/__API__/order`. The client will get a list of all orders in the shop.

In the future when the client needs to get a hold of all new or updated orders and it has recorded the ChangedTime of the last order it has fetched (for example 2012-02-10T13:11:57Z) the client simply does a `GET` request to `/__API__/order?filter?ChangedTime%3E%3D2012-02-10T13:11:57Z`. The server will then send all orders that have a changed date of 2012-02-10T13:11:57Z or later. Please note that this may very well list orders that have already been fetched if these orders have been changed in the web shop. This condition MUST be handled by the client (for example by ignoring orders it has already fetched if it doesn't care about changes). Note that the timestamp used MUST be exactly that which was returned from the API for the latest order fetched. The format of the timestamp is described in section 2.14.2.

This same procedure can be used for any resource that supports `filter`, for example to retrieve updated Customer objects for customers that have changed their profile.

4.4 Partial Updates

In the example below the client issues a request to update the credit limits of a customer. The request is a partial update containing only those attributes that are to be changed.

The Customer object before the change looks like this:

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      customer_id="1" href="/__API__/customer/1">
3   <CustomerNo>1</CustomerNo>
4   <Type>PERSON</Type>
```

```
  <OneTimeCustomer>false</OneTimeCustomer>
  <Mode>ACTIVE</Mode>
  <OrgNo xsi:nil="true" />
  <VatNo xsi:nil="true" />
  <FirstName>Tess T.</FirstName>
  <LastName>Persson</LastName>
  <Company xsi:nil="true" />
  <Address1>Testgatan 42</Address1>
  <Address2 xsi:nil="true" />
  <ZIP>123 45</ZIP>
  <City>Testcity</City>
  <State xsi:nil="true" />
  <Country>SE</Country>
  <Language>sv</Language>
  <PhoneNo>012-345678</PhoneNo>
  <CellPhoneNo xsi:nil="true" />
  <FaxNo xsi:nil="true" />
  <EMail>customer@example.com</EMail>
  <DiscountPercent xsi:nil="true" />
  <CreditLimit>10000</CreditLimit>
  <CreditUsed>5000</CreditUsed>
  <Currency xsi:nil="true" />
  <Login>customer@example.com</Login>
  <Comment xsi:nil="true" />
  <WantsNewsletter>false</WantsNewsletter>
  <HaveSentWelcomeEmail>false</HaveSentWelcomeEmail>
  <CustomerCategory href="/__API__/customercategory/2" xsi:nil="true" />
  <Pricelist href="/__API__/pricelist/1" xsi:nil="true" />
  <Warehouse href="/__API__/warehouse/3" xsi:nil="true" />
  <GroupDiscount>
    <Item>
      <DiscountGroup href="/__API__/discountgroup/1" xsi:nil="true" />
      <DiscountPercent>5</DiscountPercent>
    </Item>
    <Item>
      <DiscountGroup href="/__API__/discountgroup/2" xsi:nil="true" />
      <DiscountPercent>2</DiscountPercent>
    </Item>
  </GroupDiscount>
  <CustomerOrders href="/__API__/customer/1/orders" xsi:nil="true" />
  <Subscriptions href="/__API__/customer/1/subscriptions" xsi:nil="true"
     />
  <ExtraText1>t1</ExtraText1>
  <ExtraText2>t2</ExtraText2>
  <ExtraText3 xsi:nil="true" />
  <ExtraText4 xsi:nil="true" />
  <ExtraText5 xsi:nil="true" />
  <ExtraText6 xsi:nil="true" />
  <ExtraSelector1 textvalue="item 2">2</ExtraSelector1>
  <ExtraSelector2>0</ExtraSelector2>
  <ExtraSelector3>0</ExtraSelector3>
```

```
55   <ExtraSelector4>0</ExtraSelector4>
56   <ExtraSelector5>0</ExtraSelector5>
57   <ExtraSelector6>0</ExtraSelector6>
58   <ExtraSelector7>0</ExtraSelector7>
59   <ExtraSelector8>0</ExtraSelector8>
60   <ExtraSelector9>0</ExtraSelector9>
61   <ExtraSelector10>0</ExtraSelector10>
62   <CreateLoginToken href="/__API__/customer/1/createlogintoken"
        xsi:nil="true" />
63   <ViaAffiliate href="/__API__/affiliateprogram/1/member/1"
        xsi:nil="true" />
64   <DeliveryAddresses href="/__API__/customer/1/deliveryaddress"
        xsi:nil="true" />
65   <PreferredDeliveryAddress href="/__API__/customer/1/deliveryaddress/1"
        xsi:nil="true" />
66   <CustomAttributes>
67     <CustomAttribute href="/__API__/customattribute/1">
68       <ID xsi:nil="true" />
69       <ValueID xsi:nil="true" />
70       <Name>Size</Name>
71       <Value href="/__API__/customattribute/1/value/3">L</Value>
72     </CustomAttribute>
73     <CustomAttribute href="/__API__/customattribute/2">
74       <ID xsi:nil="true" />
75       <ValueID xsi:nil="true" />
76       <Name>Color</Name>
77       <Value href="/__API__/customattribute/2/value/4">Cyan</Value>
78       <Value href="/__API__/customattribute/2/value/5">Magenta</Value>
79       <Value href="/__API__/customattribute/2/value/6">Yellow</Value>
80       <Value href="/__API__/customattribute/2/value/7">Black</Value>
81     </CustomAttribute>
82   </CustomAttributes>
83   <CreatedTime>2009-04-05T08:22:06Z</CreatedTime>
84   <ChangedTime>2016-08-30T09:37:32Z</ChangedTime>
85   <SyncedTime>2010-02-03T17:51:23Z</SyncedTime>
86 </Customer>
```

The client sends the following partial object to perform the update:

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Customer>
3   <CreditLimit>25000</CreditLimit>
4   <CreditUsed>0</CreditUsed>
5 </Customer>
```

Which results in the Customer object below:

```
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <Customer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     customer_id="1" href="/__API__/customer/1">
3   <CustomerNo>1</CustomerNo>
4   <Type>PERSON</Type>
```

```
  <OneTimeCustomer>false</OneTimeCustomer>
  <Mode>ACTIVE</Mode>
  <OrgNo xsi:nil="true" />
  <VatNo xsi:nil="true" />
  <FirstName>Tess T.</FirstName>
  <LastName>Persson</LastName>
  <Company xsi:nil="true" />
  <Address1>Testgatan 42</Address1>
  <Address2 xsi:nil="true" />
  <ZIP>123 45</ZIP>
  <City>Testcity</City>
  <State xsi:nil="true" />
  <Country>SE</Country>
  <Language>sv</Language>
  <PhoneNo>012-345678</PhoneNo>
  <CellPhoneNo xsi:nil="true" />
  <FaxNo xsi:nil="true" />
  <EMail>customer@example.com</EMail>
  <DiscountPercent xsi:nil="true" />
  <CreditLimit>25000</CreditLimit>
  <CreditUsed>0</CreditUsed>
  <Currency xsi:nil="true" />
  <Login>customer@example.com</Login>
  <Comment xsi:nil="true" />
  <WantsNewsletter>false</WantsNewsletter>
  <HaveSentWelcomeEmail>false</HaveSentWelcomeEmail>
  <CustomerCategory href="/__API__/customercategory/2" xsi:nil="true" />
  <Pricelist href="/__API__/pricelist/1" xsi:nil="true" />
  <Warehouse href="/__API__/warehouse/3" xsi:nil="true" />
  <GroupDiscount>
    <Item>
      <DiscountGroup href="/__API__/discountgroup/1" xsi:nil="true" />
      <DiscountPercent>5</DiscountPercent>
    </Item>
    <Item>
      <DiscountGroup href="/__API__/discountgroup/2" xsi:nil="true" />
      <DiscountPercent>2</DiscountPercent>
    </Item>
  </GroupDiscount>
  <CustomerOrders href="/__API__/customer/1/orders" xsi:nil="true" />
  <Subscriptions href="/__API__/customer/1/subscriptions" xsi:nil="true"
     />
  <ExtraText1>t1</ExtraText1>
  <ExtraText2>t2</ExtraText2>
  <ExtraText3 xsi:nil="true" />
  <ExtraText4 xsi:nil="true" />
  <ExtraText5 xsi:nil="true" />
  <ExtraText6 xsi:nil="true" />
  <ExtraSelector1 textvalue="item 2">2</ExtraSelector1>
  <ExtraSelector2>0</ExtraSelector2>
  <ExtraSelector3>0</ExtraSelector3>
```

```
55    <ExtraSelector4>0</ExtraSelector4>
56    <ExtraSelector5>0</ExtraSelector5>
57    <ExtraSelector6>0</ExtraSelector6>
58    <ExtraSelector7>0</ExtraSelector7>
59    <ExtraSelector8>0</ExtraSelector8>
60    <ExtraSelector9>0</ExtraSelector9>
61    <ExtraSelector10>0</ExtraSelector10>
62    <CreateLoginToken href="/__API__/customer/1/createlogintoken"
          xsi:nil="true" />
63    <ViaAffiliate href="/__API__/affiliateprogram/1/member/1"
          xsi:nil="true" />
64    <DeliveryAddresses href="/__API__/customer/1/deliveryaddress"
          xsi:nil="true" />
65    <PreferredDeliveryAddress href="/__API__/customer/1/deliveryaddress/1"
          xsi:nil="true" />
66    <CustomAttributes>
67      <CustomAttribute href="/__API__/customattribute/1">
68        <ID xsi:nil="true" />
69        <ValueID xsi:nil="true" />
70        <Name>Size</Name>
71        <Value href="/__API__/customattribute/1/value/3">L</Value>
72      </CustomAttribute>
73      <CustomAttribute href="/__API__/customattribute/2">
74        <ID xsi:nil="true" />
75        <ValueID xsi:nil="true" />
76        <Name>Color</Name>
77        <Value href="/__API__/customattribute/2/value/4">Cyan</Value>
78        <Value href="/__API__/customattribute/2/value/5">Magenta</Value>
79        <Value href="/__API__/customattribute/2/value/6">Yellow</Value>
80        <Value href="/__API__/customattribute/2/value/7">Black</Value>
81      </CustomAttribute>
82    </CustomAttributes>
83    <CreatedTime>2009-04-05T08:22:06Z</CreatedTime>
84    <ChangedTime>2016-08-30T09:37:32Z</ChangedTime>
85    <SyncedTime>2010-02-03T17:51:23Z</SyncedTime>
86  </Customer>
```

4.5 Examples in PHP

A very simple and limited example client written in PHP can be tested online and downloaded at:
http://api.nordiskehandel.com

4.6 Examples using a Web Browser

Since this API uses the HTTP/1.1 protocol as an application protocol and not just as a transport tunnel a normal browser may be used to retrieve and inspect the resources provided by the API.

Assuming that you are accessing the shop at `apitest.testbutiken.se` you can retrieve the "Root"[3] resource by typing the following URI into your web browser:
`https://apitest.testbutiken.se/__API__/`

Your browser should prompt you for an username and password.

To access the "Customer"[4] collection just enter this URI:
`https://apitest.testbutiken.se/__API__/customer`

and to access a specific customer enter:
`https://apitest.testbutiken.se/<Customer-URI>`

where `<Customer-URI>` is the URI from the "href" attribute of the customer object you wish to view.

You can access all other objects in a similar manner. Even the picture-data URI[5] can be viewed in this manner, and should display the picture in your browser.

4.7 Examples using cURL

`cURL` is a powerful command-line client and library for retrieving and sending data using HTTP. For more information see [cURL Homepage].

4.7.1 Retrieving an Object

To retrieve the "Root" resource of the shop located at domain `apitest.testbutiken.se` using the username `api` and the password `testingAPI` one can use the following command:

```
curl -kv "https://api:testingAPI@apitest.testbutiken.se/__API__/"
```

[3] Defined in section 3.1
[4] Defined in section 3.2
[5] Defined in section 3.9.3

4.7.2 Updating an Object

To update a Customer using the `PUT` method provided the object data is in `text/xml` and is located in the file `data.xml` one can use the following command:

```
curl -kv "https://api:testingAPI@apitest.testbutiken.se/__API__/customer/1" \
    -X PUT -H "Content-Type: text/xml" --data-binary @data.xml
```

To update the image data of a picture object provided that the picture is a `PNG`-image located in the file `pic.png` one can use the following command:

```
curl -kv "https://api:testingAPI@apitest.testbutiken.se/__API__/picture/5/data" \
     -X PUT -H "Content-Type: image/png" --data-binary @pic.png
```

4.7.3 Creating an Object

To create a Customer using the `POST` method provided the object data is in `text/xml` and is located in the file `data.xml` one can use the following command:

```
curl -kv "https://api:testingAPI@apitest.testbutiken.se/__API__/customer" \
    -X POST -H "Content-Type: text/xml" --data-binary @data.xml
```

4.7.4 Deleting an Object

To delete a Customer using the `DELETE` method one can use the following command:

```
curl -kv "https://api:testingAPI@apitest.testbutiken.se/__API__/customer/1" \
    -X DELETE
```

APPENDIX A

ERROR CODES

200 OK
400 Generic error
401 Authentication required
403 Access to this object is denied.
404 Object not found
405 Used HTTP method not allowed for this object
409 The change is in conflict with the state of this resource
1000 XML document doesn't start with element:
1001 XML Data error
1002 Unknown element
1003 A mandatory element was missing
1004 Element is read only
1006 Element that may only appear once was sent multiple times
1008 Data format doesn't match type
1009 Typeerror. The data sent doesn't match the tag type
1010 Malformed URI
1011 Need title/name to generate seourl
1012 The SEOURL element is not allowed if 'generate_seo_url' is used
1013 Illegal tax value
1014 Campaign support is not active in this shop
1015 Symbol is in use.
1016 This language is not configured in shop:
1017 Lock/key index out of range
1018 Lock/key index mismatch
2001 A brand with this ID already exists
2002 A brand with this SEO URL already exists

2004 A campaign with this ID already exists
2005 This category cannot be deleted since it contains subcategories.
2006 This category cannot be deleted since it contains products.
2007 A category with that ID already exists
2009 Parent category uri doesn't refer to an existing category!
2010 Category cannot be parent of itself!
2011 One-time customers cannot be modified!
2012 Login cannot be empty when creating a customer.
2014 Unknown customer login or wrong password
2016 A customercategory with that ID already exists
2018 No such invoice
2019 Order reference required for source 'ORDER'
2020 Invoice reference required for source 'INVOICE'
2021 Invalid invoice source reference
2022 Unknown invoice source
2023 No such order
2024 The required field 'State' is missing
2025 'State' element not allowed since this is a system state
2026 There is already an order state with this ID
2027 This order state cannot be deleted since it is required by the system!
2028 The shop uses pricelists for conversion. You can't use CurrencyConversionRate.
2029 A pricelist with that ID already exists
2030 No such product
2031 A product with that SEO URL already exists
2032 Too many products/variants
2033 You can't set CurrencyConversionRate without a currency.
2034 Shop settings don't allow products without a main category
2035 No such campaign
2036 Both Product and Variant cannot be set in ProductLinks
2037 That SKU already exists
2038 Unit cannot be null
2039 This file is in use by some product and can therefore not be deleted
2040 A stockprofile with that ID already exists
2041 No such subscription
2042 A warehouse with that ID already exists
2043 Duplicate login id.
2044 Illegal characters in SKU
2045 Unknown payment_action:
2046 Payment has already been captured
2047 Payment has already been cancelled

2048 Payment method doesn't support capture/cancel.
2049 Failure communicating with the payment processor:
2050 An affiliate program with this ID already exists
2051 An affiliate program member with this ID already exists
2052 An affiliate program member with this activator already exists
2053 A symbol with this ID already exists
2055 Order is not editable
2056 No such transporter
2057 Invalid State
2058 A product with this ID already exists.
2059 A category with this SEO URL already exists.
3001 Picture uri doesn't refer to an existing picture
3002 The uri does not identify an existing document
3003 Brand uri doesn't refer to an existing brand
3004 Bad pricelist reference
3005 Warehouse URI doesn't point to existing warehouse
3006 Bad customer category reference
3007 uri doesn't identify an existing discount group
3008 URI doesn't identify an existing product variant
3009 URI doesn't refer to an existing category
3010 URI doesn't reference an existing product
3011 URI doesn't identify an existing variant:
3012 Specified view does not exist
3013 Unit uri doesn't reference an existing unit
3014 Stock profile URI doesn't identify a correct stock profile
3015 Downloadable uri doesn't refer to an existing downloadable:
3016 URI doesn't reference an existing variant
3017 Bad customer reference / customer doesn't exist.
3018 Bad affiliate
3019 The uri does not identify an existing symbol
3020 No such delivery address
3021 Attribute value (1) is not a value for attribute (2)
3022 Attribute/Value does not exist:
3023 Value ordering cannot be specified on create since no values exist yet
3024 Bad payment method reference
3025 Bad shipping method reference
3026 Bad Mode.
3027 No such discount group
4001 Shop doesn't support stock handling.
4002 Shop doesn't support Variant-pictures

4003 Shop doesn't support stock
4005 Shop doesn't support multiple price lists
4006 Shop doesn't support quantity prices
4007 Shop doesn't support multiple warehouses.
4008 Advanced campaign support required
4009 Shop doesn't support variant grouping
4010 Shop doesn't support order editing
5000 Filter syntax error at:
5001 number format error:
5002 Filter parenthesis mismatch
5003 Not a filterable field:

APPENDIX B

CHANGES

Version 1.0 (2009-05-25)
First public version.

Version 1.1 (2009-06-05)

- Added SEOURL for Product, Category and Brand.
- Documented order items in section 3.3.2.

Version 1.2 (2009-06-25)

- Added customer login/password (auth) check, see section 3.2.7.
- Changed XML definition of NULLs, these now SHOULD contain a “xsi:nil="true"” attribute. See section 2.14 for more information.

Version 1.3 (2009-08-10)

- Some elements were incorrectly marked as “str” when they should have been “empty”.
- Added the xsi namespace to xml objects, since some xml-parsers don’t like it when they see attributes for undefined namespaces.
- Specified order of returned objects when using client-based sync. See section 2.11.1.
- Added shipping address to Order object in section 3.3.2.
- Explained connection between customer login and email address in section 3.2.2.
- Added “ccode” datatype, section 2.14.2.
- Added “emailaddr” datatype, section 2.14.2.

Version 1.4 (2009-09-14)

- Added customer token/autologin procedure. See section 3.2.8.
- Added euroshop option to Root Options object.

Version 1.5 (2009-09-29)

- Added order state filter to order collections.
- Added Customer element to order collection.
- Documented Customer element in order object. See section 3.3.2.

Version 1.6 (2009-10-21)

- Added "date" type. See section 2.14.2.
- Added invoice API. See section 3.21.

Version 1.7 (2009-10-30)

- Added support for one-time customers. See section 3.2.
- Documented "StockProfile" in variant. See section 3.7.4.

Version 1.8 (2010-02-01)

- Added Discountcode mode "FIXEDSUMCOUNTDOWN".
- Added Discountcode field "LimitToOrderSum".
- Added "SimpleSizes" field in section 3.7.4.
- Added "FixedDiscountIncludesTax" field in section 3.20.2
- Added "product_with_sku" Product search parameter. See section 3.7.1.
- Added order states. See section 3.3.5.
- Added payment methods. (Section 3.4)
- Added shipping methods. (Section 3.5)
- Added subscriptions. (Section 3.22)
- Added *Product* link to *Variant* object. (Section 3.7.4)

Version 1.9 (2010-03-01)

- Qty-prices setting in Product. (Section 3.7.2).
- *ExportType* and *VatNo* added to Order object. (Section 3.3.2)

Version 1.10 (2010-04-12)

- Added *Tax* to Product. (Section 3.7.2).
- Added query parameter *generate_seo_url* to Product. (Section 3.7.2)
- Added query parameter *generate_seo_url* to Category. (Section 3.10)
- Added query parameter *generate_seo_url* to Brand. (Section 3.14)
- Added *Pricelist* to Order data. (Section 3.3.2)
- Added currency conversion rate to Order data. (Section 3.3.2)
- Added Campaign support. (Section 3.8)
- Added *Campaign* to Product. (Section 3.7.2)
- Added numeric error codes. See Section 2.9 and Appendix A.

Version 1.11 (2010-04-26)

- Customer collection *view=long* extended to provide complete Customer objects. (Section 3.2.1)
- Product collection *view=long* extended to provide complete Product objects. (Section 3.7.1)
- Variant collection *view=long* extended to provide complete Variant objects. (Section 3.7.3)

- Order collection *view=long* extended to provide complete Order objects. (Section 3.3.1)

Version 1.12 (2010-06-01)

- Added *Variant* element to order items. (Section 3.3.2)

Version 1.13 (2010-07-27)

- Added *ShippingCellPhone* to order object. (Section 3.3.2)
- Added *Barcode* to variant object. (Section 3.7.4)

Version 1.14 (2011-01-31)

- Added *RenewalPrice* to variant price data. (Section 3.7.4)

Version 1.15 (2011-04-11)

- Added *Currency* and *CurrencyConversionRate* to invoice. (Section 3.21.2)
- Added *RequestedDeliveryDate* and *PartialShipmentOK* to order. (Section 3.3.2)

Version 1.16 (2011-06-13)

- Added HTTP responce code 409 to PUT / Update requests. (Section 2.8.3)
- Added commands for capture/cancel of order payments. (Section 3.3.4)
- Added affiliate program support. (Section 3.23).
- Added information about which affiliate a customer arrived via. (Section 3.2.2).

Version 1.17 (2011-07-04)

- Added support for variant grouping. (Section 3.7.2).

Version 2 (2011-08-01) *Version numbering system changed to one nondecreasing integer.*

- Added 409-Conflict error code to DELETE. (Section 2.8.4).
- Added support for Symbols. (Section 3.24).
- Added support for setting symbols on products. (Section 3.7.2)
- Clarified that URIs provided by the API are not to be parsed but used as opaque strings by the client. (Section 2.3)
- Added “product_id” attribute to Product. (Section 3.7.2)
- Added “category_id” attribute to Category. (Section 3.10.2)
- Added “order_id” attribute to Order. (Section 3.3.2)
- Added “brand_id” attribute to Brand. (Section 3.14.2)
- Added “feature_id” attribute to Symbol. (Section 3.24.2)
- Added “customer_id” attribute to Customer. (Section 3.2.2)
- Added “variant_id” attribute to Variant. (Section 3.7.4)
- Clarified DELETE a bit. (Section 2.8.4)

Version 3 (2011-08-10)

- Added *NextRenewalDate* to Subscriptions. (Section 3.22.2)

Version 4 (2012-01-04)

- Clarified description of “Description” fields noting that they may contain HTML code.
- Added *ID* to Unit. (Section 3.15.2)
- Added *PictureURL* to Pictures. (Section 3.9.2)

Version 5 (2012-02-01)

- Added ExtraText1-6 and ExtraSelect1-6 in Order. (Section 3.3.2)
- Added ExtraText1-6 in OrderItem. (Section 3.3.2)
- Added Firstname/Lastname elements to order. (Section 3.3.2)

Version 6 (2012-03-08)

- Added language info to order. (Section 3.3.2)
- Added language to Customer. (Section 3.2.2)
- Added language support to API. (Section 2.6)
- Extended payment methods with language information. (Section 3.4)
- Extended shipping methods with language information. (Section 3.5)
- Extended stock profiles with language information. (Section 3.16.2)
- Extended pricelists with language information. (Section 3.13.2)
- Added *State* to Customer billing address. (Section 3.2.2)
- Added *BoxQty* to Variant. (Section 3.7.4)

Version 7 (2012-03-15)

- Added delivery addresses to customer. (Section 3.2.5)
- *PreferredDeliveryAddress* and *DeliveryAddresses* added to customer. (Section 3.2.2)

Version 8 (2012-04-02)

- Added support for custom attributes. (Section 3.11)
- Added custom attribute grouping to Product. (Section 3.7.2)
- Added custom attribute settings to Variant. (Section 3.7.4)

Version 9 (2012-05-04)

- Added API for getting configured Transporters. (Section 3.6)
- Added Shipment information to Order API. (Section 3.3.3)
- Added “Mode” to Customer. (Section 3.2.2)

Version 10 (2012-05-14)

- Added weight and volume to order data. (Section 3.3.2)
- Added *Iseditable* field to order (Section 3.3.2)
- Added *LineNumber* field to order items (Section 3.3.2)
- Added order creation API (Section 3.3)
- Added order editing API (Section 3.3)
- Added “OrderEditing” parameter to options object. (Section 3.1)

Version 11 (2012-10-05)

- Added *CustomAttributes* to Product. (Section 3.7.2)
- Added *Discountgroup* to Variant. (Section 3.7.4)
- Added more examples. (Chapter 4)
- Added “IsBulky” to Variant and Order data. See Section 3.7.4 and Section 3.3.2
- Added “ExtraText13” - “ExtraText20” to Variant. (Section 3.7.4)

Version 12 (2012-11-01)

- Implemented filtering support. (Section 2.10)
- Added filtering to Order. (Section 3.3.2)
- Added filtering to Customer. (Section 3.2.2)
- Added filtering to Invoice. (Section 3.21.2)
- Added filtering to Product. (Section 3.7.2)

Version 13 (2012-12-24)

- Automatic customer-number allocation using query parameter “allocate_customer_no”. (Section 3.2.1)
- Added “ID” to CustomAttribute. (Section 3.11.2)
- Added filtering to Subscription. (Section 3.22.2)
- Added filtering to Category. (Section 3.10.2)
- Removed “changed_at_or_after” and “changed_after” as they have been replaced by the filter system. (Section 2.10, 2.11.1)
- Added *CustomAttributes* to Customer. (Section 3.2.2)
- Added “ExtraText1” - “ExtraText4”, “ExtraSelector1” - “ExtraSelector4” to linked products in the Product object. (Section 3.7.2)
- Added “ManufacturerSKU” to Order lines. (Section 3.3.2)

Version 14 (2013-03-18)

- Added tax percentage data to Orders and order lines. (Section 3.3.2)
- Added tax percentage data to Invoice. (Section 3.21.2)

Version 15 (2013-04-01)

- Extended OrderStates with language information. (Section 3.3.5)
- Extended Campaign with language information. (Section 3.8)

Version 16 (2013-05-06)

- Added language support to brands (Section 3.14)
- Added language support to units (Section 3.15)
- Added Tag to attribute values. (Section 3.11). This also affects the CustomAttribute elements in section 3.2.2 and section 3.7.2.

Version 17 (2013-08-07)

- Added language support to categories (Section 3.10)
- Added language support to brand SEOURL (Section 3.14)

Version 18 (2013-09-09)

- Added Customer org. number to Order (Section 3.3.2)
- Added PaymentRefCapture to Order (Section 3.3.2)

Version 19 (2014-02-18)

- Added Language to Invoice (Section 3.21.2)

Version 20 (2014-06-12)

- Added extrafield configuration resources. (Section 3.25)
- Added extrafield resource for customers. (Section 3.2.8)
- Added extrafield resource for orders. (Section 3.3.7)
- Added extrafield resource for categories. (Section 3.10.2)
- Added extrafield resource for products. (Section 3.7.2)
- Added extrafield resource for product variants. (Section 3.7.4)
- Added extrafield resource for brands. (Section 3.14.2)
- Extended Discountcodes with more settings and information about allowed products. (Section 3.20.2)

Version 21 (2014-08-11)

- Added parameter `keep_pricelists` to variant updating. (Section 3.7.4)
- Added "IconPicture" to category. (Section 3.10.2)
- Added "CaptureTime" and "CancelTime" to order object. (Section 3.3.2)

Version 22 (2014-08-22)

- Added Language support for products (Section 3.7.2)
- Added URL and tooltip to product symbols (Section 3.7.2)
- Added Language support for variants (Section 3.7.4)
- Added Language support for Category state field (Section 3.10.2)
- Added list of supported languages to Root object (Section 3.1)
- Added "primary-language" attribute to language supported elements. (Section 2.6)
- Updated the language support section. Any API user that wishes to implement language support or already has SHOULD read it. (Section 2.6)
- Added API user information to root object. (Section 3.1)

Version 23 (2014-12-03)

- Added *handle_stock* query parameter to order creation. (Section 3.3)
- Added *recalc_ordersum* query parameter to order creation. (Section 3.3)

Version 24 (2015-04-15)

- Added *ProductOrVariantChangedTime* filter field to product collection. (Section 3.7.1)
- Added *ShipmentChangedTime* filter field to order collection. (Section 3.3.1)
- Made *CreatedTime* writable for orders marked as editable. (Section 3.3.2)

Version 25 (2016-02-22)

- Added “CreatedTime” and “ChangedTime” to ProductPicture. (Section 3.7.2)
- Added logic to better handle “CurrencyConversionRate”. (Section 3.3.2)

Version 26 (2016-05-02)

- Added “CreatedTime” and “ChangedTime” to Picture. (Section 3.9.2)

Version 27 (2016-08-25)

- Added “AnnualWorth” to Subscriptions. (Section 3.22, Section 3.22.2)
- Added “view=long” flag to Custom Attributes collection to allow all attribute data to be downloaded in one request. (Section 3.11)
- Added expanded attribute values to attribute object. (Section 3.11.2)

Version 28 (2017-04-10)

- Added “Unit” to OrderItems. (Section 3.3.2)

Version 29 (2017-10-09)

- Added “State” to Campaign. (Section 3.8)

Version 30 (2018-02-16)

- Added “Currency” to Subscriptions. (Section 3.22)

Version 31 (2022-02-14)

- Updated description on real values. (Section 2.14.2)
- Added description of meta data. (Section 2.15)
- Added opts “OSS” and “Voec” to Root object (Section 3.1)
- Updated “ExportType” for Order (Section 3.3.2)
- Added “InputFields” to OrderItems. (Section 3.3.2)
- Added Variant Collection Direct Access. (Section 3.7.5)
- Added Variant Direct Access. (Section 3.7.6)
- Added “RemoteLink” to Picture. (Section 3.9.2)
- Added Picture Bucket Data. (Section 3.9.5)
- Updated error codes (Section A)

Version 32 (2022-03-28)

- Added “Keys” to Customer. (Section 3.2.2)
- Added “Keys” to Customer category. (Section 3.12.2)
- Added “Locks” to Product. (Section 3.7.2)
- Added “Locks” to Category. (Section 3.10.2)
- Added “Locks” to ShippingMethod. (Section 3.5.2)
- Added “Locks” to PaymentMethod. (Section 3.4.2)
- Added “Locks” to Document. (Section 3.19.2)
- Added new section “Locks and keys”. (Section 3.26)
- Updated error codes (Section A)
- Corrected a typo for the “ExtraSelect” fields on the Order object (Section 3.3.2)

- Updated properties and actions for PaymentMethod object (Section 3.4.2)
- Updated properties and actions for ShippingMethod object (Section 3.5.2)

BIBLIOGRAPHY

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels
http://www.rfc-editor.org/rfc/rfc2119.txt

[RFC2616] Hypertext Transfer Protocol — HTTP/1.1
http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC2617] HTTP Authentication: Basic and Digest Access Authentication
http://www.rfc-editor.org/rfc/rfc2617.txt

[RFC3986] Uniform Resource Identifier (URI): Generic Syntax
http://www.rfc-editor.org/rfc/rfc3986.txt

[XML1.0] Extensible Markup Language (XML) 1.0
http://www.w3.org/TR/2008/REC-xml-20081126/

[XML Sch. P.2] XML Schema Part 2: Datatypes
http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[REST at Wikipedia] Representational State Transfer
http://en.wikipedia.org/wiki/Representational_State_Transfer

[cURL Homepage] cURL and libcurl
http://curl.haxx.se

[ISO-8859-1] ISO 8859-1 character set
http://en.wikipedia.org/wiki/ISO/IEC_8859-1

[ISO-639-1] Codes for the representation of names of languages
http://en.wikipedia.org/wiki/ISO_639-1

[ISO-3166-1] ISO standard Two-letter country codes
http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2