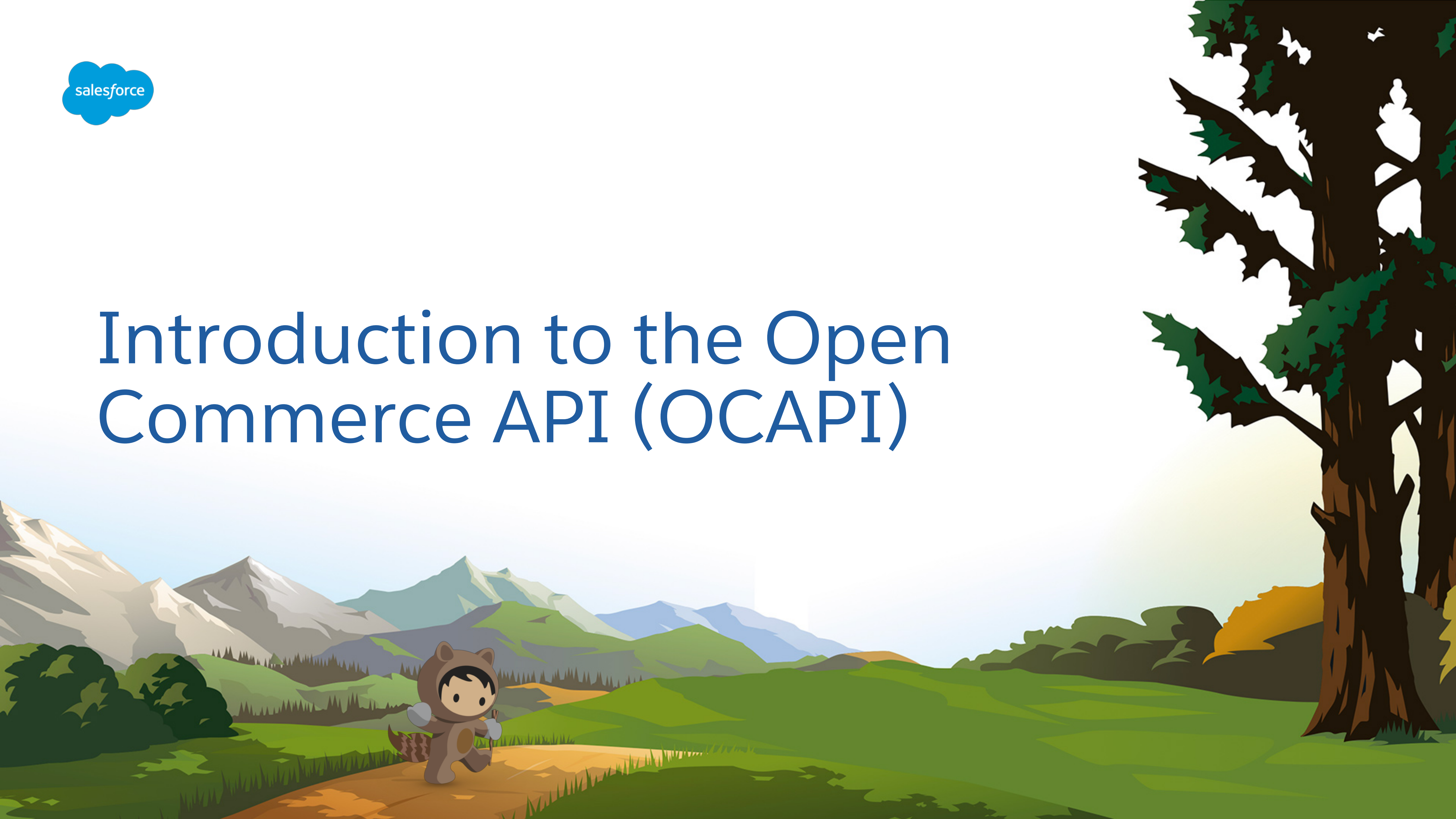
# Introduction to the Open Commerce API (OCAPI)



# Introduction to OCAPI

## What is OCAPI?

RESTful API that consists of three components APIs:

1. Shop API – used for storefront-like interactions (e.g. from a mobile app)
2. Data API – used for server-to-server integrations (e.g. OMS pushing order status)
3. Meta API – used for getting information about the available OCAPI resources

## Sample usages of the OCAPI APIs

1. Endless Aisle (Shop API)
2. Customer Service Center (Shop API)
3. Pinterest Buyable Pins (Shop API)
4. Mobile shopping applications (Shop API)

1. OCAPI Explorer (all APIs)
2. Business Manager (Data API)
3. CMS/ERP/PIM/CRM/etc. server-to-server integrations (Data API)
4. Continuous Integration (Data API)

## OCAPI Resources

## OCAPI Resources – base URL syntax

Production base URL: https://*host*/dw/*api_type*/

Other instances base URL: https://*instance-client*.demandware.net/s/*site_id*/dw/*api_type*/

*api_type* is **shop**, **data** or **meta** depending on the API used.

*host* is the production site host name

*instance-client* is usually the subdomain for the staging, development or sandbox instance

*site_id* is the identifier of the site.

## OCAPI Resources – extended URL syntax

OCAPI version identifier format: v*YEAR_RELEASE*, e.g. v18_7 (year 2018, 7th release)

Possible URL patterns for the OCAPI resources

**base_url/version_id/resource_type** – access multiple resources

**base_url/version_id/resource_type/identifier** – access individual resource

**base_url/version_id/resource_type/action** – execute a specific action for the resource – e.g. password reset

**base_url/version_id/resource_type/identifier/relationship_type** – nested resources

**base_url/version_id/resource_type/identifier/relationship_type/relationship_type_id**

**base_url/version_id/resource_type/identifier/relationship_type/action**

# OCAPI Resources Documentation

Shop API resources

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/ShopAPIResources.html

Data API resources

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/DataAPIResources.html

Each resource is documented with method, request and response document formats

Special Data API resources (system jobs – useful for CI implementations)
https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/SystemJobs.html

## HTTP Methods available

1. **GET** (s) – retrieve resource data
2. **HEAD** (s) – returns headers only (same as GET request), but not resource data
3. **DELETE** (i) – remove existing resource(s)
4. **PUT** (i) – create, update or replace a resource (not available on PRD and STG instances)
5. **PATCH** – partial resource update (unlike PUT touches only what is provided as data)
6. **POST** – create new resource with id provided from the server, override an HTTP method, execute special actions like password reset requests
7. **OPTIONS** (s) – return list of available operations for a resource (in header "Allow")

## HTTP Method Override

This is dedicated for situations where you have only POST and GET methods available, and you need to use some of the other methods.

There are two ways for HTTP method override:

1. Using URL parameter "method“
2. Using HTTP header "x-dw-http-method-override"

If both are provided: URL parameter takes precedence over the HTTP header

## JSONP requests

It is possible to obtain the data from an OCAPI call result as a JSONP script.

For this the URL parameter "callback" is used. When it is provided to a Shop API request, the result JSON is wrapped in a function call with the provided name of the function.

Due to JSONP calls being on the client-side, for security reasons it is recommended to use it only with Shop API calls. The Data API calls usually require credentials, which would be insecure to be stored on the client. More information about OCAPI client identification is given in the following slides.

## OCAPI HTTP success status codes

Successful execution

1. 200 (OK) - GET, PUT, or POST successfully completed
2. 201 (Created) - POST or PUT request successfully created a new resource or relationship resource
3. 204 (No Content) - DELETE, HEAD, or OPTIONS (or less typically a POST, PUT, or PATCH) request successfully completed and returned no content.

## OCAPI HTTP faults (error status codes)

Failed execution

1. 400 (Bad Request) - Malformed parameters, header values, or body
2. 401 (Unauthorized) – No permissions or no authentication provided
3. 403 (Forbidden) – No permissions
4. 404 (Not Found) - Resource does not exist
5. 405 (Method Not Allowed) - Resource does not support supplied HTTP method.
6. 409 (Conflict) – Conflict between current state and the operation
7. 412 (Precondition Failed) - PATCH request provided an outdated last-known base point, which means the resource was changed on server, possibly by a concurrent request.
8. 415 (Unsupported Media Type) – provided in "format" parameter or "Accept" header
9. 500 (Internal Server Error)

## OCAPI HTTP faults (continued)

Each unsuccessful OCAPI call will also produce a fault document in response, that will give more details about the fault and the related exception.

More information is found in the following documentation articles:

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/HttpStatusCodesAndFaults.html

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/Exceptions.html

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/OptimisticLocking.html

## OCAPI Resources and Localization

Shop API resources are served in one locale with fallback to country and default locale.

Parameter “locale” or Header “Accept-Language” or default locale for the site

Provided in format *<language code>-<country code>*

Data API support multiple locales and no locale fallback is performed.

Parameter “locale” or Header “Accept-Language” can be used to limit to one locale.

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/Localization.html

# OCAPI authentication

# OCAPI client application identification

Each OCAPI client needs to identify itself using a client ID.

The client ID can be obtained from Account Manager: https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/AccountManager/AccountManagerAddAPIClientID.html

The client ID is to be included in every request. Here are ways to provide it in order of precedence:

1. Header "Authorization:Bearer -token-“. Client ID is determined by provided -token- string
2. Request parameter client_id
3. Header "x-dw-client-id:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa" where 30a’s is the client ID.

The 30a’s client ID is a default available only on sandbox instances.

No production code should use the 30a’s client ID.

## JSON Web Token (JWT)

Authentication mechanism required by several Shop API resources

JWT is always obtained in the context of a customer (using /customers/auth resource)

Requires application client ID (30a's, or obtained from Account Manager – see previous slide)

Can login as guest customer, using "type": "guest“

Can login as a registered customer, using "type": "credentials"
(customer login and password provided using Basic Authentication scheme)

Token expires after 30 minutes, and can be replaced with a new token using "type": "refresh".

Response contains Authorization:Bearer -token- header that is needed for subsequent requests.

## OCAPI OAuth 2.0

Used in cases where JWT does not fit:

1. Data API for server-to-server communication (client credentials grant)
2. Shop API or Data API when Business Manager user interacts with the system (Business Manager user grant)

Steps of using Oauth 2.0

1. Register client application using Account Manager (client_id and password).
   (The password for the system 30a account is the same as the account name)
2. Request an access token, using the corresponding grant
3. Include the access token in the subsequent OCAPI requests

# OAuth 2.0 - client credentials grant

Make sure request is done over HTTPS/TLS!

```
POST /dw/oauth2/access_token HTTP/1.1

Host: account.demandware.com

Authorization: Basic Y2xpZW50X2lkOnBhc3N3b3Jk

Content-Type: application/x-www-form-urlencoded

Body:

grant_type=client_credentials
```

The Authorization:Basic header contains the client id and password separated by colon in Base64 encoding (*client_id:password*, base64 encoded)

# OAuth 2.0 - obtaining Business Manager user grant

Required to operate as a BM user

Make sure request is done over HTTPS/TLS!

```
POST /dw/oauth2/access_token?client_id=aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa HTTP/1.1

Host: insance-client.demandware.net

Authorization: Basic Qk11c2VyOkJNcGFzc3dvcmQ6Y2xpZW50X3Bhc3N3b3Jk

Content-Type: application/x-www-form-urlencoded

Body:

grant_type=urn:demandware:params:oauth:grant-type:client-id:dwsid:dwsecuretoken
```

The Authorization:Basic header contains the Business Manager user and password, followed by Account Manager client password separated by colon in Base64 encoding (*BMuser:BMpassword:clientid_password*, base64 encoded)

## OCAPI OAuth 2.0 Response

Response looks something like the following

```
{
"access_token": "-token-",
"expires_in":899,
"token_type":"Bearer“
}
```

Client credentials grant type tokens expire in 30 minutes (expires_in 1799 seconds)

Business Manager user grant type tokens expire in 15 minutes (expires_in 899 seconds)

In subsequent OCAPI calls, you need to take the value of access_token and construct a header "Authorization: Bearer -token-" with it.

## OCAPI Configuration

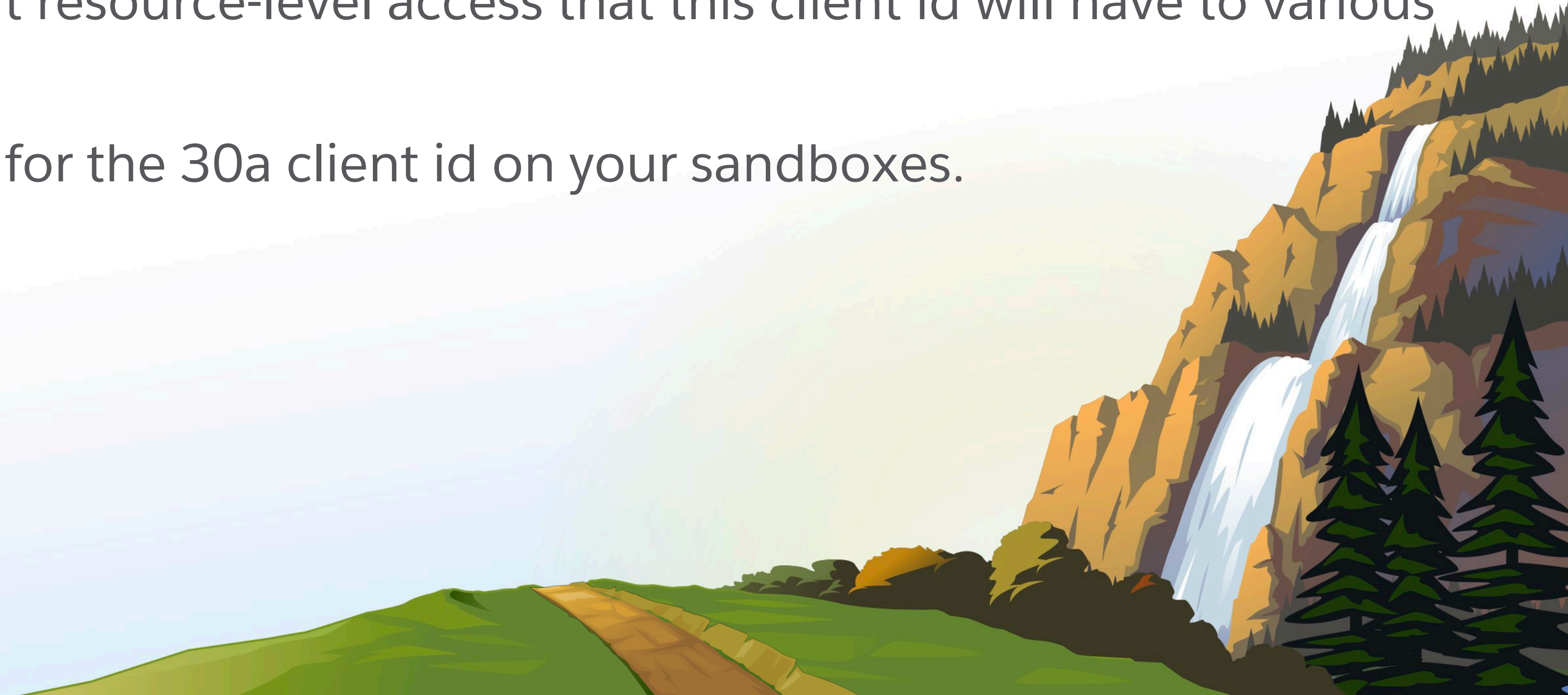
## Client ID

Client ID is needed for all OCAPI calls and is configured in Account Manager

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/AccountManager/AccountManagerAddAPIClientID.html

Once client ID is configured and enabled in Account Manager, it is needed to do corresponding configuration about resource-level access that this client id will have to various OCAPI resources.

Such configuration is also needed for the 30a client id on your sandboxes.



## OCAPI Settings

Control OCAPI client id permissions and also the caching of various OCAPI resources.

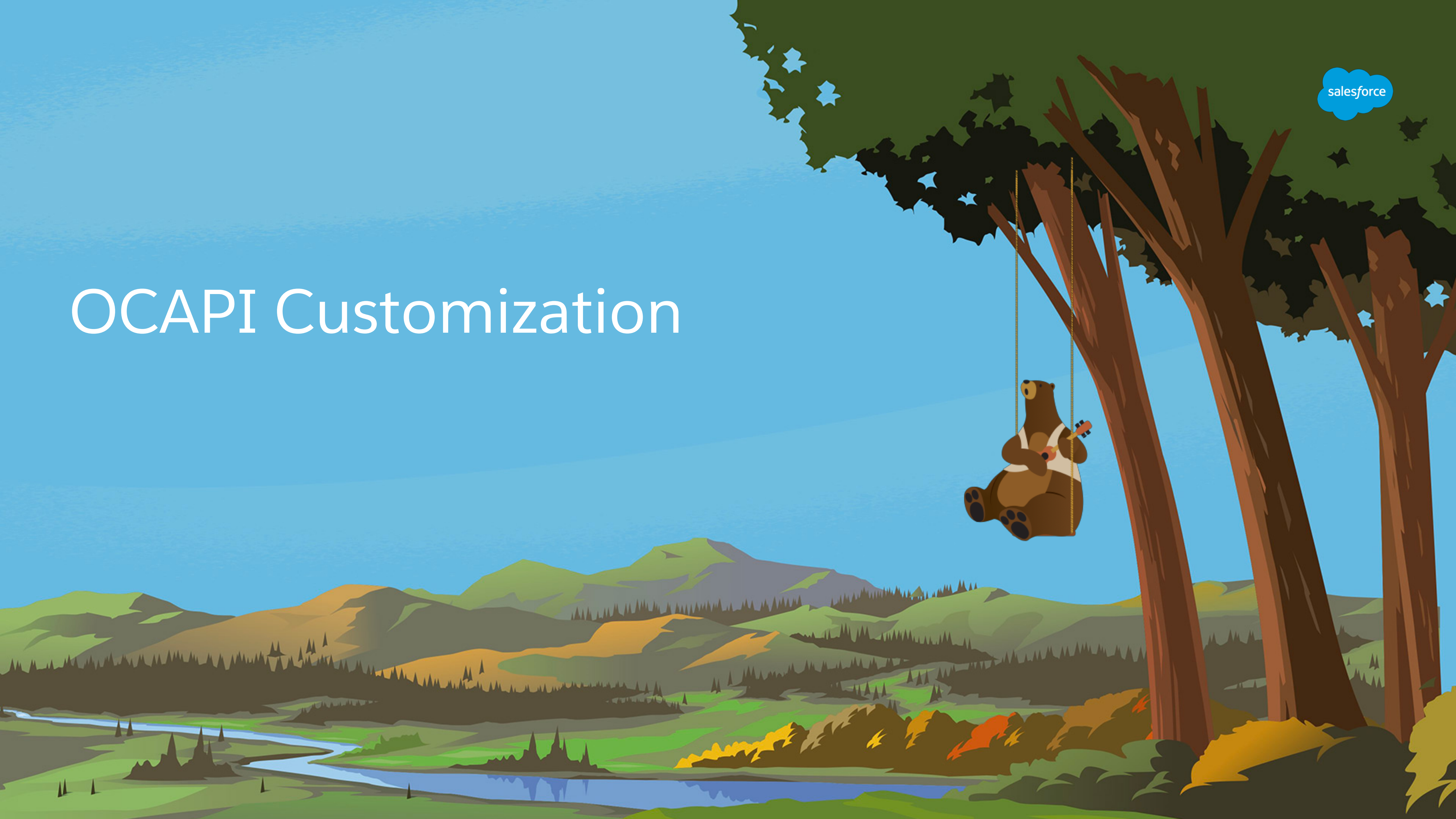
Done from Administration > Site Development > Open Commerce API Settings

The settings start with JSON document format version identifier, it is the format of the configuration, it does NOT limit the calls to that specific version.

More information about the configuration is available in: https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/OCAPISettings.html

It is possible to limit the resource properties visible for a particular client: https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/PropertySelection.html

## OCAPI Customization



## OCAPI hooks

Server-side script functions, typically called **before** and **after** the HTTP method execution.

Some are used only to **modify the response** from the method execution.

For each resource, list of hooks is provided in section named "Customization“.

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/Hooks.html - full list of hooks

Defined in hooks.json, referenced by package.json (used for cartridge initialization)

Special case: validation hooks for adding OCAPI flashes (validation error messages) https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/Flash.html

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/Customization.html for more details.

# Custom properties

Marked with a prefix "c_“

Available only if request or response document are marked as supporting custom properties

Can be introduced by hooks

All of the standard attribute types

Limitations for using HTML and Image types only for output

https://documentation.b2c.commercecloud.salesforce.com/DOC1/topic/com.demandware.dochelp/OCAPI/19.1/usage/CustomProperties.html

## Customization using controllers

Complex functionalities may not be possible to easily implement with OCAPI and hooks.

In some cases it may make more sense to implement logic as a custom controller.

Preferred approach is to implement the customization with hooks, whenever possible.

# THANK YOU