Importing data into Delogue

Version 1.3

—

Delogue.

Table of Content:

1. INTRODUCTION

This document explains how to import data into the Delogue system, which covers API requests and responses. These APIs are available exclusively to Delogue clients and are limited to users with approved credentials.

Delogue exposes REST-based services to import data into Delogue from an external system. To call any API, the client needs to authenticate himself by calling the authentication endpoint. The authentication endpoint will return a token that the client has to use for subsequent requests sending data to Delogue. We will explain the details of each request-response in the following sections of this document.

2. PREREQUISITE

For using these APIs, the organization admin **should create a user who only has access to this organization to avoid data from being created in an**other company. In Delogue, a user can work with multiple companies as a supplier or a designer.

3. AUTHENTICATION

Importing data in Delogue is a 2-step process:

1) Getting authentication token first
2) Using authentication token calling service

The endpoint of authentication service services is https://my.delogue.com/auth/token
While calling these API, provide user credentials (new user credentials as mentioned in the prerequisite section) using the post method shown in the below

API calls in Delogue require an authorization token which can be generated using this endpoint. This should be used in subsequent calls in the Authorization header (Example: “bearer XXXXXxxXX-99”)
Method: POST
Domain: https://my.delogue.com
Path: /auth/token
Headers: Content-Type: application/x-www-form-urlencoded
Body as x-www-form-urlencoded:

grant_type	password
username	your username
password	your password

Example:

grant_type=password&username=test@example.com&password=yourpass

Success response as JSON:

access_token	Token to use in Authorization header in other endpoints
token_type	Token type to use in front of access_token in Authorization header
expires_in	Seconds to expiration

Example:


```
{
  "access_token":
"w4TSCp1S0lmBKz4CYQYSMvtjzBdVks4g1RhivcE2er636QdvGCC0jZCn0T8_G5smiNzqvSpK5TlyNNihxSl0cXzaHAmq
N-a3OAxuXWh6vYwahD3aUvMXXxXxXXXXXxXXuPVM7jPUC7GA8UzkrHHetMTfgTz490540954945845949 58459YYAB
UwBCG7rD4N6×7NOG1esQUWQribDX-1lEOdYgvKUqOnzsIDGNBSgCxxXXxxXXXXXXxxxLYBwyl4aM8ZLq-EiJXXxxxv_
2-nNTErkCfuAQsVhRl5JwZ81h_-ujBKs99X",
  "token_type": "bearer",
  "expires_in": 604799
}
```


Error response as JSON:

error	Error message

Example:


```
{
  "error": "The user name or password is incorrect."
}
```


example.

```
$.ajax({
    url: "https://my.delogue.com/auth/token",
    type: "POST",
    contentType: "application/json; charset=utf-8",
    dataType: 'json',
    cache: false,
```

```
        crossDomain: true,
        data: { "grant_type": "password", "username": "yourusername", "password":
"yourpassword" },
        success: function (data) {
                token = data.access_token;
        },
        error: function (x, y, z) {
                alert(x + '\n' + y + '\n' + z);
        }
});
```

Post JSON data format is: {"grant_type": "password", "username": "youruseremail", "password": "yourpassword"}

On success: String 'access_token' will be returned as shown in the above example. You have to use this access-token for the subsequent calls.
On error: If you send the wrong credentials, you will get an error message saying: "Bad Request".

4. IMPORTING COLORS

End point for importing Admin Colors in Delogue – https://my.delogue.com/Import/AdminColor/

The string token, the authentication endpoint had returned, should be used in these calls. This token should be sent as a 'Bearer' token with your JSON data. E.g. in jQuery call -

beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer ' + token); },

To import multiple colors in Admin color section, you need to send the data for the following fields: Name, ID, ID2, pantone, isActive, Action
The JSON object for color looks like this:

```
Colors = [{
        "Name": "Red",
        "Id": "test 1",
        "Pantone": "18-1447 tcx",
        "IsActive": "true",
        "Id2": "cc1",
        "action":""
},
{
```

```
        "Name": "Blue",
            "Id": "test 2",
            "Pantone": "18-1444 tcx",
            "IsActive": "true",
            "Id2": "cc3",
            "action":"delete"
}]
```

Add new color: You need to keep the "action" field empty in JSON, in order to create a new color. If you are using Id2, then the Id2 value must not be present in the Delogue database to create a new color.
Update: If passed JSON Id2 value matches with existing color then that matched record will be updated.
Delete color: You need to set the action field value to 'delete' in JSON. The web service will use the Id2 to find the color to delete. If the color is used in any style, the color will not be deleted, and an error message will be returned.

Validations while importing color:

Field	Validation	Action	Error message
Name	If this field is missing	Color is not created	Missing Color Name For record no: {#}
Id2	If multiple records found for Id2	Color will not be updated	Multiple records found for Id2 at record no: {#}
Id2+Action	If Action is "delete" and Id2 is Missing If Action is "delete" and Id2 does not exist in Delogue If Action is "delete" and multiple records found for Id2 in Delogue	Color will not be deleted Color will not delete Color will not delete	Id2 is required to delete Admin Color at record no: {#} Admin Color not found at record no: {#} Multiple records found for Id2 at record no: {#}

Example of call is:

```
 $.ajax({
        url: "http://my.delogue.com/Import/AdminColor/",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        dataType: 'json',
        data: json.stringify([
        {"Action": "Delete", "Id2": "2"},
        {"Name": "Black", "Id": "999", "Pantone": "19-4006", "IsActive": "true", "Id2": "1"},
        {"Name": "Green", "Id": "862", "Pantone": "18-1658", "IsActive": "true", "Id2": "2"},
```

```
        ]),
        cache: false,
        crossDomain: true,
        beforeSend: function (xhr, settings) {xhr.setRequestHeader('Authorization', 'Bearer ' +
token);},
        success: function (data) {
        alert('Data imported successfully');
        },
        error: function (x, y, z) {
        alert(x.responseText);
        }
})
```

On success: Call will return in success and you will receive the last created object.
On error: If any validation fails that details will be in the response text (last parameter of error function).

5. IMPORTING ITEM DATA TO DELOGUE

Endpoint for importing Items in Delogue – https://my.delogue.com/Import/ Items/
The authentication endpoint has returned your string token which should be used in these calls. This token should be sent as 'Bearer' token with your JSON data.
E.g. in jQuery call - beforeSend: function (xhr, settings) {xhr.setRequestHeader('Authorization', 'Bearer ' + token);},

5.1 ASSUMPTION

ERP systems will always send only one Item part. That's why Item part name and part composition details are part of the item header section.

5.2 FTP DETAILS

For each item, an image logo is associated which can be added to the Delogue system. For getting this image into Delogue, the client has to specify FTP location details in JSON. Format of JSON is shared below. These FTP details are common for all items, so there are specified ones at the top before the item collection.
FTP Details is used when you want to add a logo (thumbnail image) for the Item while you create or update the Item. It should contain these three fields:

FTPLocation: Provide your FTP location where the image is stored. e.g. "FTPLocation": "ftp.test.net/home/images"
UserId: Provide your ftp username.

Password: Provide your ftp password.

NOTE: Getting the file from FTP and linking it with the Item will be a time-consuming process. So, we will first process items (creation/updating) and then we will start the image linking process. If any error occurs while fetching an image from FTP, it will not affect item creation/updating. This means an item will be created even if the FTP operation fails.

5.3 ITEM HEADER

This will contain item information mentioned in below table:

Field Name in JSON	Delogue UI Name	Required (Mandatory)	Description
ErpItemId		Yes	UniqueId of item in ERP (external to Delogue) system
Brand	Brand	Yes	
CompanyContactPerson	Contact Person	Yes	
Name	Item name	Yes	
ItemNumber	Item no	No	
Logo	Item logo	No	Image name for Item logo present at provided ftp location
Description	Description	No	
SupplierId	Supplier Id field from Admin→Supplier tab	No	
Categories	Categories		
ItemPartName	Item Part Name	No	User can create item without Item Part.
ItemPartComposition	Item Part Composition	No	
Action		Yes, for update/delete	For creation of new item this field should be blank and for updating of existing system this should be "update" for deletion of existing system this should be "delete"

5.4 SIZES

Field Name in JSON	Delogue UI Name	Required (Mandatory)	Description

ErpSizeId		Yes	UniqueId of item size in erp (external to Delogue) system
Name	Size name/weight	Yes	

5.5 ITEM COLORS

Field Name in JSON	Delogue UI Name	Required (Mandatory)	Description
Id	Library color Id2	Yes	UniqueId of item color in erp (external to Delogue) system

5.6 EXAMPLE OF JSON

The JSON object for item is like:

```
{
        "FtpDetails": {
                "FTPLocation": "ftp://ftp.test.net/home/Images/",
                "UserId": "testUser",
                "password": "abcdefg"
        },
        "Items":
        [
                {
                        "ErpItemId": "23",
                        "Brand": "Adidas",
                        "CompanyContactPerson": "abc@gmail.com",
                        "Name": "button",
                        "ItemNumber": "b-1",
                        "Logo": "Itemlogo.jpg",
                        "Description": "square button",
                        "SupplierId": "7896",
                        "Categories": ["ItemCat1"],
                        "ItemPartName": "TestPart",
                        "ItemPartComposition": "TestComposition",
                        "Action": "",
                        "Sizes":
                         [
                                {"ErpSizeId": "11", "Name": "S"},
                                {"ErpSizeId": "22", "Name": "M"},
                        ],
```

```
                    "ItemColors":
                    [
                        {"id": "blue11"},
                        {"id": "red21"}
                    ]
                }
        ]
}
```

5.7 IMPORTING NEW ITEM

If you want to create a new item in Delogue, you have to send JSON in the above format and set the Action field value as an empty string for insertion. The value of field ErpItemId should be set as the unique id of your system. This value will be used in case of update/delete to find the correct record. Per assumption, ERP system will send only one-part details in fields named ItemPartName and ItemPartComposition of Item part, color cards will be added as per ItemColors data sent in JSON.

5.7.1 VALIDATION DETAILS FOR INSERT

Some validations are done while creating a new item in Delogue and these different validation details are mentioned below.

Field	Validation	Action	Error Message
Item Header			
ErpItemId	If this value is missing	Item will not be created, and further processing of record will be stopped	Missing ErpItemId value for record no {#}
ErpItemId + Action	If ErpItemId value already present in Delogue and Action is empty	Item will not be created, and further processing of record will be stopped	Item with ErpItemId already exist in Delogue for record no {#}
Brand	If this value is missing or wrong brand value	Item will not be created, and further processing of record will be stopped	Missing/Wrong Brand value for record no {#}
Name	If this value is missing	Item will not be created, and further processing of record will be stopped	Missing item name value for record no {#}
CompanyContact Person	If this value is missing or wrong Company Contact Person value	Item will not be created, and further processing of record will be stopped	Missing/Wrong Company Contact Person value for record no {#}
Brand + Name + ItemNumber +SupplierId	If for this combination item already exist in Delogue	Item will not be created, and further processing of record will be stopped	Wrong combination of Brand, Name, Item Number and Supplier Id value for record no {#}

Sizes			
ErpSizeId	If this value is missing	Item Size will not be created	Missing ErpSizeId value for record no {#}
ErpSizeId + Action	If ErpSizeId value already present in Delogue and Item header Action is empty	Item Size will not be created	Item with ErpSizeId already exist in Delogue for record no {#}
Name	If Name value already present in Delogue and Action is empty	Item Size will not be created	Item size with same name already exist in Delogue for record no {#}
Item Part			
ItemPartName	If this value is missing	Item Part will not be created, Item color will not be created	
Item Color	If this value is missing	No color cards for item will be created	

5.7.2 VALIDATION DETAILS FOR ITEM LOGO IMAGE

Field	Validation	Action	Error Message
FTPLocation UserId & Password	- If not able to connect FTP location with given details - If given credentials are wrong	Will not affect any item creation process.	- Not able to connect ftp site. - Not able to login FTP
Logo	Logo Image type		{#}: Invalid file type. Please select any of the following types only jpg,gif,png,jpeg,jpe,bmp,jfif,tif,tiff,dib for ErpItemId: {#}
	Logo Image size		{#}: Please select file less than 5MB for Item Logo for ErpItemId: {#}
	Logo Image name		{#}: Image Does not exist at FTP location for ErpItemId: {#}Image download Error!

5.7.3 UPDATE EXISTING ITEM

HEADER

To update existing items in Delogue, you have to send JSON data in a specified format and the Action field value should be set as "update". Passed 'ErpItemId' values will be used to find out old records in Delogue, and that record will be updated as per value sent in JSON.

SIZES

If Action is 'update' in the Item header level, first check will be done as if size with ErpSizeId is present in Delogue. If yes, that size will update. Otherwise, a new size will be created. If there are old sizes with no ErpSizeId present the old size will be deleted from item sizes.

ITEM PART

There will be only one item part sent, so the same will be updated with the new data sent. If an empty value is sent while updating, the item part will not be deleted, and no action will be performed for item color cards.

ITEM COLORS

If Action is “Update” in the Item header level, first check will be done as if a color card with Id is present in Delogue. If yes then the color card will be updated. Otherwise, a new color card will be created. If there are any old color cards with no Id, these color cards will be deleted from item colors.

VALIDATION DETAILS FOR UPDATE

Some validations are done while updating items in Delogue. The different validation details are mentioned below:

Field	Validation	Action	Error Message
Item Header			
ErpItemId	If this value is missing	Item will not be updated, and further processing of record will be stopped	Missing ErpItemId value for record no {#}
Action	If this value is not "update"	Item processing will not be treated as update for whole item. It could be insert if value is empty or Delete if value is "delete"	
Brand	If this value is changed compare to old value	Item will not be updated, and further processing of record will be stopped	Can't Change Brand value for record no {#}
Name	If this value is missing	Item will not be updated, and further processing of record will be stopped	Missing item name value for record no {#}

CompanyContact Person	If this value is missing or wrong Company Contact Person value	Item will not be updated, and further processing of record will be stopped	Missing/Wrong Company Contact Person value for record no {#}
SupplierId	For provided value if supplier company is not present in Delogue or supplier company is not active	Item will not be updated, and further processing of record will be stopped	Wrong Supplier Id value for record no {#}
Brand + Name + ItemNumber +SupplierId	If for this combination item already exist in Delogue	Item will not be updated, and further processing of record will be stopped	Wrong combination of Brand, Name, Item Number and Supplier Id value for record no {#}
Sizes - Pre-Condition – Action value for Item header should be "update"			
ErpSizeId	If this value is missing	Item Size will not be updated	Missing ErpSizeId value for record no {#}
Name	If Name value already present in Delogue and Action is empty	Item Size will not be created	Item size with same name already exist in Delogue for record no {#}
ItemColors Pre-Condition – Action value for Item Header should be "Update"	- If Multiple Item part exists for Item - If no Item Part exists for Item - If Id not found in Delogue	- Item Color will not be created or deleted - Item color will not be created or deleted - Item color will not be created or deleted	- Multiple item parts found for item at record no: {#}. - No item parts for item record no: {#} - Color Id ({#}) does not exists in Delogue at item record no: {#} and item color record no {#}.

Validation for logo image will be the same as mentioned in the item creation section.

DELETING ITEM

If Action is "delete" in the Item header level, the item will be marked as Inactive. An item is never deleted.

Note: In case of Update or Delete, it will not be possible to delete size, part and color that are used in existing Styles of Delogue. In this case, an appropriate error message will be sent.

RESULT OF IMPORT

On Success: Call will return in success and you will get an object which contain:

- Successfully created Item ErpItemIds
- ·Errors while importing item logo from FTP

On Error: If any validation fails you will get the details in the response text which will contain:

- Successfully created Item ErpItemIds
- Errors while importing item logo from FTP.
- Validation errors while importing item header, item size, item part, item color

6. IMPORTING STYLE DATA TO DELOGUE

This chapter explains how external systems can send style data to be imported to Delogue.

6.1 IMPORT STYLE (UPDATE NUMBER, NAME, STATE & CUSTOM FIELD DATA TO DELOGUE) BY STYLE NUMBER

Service API detail: Importing style number, name, state & custom field data (means only updating the existing style data) by style number in Delogue.

End point for this API: https://my.delogue.com/v1.0/import/styles/
For this API you can send input as a JSON or XML.
In above rest API URL section description is as below
v1.0 - current first 1.0 version
import – it's importing data in Delogue
styles- updating styles collection
HTTP method type – PATCH
The token from authentication should be sent as 'Bearer' token with your input data. E.g., in a jQuery call -
beforeSend: function (xhr, settings) {xhr.setRequestHeader('Authorization', 'Bearer ' + token);},

In this first version "v1.0" it will only support updating the style number, style name, style state, custom fields, Custom field per color.
Style will be identified by the style number in input JSON.

Input data is in the form of a collection of styles:

- StyleNumber. From which we can find the style within the Delogue system.
- Fields.
    i. NewStyleNumber (optional). new number for the current style.
    ii. NewStyleState (optional). The current updating style must be in published state & NewStyleState must be either Cancelled or Delivered state.
    iii. NewStyleName (optional)
    iv. UpdateMultipleStyles (Optional). This defines whether the values should added to all matching styles or give an error if it can find more than one style with the current style number. Value should be either true or false.
    v. Custom Fields (optional)
    vi. Custom Fields Per Color (optional)

Note:

- Custom field & Custom field per color input is one of the following types:
    - **Text** with text value
    - **Date** with date value in DD-MMM-YYYY format e.g., 31-OCT-2017
    - **Single Allowed Value**
- The old value will be deleted and replaced with the new value.
- ColorId and ColorName should be the exactly the same as added in admin color list with same case.

Required Headers:

Content-Type: application/json
Authorization: Bearer {authentication Token}
Accept: application/json

Custom field per color:

We have added support to import custom field and custom field per color allowed value by allowed value id.
User can send the id like following in JSON: {"Name": "abc", "Value": "xxx", "Id": "x01"}

- Users can send value / Id / both value & id to add allowed value.

Clear Content:

User can clear the custom field value of type text, date and allowed value.
We have added a boolean field "ClearContent" for clearing the value of a given custom field.

If the ClearContent field is set to true then we will clear the value of the custom field ignoring the value sent.
If the ClearContent field is set to false or missing (not included) then it will work the same like old behavior with all validation.
In json just send clearContent field like "ClearContent": true
In xml just send clearContent field like <ClearContent>true</ClearContent>

OPTION WITH JSON ⇒ EXAMPLE JSON DATA FOR STYLE FIELDS LIKE:


```
{ "ImportStyles": [
        {       "StyleNumber": "0444",
                "Fields" :  {
                                "NewStyleNumber": "0444",
                "NewStyleState": "Delivered",
                "NewStyleName": "0444",
                "UpdateMultipleStyles": true,
                                "CustomFields":
                                        [{ "Name": "Dessin", "Value": "xxxx02" },
                                        { "Name": "EAN", "Value": "EANxxx02", "ClearContent": true
},
                                        { "Name": "Expiry date CE", "Value": "30-OCT-2017",
"ClearContent": false },
                                        { "Name": "Washing instructions (UK)", "Id": "60", "Value":
""},
                                        { "Name": "Backend", "Id": "", "Value": "Yes"},
                                { "Name": "Web", "Value": "Yes", "ClearContent": true},
                                { "Name": "Pallevare", "Id": "N"},
                                { "Name": "Sex", "Id": "G", "Value": "Girl"},
                                { "Name": "Marketing media", "Id": "W", "Value": "Web"}],
                        "CustomFieldsPerColor": [{ "Name": "Dessin", "ColorId": "R1", "ColorName":
"Red", "Value": "abcd1" },
                                        { "Name": "Dessin", "ColorId": "G1", "ColorName": "Green",
"Value": "abcd2" },
                                        { "Name": "EAN", "ColorId": "R1", "ColorName": "Red",
"Value": "def1" },
                                        { "Name": "EAN", "ColorId": "G1", "ColorName": "Green",
"Value": "def2" },
                                        { "Name": "Expiry date CE", "ColorId": "R1", "ColorName":
"Red", "Value": "10-SEP-2019" },
```

```
                                        { "Name": "Expiry date CE", "ColorId": "G1", "ColorName":
"Green", "Value": "15-SEP-2019" },
{ "Name": "Backend", "ColorId": "G1", "ColorName": "Green", "Id": "", "Value": "No"},
                                        { "Name": "Web", "ColorId": "G1", "ColorName": "Green",
"Value": "No"},
                                        { "Name": "Pallevare", "ColorId": "G1", "ColorName": "Green",
"Id": "Y"},
                                        { "Name": "Sex", "ColorId": "G1", "ColorName": "Green", "Id":
"B", "Value": "Boy"}]
                                }
                        }
                ]
}
```

Example of input JSON data call:

```
$.ajax({
        url: "https://my.delogue.com/v1.0/import/styles/",
        type: "Patch",
        contentType: "application/json; charset=utf-8",
        dataType: 'json',
        data: JSON.stringify({ "ImportStyles": [
        {
                  "StyleNumber": "0444",
                "Fields" :  {
                                "NewStyleNumber": "0444",
                                "NewStyleState": "Delivered",
                        "NewStyleName": "0444",
                        "UpdateMultipleStyles": true,
                                "CustomFields":
                                        [{"Name": "Dessin", "Value": "xxxx02"},
                                        {"Name": "EAN", "Value": "EANxxx02", "ClearContent": true},
                                        {"Name": "Expiry date CE", "Value": "30-OCT-2017",
"ClearContent": false},
                                        {"Name": "Washing instructions (UK)", "Id": "60", "Value": ""},
                                {"Name": "Backend", "Id": "", "Value": "Yes"},
                                {"Name": "Web", "Value": "Yes", "ClearContent": true},
                                {"Name": "Pallevare", "Id": "N"},
                                        {"Name": "Sex", "Id": "G", "Value": "Girl"},
                                {"Name": "Marketing media", "Id": "W", "Value": "Web"}],
```

```
                    "CustomFieldsPerColor": [{"Name": "Dessin", "ColorId": "R1", "ColorName":
"Red", "Value": "abcd1"},
                                {"Name": "Dessin", "ColorId": "G1", "ColorName": "Green",
"Value": "abcd2"},
                                {"Name": "EAN", "ColorId": "R1", "ColorName": "Red",
"Value": "def1"},
                                {"Name": "EAN", "ColorId": "G1", "ColorName": "Green",
"Value": "def2"},
                                {"Name": "Expiry date CE", "ColorId": "R1", "ColorName":
"Red", "Value": "10-SEP-2019"},
                                { "Name": "Expiry date CE", "ColorId": "G1", "ColorName":
"Green", "Value": "15-SEP-2019"},
        { "Name": "Backend", "ColorId": "G1", "ColorName": "Green", "Id": "", "Value": "No"},
                                {"Name": "Web", "ColorId": "G1", "ColorName": "Green",
"Value": "No"},
                                {"Name": "Pallevare", "ColorId": "G1", "ColorName": "Green",
"Id": "Y"},
                                {"Name": "Sex", "ColorId": "G1", "ColorName": "Green", "Id":
"B", "Value": "Boy"}]
                        }
                }
                ]
        }),
        cache: false,
        crossDomain: true,
        beforeSend: function (xhr, settings) {xhr.setRequestHeader('Authorization', 'Bearer ' +
token);},
        success: function (data) {
        alert('Updated styles are follows:\n' + data.styleNumberList + '\n' + data.importErrors);
        },
        error: function (x, y, z) {
        alert('Error while importing style data\n' + x.responseText);
        }
 })
```

On success: Call will return in success ***with information of Updated style number list & Import errors***

On error: If API fails you get message in response text

6.1.1.2 OPTION WITH XML ⇒ EXAMPLE XML DATA FOR STYLE FIELDS LIKE:

```
<ImportStyleCommand>
  <ImportStyles>
      <ImportStyle>
          <StyleNumber>0444</StyleNumber>
          <Fields>
              <NewStyleNumber>0444</NewStyleNumber>
              <NewStyleName>0444</NewStyleName>
              <NewStyleState>Delivered</NewStyleState>
              <UpdateMultipleStyles>true</UpdateMultipleStyles>
              <CustomFields>
                  <ImportStyleCustomField>
                      <Name>Dessin</Name>
                      <Value>xxxx01</Value>
                  </ImportStyleCustomField>
                  <ImportStyleCustomField>
                      <Name>EAN</Name>
                      <Value>EANxxx01</Value>

                                          <ClearContent>true</ClearContent>
                  </ImportStyleCustomField>
                  <ImportStyleCustomField>
                      <Name>Washing instructions (UK)</Name>
                      <Value></Value>
                      <Id>60</Id>
                  </ImportStyleCustomField>
                  <ImportStyleCustomField>
                      <Name>Backend</Name>
                      <Value>Yes</Value>
                      <Id></Id>
                  </ImportStyleCustomField>
                  <ImportStyleCustomField>
                      <Name>Web</Name>
                      <Value>Yes</Value>
                  </ImportStyleCustomField>
                  <ImportStyleCustomField>
                      <Name>Pallevare</Name>
                      <Id>N</Id>
                  </ImportStyleCustomField>
                  <ImportStyleCustomField>
```

```
                <Name>Sex</Name>
                <Value>Girl</Value>
                <Id>G</Id>
            </ImportStyleCustomField>
            <ImportStyleCustomField>
                <Name>Marketing media</Name>
                <Value>Web</Value>
                <Id>W</Id>
            </ImportStyleCustomField>
    </CustomFields>
    <CustomFieldsPerColor>
            <ImportStyleCFPerColor>
                <Name>Dessin</Name>
                <Value>xxxx01</Value>
                <ColorId>R1</ColorId>
                <ColorName>Red</ColorName>
            </ImportStyleCFPerColor>
            <ImportStyleCFPerColor>
                <Name>EAN</Name>
                <Value>EANxxx01</Value>
                <ColorId>R1</ColorId>
                <ColorName>Red</ColorName>
            </ImportStyleCFPerColor>
            <ImportStyleCFPerColor>
                <Name>Backend</Name>
                <Value>No</Value>
                <Id></Id>
                <ColorId>R1</ColorId>
                <ColorName>Red</ColorName>
            </ImportStyleCFPerColor>
            <ImportStyleCFPerColor>
                <Name>Web</Name>
                <Value>No</Value>
                <ColorId>R1</ColorId>
                <ColorName>Red</ColorName>
            </ImportStyleCFPerColor>
            <ImportStyleCFPerColor>
                <Name>Pallevare</Name>
                <Id>Y</Id>
                <ColorId>R1</ColorId>
```

```
                        <ColorName>Red</ColorName>
                    </ImportStyleCFPerColor>
                    <ImportStyleCFPerColor>
                        <Name>Sex</Name>
                        <Value>Boy</Value>
                        <Id>B</Id>
                        <ColorId>R1</ColorId>
                        <ColorName>Red</ColorName>
                    </ImportStyleCFPerColor>
                </CustomFieldsPerColor>
            </Fields>
        </ImportStyle>
    </ImportStyles>
</ImportStyleCommand>
```

Example of input JSON data call:

```
var dataInput =
"<ImportStyleCommand><ImportStyles><ImportStyle><StyleNumber>0444</StyleNumber><Fiel
ds><NewStyleNumber>0444</NewStyleNumber><NewStyleName>0444</NewStyleName><Ne
wStyleState>Delivered</NewStyleState><UpdateMultipleStyles>true</UpdateMultipleStyles><Cu
stomFields><ImportStyleCustomField><Name>Dessin</Name><Value>xxxx01</Value></ImportS
tyleCustomField><ImportStyleCustomField><Name>EAN</Name><Value>EANxxx01</Value><Cl
earContent>true</ClearContent></ImportStyleCustomField><ImportStyleCustomField><Name>W
ashing instructions
(UK)</Name><Value></Value><Id>60</Id></ImportStyleCustomField><ImportStyleCustomField>
<Name>Backend</Name><Value>Yes</Value><Id></Id></ImportStyleCustomField><ImportStyle
CustomField><Name>Web</Name><Value>Yes</Value></ImportStyleCustomField><ImportStyle
CustomField><Name>Pallevare</Name><Id>N</Id></ImportStyleCustomField><ImportStyleCust
omField><Name>Sex</Name><Value>Girl</Value><Id>G</Id></ImportStyleCustomField><Impor
tStyleCustomField><Name>Marketing
media</Name><Value>Web</Value><Id>W</Id></ImportStyleCustomField></CustomFields><Cu
stomFieldsPerColor><ImportStyleCFPerColor><Name>Dessin</Name><Value>xxxx01</Value><
ColorId>R1</ColorId><ColorName>Red</ColorName></ImportStyleCFPerColor><ImportStyleCFP
erColor><Name>EAN</Name><Value>EANxxx01</Value><ColorId>R1</ColorId><ColorName>Re
d</ColorName></ImportStyleCFPerColor><ImportStyleCFPerColor><Name>Backend</Name><V
alue>No</Value><Id></Id><ColorId>R1</ColorId><ColorName>Red</ColorName></ImportStyleC
FPerColor><ImportStyleCFPerColor><Name>Web</Name><Value>No</Value><ColorId>R1</Col
orId><ColorName>Red</ColorName></ImportStyleCFPerColor><ImportStyleCFPerColor><Name
>Pallevare</Name><Id>Y</Id><ColorId>R1</ColorId><ColorName>Red</ColorName></ImportSt
```

yleCFPerColor><ImportStyleCFPerColor><Name>Sex</Name><Value>Boy</Value><Id>B</Id><ColorId>R1</ColorId><ColorName>Red</ColorName></ImportStyleCFPerColor></CustomFieldsPerColor></Fields></ImportStyle></ImportStyles></ImportStyleCommand>";

```
    $.ajax({
        url: "https://my.delogue.com/v1.0/import/styles/",
        type: "PATCH",
        contentType: "text/xml; charset=utf-8",
        dataType: 'text',
        data: dataInput,
        cache: false,
        crossDomain: true,
        beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer ' + token); },
        success: function (data) {
          alert(data);
        },
        error: function (x, y, z) {
          alert('Error while importing style data\n' + x.responseText);
        }
    })
}
```

VALIDATION:

Field	Validation	Error message	Action
StyleNumber	If this field value is missing	Style number missing for record number: {#}	Style will not be updated
StyleNumber	If for this field no record found in Delogue, Style not updated	No record found for style number ... in Delogue for record no: {#}	Style will not be updated
Fields	If the field collection is missing, Style not updated.	Style data missing to update of the style number ... at record number: {#}	Style will not be updated

Fields: NewStyleNumber	If this number is already used for some other style in Delogue	New style number ... already exist in Delogue at record no: {#}, Style number not updated	Style number will not update and custom fields are not updated
Fields: NewStyleNumber	If this field is missing value	New style number missing at record no {#}, Style number not updated	Style number will not update
Fields: NewStyleState	If this field exists & existing style state is not in published state.	Style is not in published state, style state not updated for record no {0}	Style state will not update
Fields: NewStyleState	If this field is exist & value provided is other than "Cancelled" /" Delivered"	New style state must be Cancelled/Delivered, style state not updated for record no: {0}	Style state will not update
StyleNumber and UpdateMultipleStyles	If for this field multiple styles found, we could not identify which style to update when UpdateMultipleStyles is false	Multiple records found for style number ... at record no: {0}	Style will not update
Fields: CustomField: Name	Custom field with name is missing	Custom field name missing at record no: {#}	Custom field will not update

Fields: CustomField: Name	Custom field with name records not found in the system.	Custom field with name ... not found at record no: {#}	Custom field will not update
Fields: CustomField: Value /Id (Id in case allowed value) & ClearContent = false	(for other than allowed value) Custom field with name, value missing (for allowed value) Custom field with name, value & id missing	Custom field with name ..., value missing at record no: {#} Custom field with name ..., value or id missing at record no: {#}	Custom field will not update
Fields: CustomField: Value/Id (Id in case allowed value) & ClearContent = false	(for other than allowed value) If custom field allowed value type with wrong allowed value from Delogue (for allowed value) 1. If custom field allowed value type with wrong allowed value & id from Delogue (when both value and id sent) 2. If custom field allowed value type with wrong allowed value from Delogue (when value sent without id) 3. If custom field allowed value type with wrong allowed value id from Delogue (when is sent without value)	(for other than allowed value) Custom field with name ... provided with wrong allowed value .... at record no: {#} (for allowed value) 1. Custom field with name ... provided with wrong allowed value ... and allowed value id ...at record no: {#} 2. Custom field with name ... provided with wrong allowed value ... at record no: {#} 3. Custom field with name ... provided with wrong allowed value id ...at record no: {#}	Custom field will not update

Fields: CustomField: Value & ClearContent = false	If custom field date type with wrong date format. Expected format must be in dd-MMM-yyyy e.g., 31-OCT-2017	Custom field with name ... provided with wrong date value at record no: {#}	Custom field will not update
Fields: CustomField: Value & ClearContent = false	If custom field text type with wrong limited maximum characters.	Custom field with name ... provided with wrong limit of maximum number of characters value ... at record no: {#}	Custom field will not update
Fields: CustomFieldsPerColor	Id custom field per color values sent when does not having custom field per color module access	Organization does not have custom field per color module access. So custom field per color values not imported at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: Name	Custom field with name is missing	Custom field per color - Custom field name missing at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: Name	Custom field with name records not found in the system.	Custom field per color - Custom field with name ... not found at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: Value /Id (Id in case allowed value)	(for other than allowed value) Custom field with name, value missing (for allowed value) Custom field with name, value & id missing	(for other than allowed value) Custom field per color - Custom field with name ..., value missing at record no: {#} (for allowed value) Custom field per color - Custom field with name ..., value or id missing at record no: {#}	Custom field per color will not update

Fields: CustomFieldsPerColor: Name	Custom field with name is not diff per color	Custom field per color - Custom field with name ... is not diff per color at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: Name	Custom field with name is of nested allowed value type	Custom field per color - Custom field with name {0} is of nested allowed value type (which is not diff per color & value added only with import web service) at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: ColorId	Custom field with ColorId is missing	Custom field per color - color id missing at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: ColorName	Custom field with ColorName is missing	Custom field per color - color name missing at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: ColorId/ColorName	Custom field with wrong ColorId/ColorName which is not found in system or not included in style color.	Custom field per color - invalid style color with name ... & id ... at record no: {#}	Custom field per color will not update

Fields: CustomFieldsPerColor: Value	(for other than allowed value) If custom field allowed value type with wrong allowed value from delogue (for allowed value) 1. If custom field allowed value type with wrong allowed value & id from delogue (when both value and id sent) 2. If custom field allowed value type with wrong allowed value from delogue (when value sent without id) 3. If custom field allowed value type with wrong allowed value id from delogue (when is sent without value)	(for other than allowed value) Custom field per color - Custom field with name ... provided with wrong allowed value .... at record no: {#} (for allowed value) 1. Custom field per color - Custom field with name ... provided with wrong allowed value ... and id ... at record no: {#} 2. Custom field per color - Custom field with name ... provided with wrong allowed value ... at record no: {#} 3. Custom field per color - Custom field with name ... provided with wrong allowed value id ... at record no: {#}	Custom field per color will not update
Fields: CustomFieldsPerColor: Value	If custom field date type with wrong date format. Expected format must be in dd-MMM-yyyy eg. 31-OCT-2017	Custom field per color - Custom field with name ... provided with wrong date value .... at record no: {#}	Custom field per color will not update

Fields: CustomFieldsPerColor: Value	If custom field text type with wrong limited maximum characters.	Custom field per color - Custom field with name ... provided with wrong limit of maximum number of characters value ... at record no: {#}	Custom field per color will not update

Note: {#} indicate the record number of styles in input.

7. IMPORT HIERARCHY VALUES

Prerequisite and Authentication please see section 2 and section 3.

7.1 IMPORT HIERARCHY VALUES FOR ADMIN STYLE CUSTOM FIELD IN DELOGUE

Service API detail: This API will import hierarchy values for admin style custom fields of type "Hierarchy values"

End point for this API: **https://**my.delogue.com**/Import/NestedCustomField/**
For this API you can send input as a JSON.
In above rest API URL section description is as below
Import – it's importing data in Delogue
NestedCustomField - Clears old values collection & relations of hierarchy values of style custom field & read into Delogue with new input values.
HTTP method type – POST
The token from authentication should be sent as a 'Bearer' token with your input data. ex. in jquery call -
beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer ' + token); },
Required Headers:
Content-Type:application/json
Authorization:Bearer {authentication Token}
Accept:application/json

NOTE:
- User can create new custom field of type "Hierarchy values" from Add Custom Fields (Admin ⇒ Custom Fields ⇒ Styles tab)

- Before adding hierarchy values with service, these custom fields should be created from Delogue UI.
- Also, avoid circular relationships between data child and parent when adding allowed values.
- Code (Id of allowed value) should be unique for a single custom field.

INPUT JSON ⇒ Input data is in the form of a collection of hierarchy values of custom fields like:

Code: It is the ID of the allowed value that the user can select in the drop down of hierarchy type custom fields.

Parent: It is the ID of the allowed value of the parents Hierarchy Custom Field

ParentCFName: It is the Name of the parents Hierarchy Custom Field e.g. ICL1

Description: allowed value for custom field (This is the value that the user can select in the drop down)

Name: The name of the Hierarchy Custom Field e.g. ICL2


```
{
  "StyleCustomFieldNestedAllowedValues":[
        {
        "Code":"A",
        "Parent":"",
        "ParentCFName":"",
        "Description":"a",
        "Name":"ICL1"
        },
        {
        "Code":"B",
        "Parent":"",
        "ParentCFName":"",
        "Description":"b",
        "Name":"ICL1"
        },
        {
        "Code":"C",
        "Parent":"",
        "ParentCFName":"",
        "Description":"c",
        "Name":"ICL1"
        },
        {
        "Code":"A1",
```

"Parent":"A",
"ParentCFName":"ICL1",
"Description":"a1",
"Name":"ICL2"
},
{
"Code":"A2",
"Parent":"A",
"ParentCFName":"ICL1",
"Description":"a2",
"Name":"ICL2"
},
{
"Code":"B1",
"Parent":"B",
"ParentCFName":"ICL1",
"Description":"b1",
"Name":"ICL2"
},
{
"Code":"C1",
"Parent":"C",
"ParentCFName":"ICL1",
"Description":"c1",
"Name":"ICL2"
},
{
"Code":"A11",
"Parent":"A1",
"ParentCFName":"ICL2",
"Description":"a11",
"Name":"ICL3"
},
{
"Code":"A12",
"Parent":"A1",
"ParentCFName":"ICL2",
"Description":"a12",
"Name":"ICL3"
},

```
        {
        "Code":"A21",
        "Parent":"A2",
        "ParentCFName":"ICL2",
        "Description":"a21",
        "Name":"ICL3"
        },
        {
        "Code":"B11",
        "Parent":"B1",
        "ParentCFName":"ICL2",
        "Description":"b11",
        "Name":"ICL3"
        }
  ]
}
```

Example of input json data call:

```
 $.ajax({
        url: "https://my.delogue.com/Import/NestedCustomField/",
        type: "POST",
        contentType: "application/json; charset=utf-8",
        dataType: 'json',
        data: JSON.stringify({
  "StyleCustomFieldNestedAllowedValues":[
        {
        "Code":"A",
        "Parent":"",
        "ParentCFName":"",
        "Description":"a",
        "Name":"ICL1"
        },
        {
        "Code":"B",
        "Parent":"",
        "ParentCFName":"",
        "Description":"b",
        "Name":"ICL1"
        },
        {
```

"Code":"C",
"Parent":"",
"ParentCFName":"",
"Description":"c",
"Name":"ICL1"
},
{
"Code":"A1",
"Parent":"A",
"ParentCFName":"ICL1",
"Description":"a1",
"Name":"ICL2"
},
{
"Code":"A2",
"Parent":"A",
"ParentCFName":"ICL1",
"Description":"a2",
"Name":"ICL2"
},
{
"Code":"B1",
"Parent":"B",
"ParentCFName":"ICL1",
"Description":"b1",
"Name":"ICL2"
},
{
"Code":"C1",
"Parent":"C",
"ParentCFName":"ICL1",
"Description":"c1",
"Name":"ICL2"
},
{
"Code":"A11",
"Parent":"A1",
"ParentCFName":"ICL2",
"Description":"a11",
"Name":"ICL3"

```
        },
        {
        "Code":"A12",
        "Parent":"A1",
        "ParentCFName":"ICL2",
        "Description":"a12",
        "Name":"ICL3"
        },
        {
        "Code":"A21",
        "Parent":"A2",
        "ParentCFName":"ICL2",
        "Description":"a21",
        "Name":"ICL3"
        },
        {
        "Code":"B11",
        "Parent":"B1",
        "ParentCFName":"ICL2",
        "Description":"b11",
        "Name":"ICL3"
        }
  ]
}),
        cache: false,
        crossDomain: true,
        beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer ' +
token); },
        success: function (data) {
        var importErrors = "";
        if (Object.keys(data.importErrors).length > 0) {
                importErrors = "\n Error while importing details : \n";
                $.each(data.importErrors, function (i, item) {
                importErrors += i + ':' + item + '\n';
                });
        }
        },
        error: function (x, y, z) {
        alert('Error while importing style data\n' + x.responseText);
        } })
```

On success: Call will return in success ***means data imported success with list of errors***
On error: If API fails you get message in response text

*** Validation:**

Field	Validation	Error message	Action
Wrong format input	If user send out wrong format data in JSON or missing something in JSON input	Wrong import hierarchy values input: collection is blank or null. (maybe due to wrong format sent)	Allowed value & relation will not add.
Correct format input	If the organization does not have "Hierarchy Values" custom fields module access	Organization does not have hierarchy value module access.	Data will not import, or Old data will not overwrite.
Correct format input	If the organization does not have "Hierarchy value" custom fields added in admin style custom fields	Organization does not have any hierarchy value type custom fields in Delogue.	Data will not import, or Old data will not overwrite.
Code/Description/Name	If value is missing for this field code,	Wrong import hierarchy values input: custom field	Data will not import, or Old data will not overwrite.

	description or name.	code / name / description missing for some records.	
Parent/ParentCFName	If value for Parent field missing when value for ParentCFName is provided	Wrong import hierarchy values input: parent custom field code is missing where parent custom field name provided.	Data will not import, or Old data will not overwrite.
Parent/ParentCFName	If value for ParentCFName field missing when value for Parent is provided	Wrong import hierarchy values input: parent custom field name is missing where parent custom field code provided.	Data will not import, or Old data will not overwrite.
Name	If the custom field name for the given allowed value is not found in the admin custom field Delogue.	Custom Field with name {0} for allowed value code {1} not found in Delogue hence skipped.	Data will be imported with skipping this record and also relation for this record.
ParentCFName	If the parent custom field name for the given allowed value not	Parent custom field with name {0}	Data will be imported with skipping adding

	found in the admin custom field Delogue.	for allowed value code {1} not found in Delogue hence skipped.	reactions to this record.
Code/Description/Name	If value for the fields code, description & name duplicated in the input collection.	For custom field name {0} custom field allowed value with code {0} & description {1} duplicated hence skipped.	Data will be imported for this record
Code/Name	If value for the fields code, name duplicated in the input collection.	For custom field name {0} custom field allowed value with code {0} duplicated hence skipped.	Data will be imported for this record

8. IMPORT SUPPLIERS

8.1 OVERVIEW

Delogue can import Supplier data from external web services. We have implemented a nightly (i.e. GMT 00:00) automated process for this. This process will call client web service on the basis of Supplier's Id. Other suppliers' details in Delogue will get updated from data provided by client web service. Details of each request (call to client web service) and response (response of client web service) are explained in the following sections of the document.

8.2 PREREQUISITE

- For using this functionality client need to write a SOAP service which will return supplier details*.

- Organization should have access to the "Supplier Sync" module in Delogue.
- Login to Delogue with Admin role and Add supplier (under Admin → Supplier tab) with Supplier ID.
- Add supplier web service details into Delogue. So that we can call your web service.
- Please see the following visual: Screenshot 1: Supplier Sync.

Service URL: Enter URL of web service implemented by client
User Name: Enter Authenticated user name of web service
Password: Enter Password for user of this web service
Transport Credential Type: Select transport credential type either Windows or Windows-Ntlm

8.3 IMPORTING SUPPLIER PROCESS IN DELOGUE:

After completing prerequisite step Delogue performs following steps:

1. Delogue calls web service, provided in the above screenshot by the user.
2. Supplier web service will return the Supplier Details list to Delogue.
3. Following is an example of a response from a web service.
4. Example:


```
[
 {
 "No": "100",
 "Name": "Suplier company name",
 "Address": "5/F., ABC building, MG Rd",
 "Post_Code": "112233",,
 "Country_Region_Code": "CN",
 "Phone_No": "99999999",
 "City": "your city",
 "Blocked": 0,
 "Home_Page": null,
 "Fax_No": null,
 "VAT_Registration_No": null,
 "Currency_Code": "USD"
 }

 ]
```


- We will get "No" field from the response and find that No's supplier Id is present in Delogue. If it exists, then we will update the details of the supplier provided in response.
- If any failure occurs, Delogue will send failure email to Designer Organization Email Id.

8.4 SAMPLE OF WEB SERVICE

To call your SOAP service you need to follow guidelines described below:

Your service should have interface as below (please follow names as per description)
We will call your webservice method ReadMultiple() and we required response from your side in the format DelogueVendorWS[]

```
public interface DelogueVendorWS_Port
        {
 ReadMultiple_Result ReadMultiple(ReadMultiple request);
        }

        public partial class ReadMultiple
        {
        public DelogueVendorWS_Filter[] filter;
        public string bookmarkKey;
        public int setSize;
        public ReadMultiple()
        {
        }

public ReadMultiple(DelogueVendorWS_Filter[] filter, string bookmarkKey, int setSize)
{
this.filter = filter;
this.bookmarkKey = bookmarkKey;
this.setSize = setSize;
}
}


public partial class DelogueVendorWS_Filter : object,
System.ComponentModel.INotifyPropertyChanged
        {
        private DelogueVendorWS_Fields fieldField;
        private string criteriaField;
        public DelogueVendorWS_Fields Field{get;set;}
        public string Criteria{get;set;}
```

```
        }

        public enum DelogueVendorWS_Fields
                {
          No, //Supplier ID
          Name,
                Address,
                Post_Code, //ZIP Code
                Country_Region_Code, //Country Code
                Phone_No, //PHONE NUMBER
                City,
                Blocked, //IsActive
                Home_Page, //Website
                Fax_No, //Fax number
                VAT_Registration_No,//VAT#
                Currency_Code, //Currency
                }
public partial class DelogueVendorWS : object, System.ComponentModel.INotifyPropertyChanged
        {
        private string noField { get; set; }
        private string nameField { get; set; }
        private string addressField { get; set; }
        private string post_CodeField { get; set; }
        private string country_Region_CodeField { get; set; }
        private string phone_NoField { get; set; }
        private string cityField { get; set; }
        private string blockedField { get; set; }
        private string home_PageField { get; set; }
        private string fax_NoField { get; set; }
        private string vat_Registration_NoField { get; set; }
        private string currency_CodeField { get; set; }
        }
```

8.5 DISCLAIMER

Classes, interfaces used in section "SOAP Service Details" have to be implemented by the company.