Custom Export Guide

Version 2.77

—

Delogue.

Table of Content:

1 INTRODUCTION..... 3
2 EXPORT DATA FROM DELOGUE PLM..... 3
2.1 AUTHENTICATION.....3
Getting the API Key..... 4
Using the API Key.....5
3 EXPORT DATA FROM DELOGUE USING THE DELOGUE REST WEB SERVICE..... 6
3.1 OVERVIEW.....6
3.2 PREREQUISITE: GETTING API KEY FOR AUTHENTICATION..... 6
3.3 STEP 1: GETTING STYLE JSON DATA BY STYLE NUMBER IN DELOGUE..... 6
3.3.1 SERVICE API DETAIL..... 6
3.3.2 VALIDATIONS.....8
3.3.3 JSON EXPLAINED..... 8
3.4 STEP 2: STYLE WEBHOOK SERVICE DETAIL IN DELOGUE..... 16
3.5 INPUT JSON FORMAT FOR THE SERVICE REQUIRED ARE AS FOLLOWS:..... 21
4 GETTING AN UPDATED STYLE NUMBER LIST BY GIVING A TIMESTAMP FROM DELOGUE..... 23
4.1 AUTHENTICATION.....23
4.2 SERVICE API DETAIL.....23
4.3 RESPONSE FROM API STYLE JSON DATA OBJECT STRUCTURE..... 25
5 API FOR GETTING STYLE NUMBERS PER BRAND AND SEASON..... 27
5.1 INTRODUCTION..... 27
5.2 AUTHENTICATION.....27
Getting the API Key..... 27
Using the API Key.....28
5.3 API OVERVIEW.....28
5.4 MANUAL PUSH OF STYLE NUMBERS PER BRAND AND SEASON.....29
6 EXPORTING CUSTOM EXPORT STYLE DATA FROM DELOGUE VIA FTP..... 29
6.1 MARKING STYLES AS READY FOR EXPORT.....30
6.2 SETTING UP FTP SERVER.....30
6.3 OPTIONS FOR FTP EXPORT.....30

1 INTRODUCTION

This document will explain how and which data you can export from Delogue. We will update this document every time we release new features.

Exporting data from Delogue can be done in 3 different ways:

1. A webhook solution where the user can click a button and send a style number to a web service created by the system provider.
2. Calling a web service with a timestamp and get a list of styles that have changed in return
3. Sending a style number to our web service and getting JSON with all relevant style data in return.

2 EXPORT DATA FROM DELOGUE PLM

This chapter explains how to export data from the Delogue PLM system. These APIs are available exclusively to Delogue PLM customers and are limited to users with approved credentials.

Delogue PLM exposes REST-based services to export data from Delogue PLM to external systems. Therefore, you must authenticate yourself by passing the API key while calling Delogue PLM endpoints to call any API. We have explained the details of each request-response in the following sections of this document.

2.1 AUTHENTICATION

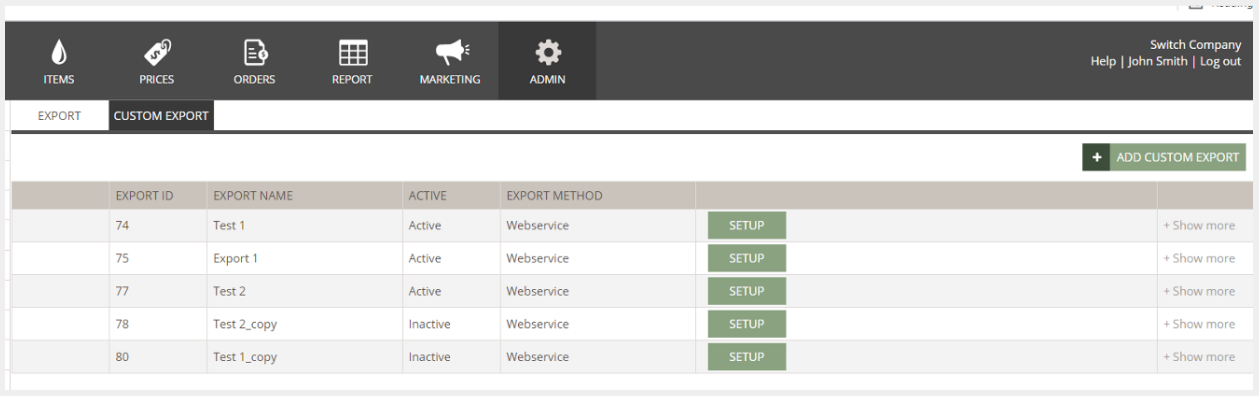
To export data from Delogue PLM, the calling application needs to authenticate itself with Delogue PLM first. The authentication for export is facilitated by checking the API key passed to the Delogue PLM application while calling the export API.

This section will show how the API key can be obtained and passed to the Delogue PLM application for exporting the data.

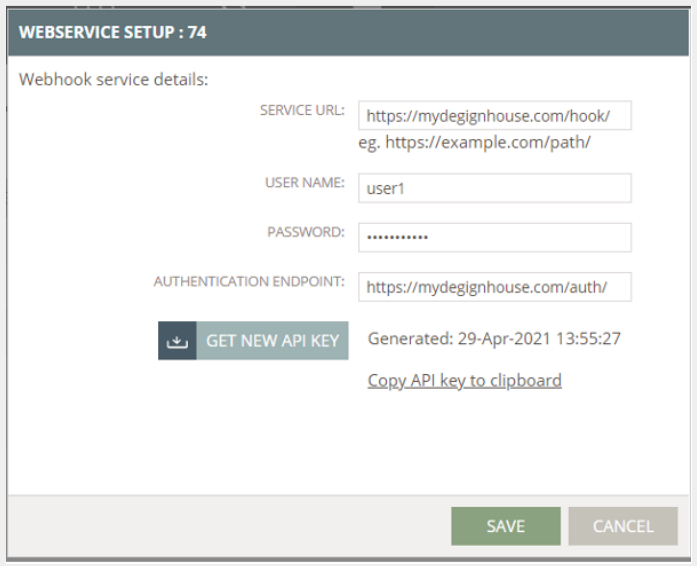
Getting the API Key

To use the export feature, they will have to configure their webhook details in Delogue PLM. You can locate this configuration in the Admin sections under Import/Export settings(ADMIN→IMPORT/EXPORT→CUSTOM EXPORT).

The webhook configuration popup can be opened by clicking on the setup button for the export.



After clicking on the setup button, the webhook settings popup will show. Here is what the webhook configuration popup looks like:



We will discuss the other fields in this popup in more detail in the coming chapter, but we need to copy the API key from this popup for authentication purposes. The API key will typically be of the following format:

IAp74oCL4oCL4oCL4oCL4oCLCiJPcmdhbml6YXRpb25JZCI6ICIyIiwKIkV4cG9ydElkIjogIjEiLAoiQXBpS2V5QmFzZSI6ICIyMjItMjIyLTIyMi0yMjIiCn3igIvigIvigIvigIvigIsK

Please save this API key. You must pass the API key as a header in all the export requests to Delogue PLM.

Note: There is an option to generate the API key again. If you create the API key again, it will invalidate the existing API key and create a new one.

Using the API Key

The API key generated in the previous step should now be passed as an HTTP authorization header as a bearer token to the export requests in Delogue PLM.

Authorization: Bearer
lAp74oCL4oCL4oCL4oCL4oCLCiJPcmdhbml6YXRpb25JZCI6ICIyIiwKIkV4cG9ydElkIjogIjEiLAoiQXBpS2V5QmFzZSI6ICIyMjItMjIyLTIyMi0yMjliCn3iglviglviglviglviglsK

Below is an example showing how the API key should be passed as a bearer token in the export requests.

The endpoint for the API:

https://my.delogue.com/External/api/CustomExport/Styles/{styleNumber}

?allowHalfSamples=0

NOTE: The `allowHalfSamples query` parameter is **optional** and defaults to `0`. Toggling this to `1` will make the sample quantity field below return a `decimal`, instead of the default of an `integer`:

`styleSampleRequests[].requestedSampleSizeSpecs[].quantity`

For this API, you have to send input as a style number & get output as a JSON object.

The API Key should be sent as 'Bearer' token with your input data. ex. in jQuery call -

beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer ' + 'IAp74oCL4oCL4oCL4oCL4oCLCiJPcmdhbml6YXRpb25JZCI6ICIyIiwKIkV4cG9ydElkIjogIjEiLAoiQXBpS2V5QmFzZSI6ICIyMjItMjIyLTIyMi0yMjliCn3iglviglviglviglviglsK'); },

3 EXPORT DATA FROM DELOGUE USING THE DELOGUE REST WEB SERVICE

3.1 OVERVIEW

This tutorial guides you on how to export style from Delogue to an external system like ERP.

There are two ways you can use to export a style:

a) Step 1: The client can invoke a service to get style data by style number from Delogue.

b) Step 2: Additionally. There is a trigger point in Delogue UI from which you can export style data. The organization must have the Webhook module & must create a REST web service, which can take the input: style number.

3.2 PREREQUISITE: GETTING API KEY FOR AUTHENTICATION

Delogue exposes REST-based services to import/export data into/from Delogue from an external system. To call any API you need to authenticate yourself by generating and copying the API key from the Delogue application. Refer to the authentication section above to see more details on this.

3.3 STEP 1: GETTING STYLE JSON DATA BY STYLE NUMBER IN DELOGUE

Delogue exposes REST-based services to import/export data into/from Delogue from an external system.

3.3.1 SERVICE API DETAIL

Endpoint for this API:

https://my.delogue.com/External/api/CustomExport/Styles/{styleNumber}

?allowHalfSamples=0

For this API, you have to send input as a style number & get output as a JSON object.

In the above rest API, the URL section description is as below:

- **`External/API/CustomExport/Styles/`**– it fetches style data in Delogue
- **`style/{styleNumber}`** - style number of the style for which you need to get style data
- `?allowHalfSamples=0` - optional query parameter that'll return sample sizes as decimals if toggled.
- HTTP method type – GET
- The APIKEY should be sent as a 'Bearer' token with your input data. ex. in jQuery call –
    - beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer ' + APIKEY); },

Note:

- This API will return the style json data.
- Here API will search style in Delogue as below
    - It will find the style for the input style number.
    - If the style is not found, then it will return an error.
    - If multiple styles are found for the input style number, it will return an array of styles.
    - In case multiple primary supplier styles are found, it will also return the validation error message.

An example service invoke call is like this: (for style number 1186)

```
$.ajax({
        url: "https://my.delogue.com/External/api/CustomExport/Styles/1186",
        type: "GET",
        contentType: "application/json; charset=utf-8",
        dataType: 'json',
        cache: false,
        crossDomain: true,
        beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer ' + APIKEY); },
        success: function (styleJsonData) {
        alert("Success");
        },
```

```
error: function (x, y, z) {
alert('Error while getting style json data\n' + x.responseText);
}
})
```

On success: Call will return in success with information on style number list.

On error: If API fails, you get an error message in response text with a bad request status.

3.3.2 VALIDATIONS

Validation	Error message	Action
If the input style number is not found.	Error fetching style data - Style does not exist.	Style data will not return. But the error returns with an error message.
If there are multiple styles found for the input style number but without any primary supplier.	Error fetching style data - Multiple styles found for style number: {#}, but no primary supplier style is found in the system.	Style data will not return. But error returns with an error message
If there are multiple styles found for the input style number and primary supplier.	Error fetching style data - Multiple styles found for the style number: {#} & primary supplier.	Style data will not return. But error returns with an error message
If the style has half samples configured, but the API is not called with the `allowHalfSamples=1` query parameter.	Sample Request contains Half sample sizes. Please pass 'allowHalfSamples=1' as query string in the API call	Style data will not return. But error returns with an error message
Any other exception within the service API	Error in fetching style JSON data: Exception error message	Style data will not return. But error returns with an error message

3.3.3 JSON EXPLAINED

Here is an explanation of the content of the JSON file

```
[
  {
    "id": 22108, //Internal Delogue Id (Numeric)
    "styleState": "Published", //Allowed states: Unpublished, Published, Cancelled, WorkInProgress, Delivered, Deleted (String)
    "styleName": "Casual Shirt", //Name of the style (String)
    "isPrimary": true, //Is it the primary style (Boolean)
    "userDefinedId": "CS121", // Style number: "Style No" in the style header (String)
    "description": "Casual shirt for everyday use. Slim fit", //The description field in the style header (String)
    "pictureUrl": "https://d2qm6ct9hhtmfg.cloudfront.net/style_22108/CasualShirt.jpg", //URL to the style picture (String)
    "companyName": "Best Clothing", //Name of the company (String)
    "companyContactPersonName": "Mikkel Thormod", //Name of the company contact person (String)
    "brandName": "First Brand1", //Name of the brand (String)
    "supplierName": "Supplier Org", //Name of the style supplier (String)
    "supplierId": "4", //Internal Delogue Id (String)
    "userDefinedSupplierId": "S123", //Id of the style supplier (String)
    "supplierCountry": "India", //Country of the supplier (String)
    "supplierCountryID": "IN", //Id of the supplier country (String)
    "supplierContactPerson": {
      "name": "Dorthe", //Name of the supplier contact person (String)
      "email": "dorthe@acme.com", //Email of the supplier contact person (String)
      "status": "Yes" //"Yes" or "No", if yes then the supplier contact person is active (String)
    },
    "supplierFacilities": [ // List of supplier facilities
      {
        "facilityType": "Wash unit 4", // Type of facility type set by Admin (String)
        "facility": { // Details of facility type
          "id": "",   // Id of facility (String)
          "name": "Washing station",  // Name of facility (String)
          "address": "Pune",  // Address of facility (String)
          "zipCodeOrCity": "411027-Pune", // zip code of facility (String)
          "country": "Bangladesh",    // Country of facility (String)
          "status": "no" //“Yes” or “No”, if yes then the facility is active (String)
        }
      }
    ],
    "seasonName": "SS21", //Name of the Season (String)
    "project": "2nd Drop", //Name of the project (Sub season) (String)
    "groupName": "Shirts", //Name of the Group (String)
    "categories": [ //List of categories selected in the style header
      "Shirt",
      "Casual"
    ],
    "readyForExport": true, // Indicate whether this style is ready for export or not
    "styleFiles": [ // list of style files
      {
        "colorwayId": "35", // Color Way id
        "colorwayId2": "26",    // COlor way id 2
```

```
            "colorwayName": "Agave Green",  // Color way name
            "fileName": "images.jpg", // Style file name
            "url":
"https://s3-eu-west-1.amazonaws.com/designhubtest/organization_2/style_63573/1ca5938b-87e6-4e45-8bca-fa474a68798a/images.j
pg", //URL to the style file (String)
            "status": "Active" //style file is active (Boolean)
          }
        ],
        "styleColors": [ //List of style colors
          {
            "id": "912", //Color Id (String)
            "id2": null, //Color Id2 (String)
            "name": "Red rose", //Name of the color (String)
            "status": true, //Color is active (Boolean)
            "colorReference": "19-2030tcx", //Color reference. (Pantone code or Coloro code, RGB etc.) (String)
            "colorNote": "This looks best on velvet fabric", //User-defined note, set in admin/colors per color (String)
            "colorLanguages": [ //List of languages the color name is translated into
              {
                "languageId": "101", //User-defined id of the language (String)
                "languageName": "Danish", //User-defined name of the language (String)
                "name": "Rød rose" //Name of the color translated into the above language (String)
              }
            ],
            "colorGroups": [ //List of the color groups the color is part of
              {
                "id": "33", //user defined id for the color group (String)
                "name": "Green", //User defined name for the color group (String)
                "status": "Active" //Is the color group "Active" or "Inactive" (String)
              }
            ]
          }
        ],
        "sizeRangeName": "Standard", //Name of the size range (String)
        "sizeRangeUserDefinedId": "123456", //Id of the size range (String)
        "sizes": [
          {
            "id": 6, //Internal Delogue Id of the size (Numeric)
            "name": "XS", //Name of the size (String)
            "isActive": true, //Is the color active: true or false (Boolean)
            "sizeRangeId": 25, //Interal Delogue Id of the size range (Numeric)
            "measurementChartId": 22107, //Internal Delogue Id of the Measurement chart (Numeric)
            "sizeId": "XS-1" //User defined size id (String)
          }
        ],
        "allMeasurements": [ // List of all all measurements
          {
            "id": 37723, //Internal Delogue Id of all measurements (Numeric)
            "description": "line1", // Description of measurement
            "position": 1, // The relative position of the measurement with all measurements
            "measurements": [ // List of measurements
```

```
            {
              "size": {
                "id": 265, //Internal Delogue Id of measurements (Numeric)
                "name": "11", // Name of measurement (String)
                "isActive": true, //Is the color group "Active" or "Inactive" (String)
                "sizeRangeId": 51, // Size range id (Numeric)
                "measurementChartId": 63665, // Measurement chart Id (Numeric)
                "sizeId": "A11" // User defined size id (String)
              },
              "computedValue": 70.0, // Computed value of the measurements (Numeric)
              "value": 70.0 // Value of the measurement (Numeric)
            }
          ],
          "measurementTag": { // Measurement tag
            "id": 3, //Internal Delogue Id of measurement tags (Numeric)
            "userDefinedId": null, //user defined Id of measurement tag (String)
            "name": "tag1" // Name of the measurement tag (String)
          }
        }
      ],
      "measurementFiles": [
        {
          "fileName": "images.jpg", // Name of the measurement file
          "url":
"https://s3-eu-west-1.amazonaws.com/designhubtest/organization_2/style_63573/184c8017-3ee5-4f57-b077-4ea37ca2aad3/images.jpg", //URL to the measurement file (String)
          "status": "Active" //measurement File is active (Boolean)
        }
      ],
      "items": [
        {
          "id": 7525, //Internal Delogue Id (Numeric)
          "itemName": "Button Wood 3", //Name of the Item (String)
          "userDefinedId": "BW3", //Item number defined in the Items header (String)
          "description": "Button made of Oak wood style 3", //Description of the item header (String)
          "supplierName": "Test supplier", //Name of the item supplier (String)
          "supplierId": 3, //Internal Delogue Id (Numeric)
          "userDefinedSupplierId": "test 123",//User Defined Supplier Id
          "brandName": "First Brand1", //Name of the item brand (String)
            "categories": [ // List of Item Categories
            "Zippers"
            ],

          "compositions": [
            {
              "description": "Brodery", //Name of the composition (String)
              "composition": "95% cotton\n5% elastane" //Description of the composition (String)
            }
          ],
          "itemUsageInStyle": {
```

```
      "placementDescription": "Top left pocket", //Placement description of the item added in the Styles/Item List (String)
      "selectedItemSizeName": "13mm", //Name of the size selected for the item added in the Styles/Item List (String)
      "quantity": 1.00000, //The quantity set for the item added in the Styles/Item List (String)
      "styleColors": [
        {
          "styleColorName": {
            "id": null, //Color Id (String)
            "id2": null, //Color Id2 (String)
            "name": "912 Red rose", //Name of the color (String)
            "status": true, //Color is active (Boolean)
          },
          "itemColorCardName": "C1", //Name of the chosen Item color card for the above chosen color (String)
          "itemColors": [
            {
              "description": "Brodery", //Description of the item Details (String)
              "composition": "95% cotton\n5% elastane", //Composition of the item (String)
              "color": "Grey", //Name of the color(If item colors are set to user Library color then this should be library color) (String)
              "colorReference": "17-1212 TPX", //Color reference. (Can be a Pantone code or Coloro code, RGB etc.) (String)
              "libraryColor": "Air Blue23", // Library color name
              "libraryColorReference": "15-3932 TCX" // Library color Reference
            }
          ]
        }
      ]
    },
    "unitName": "cm" //Name of the unit for the quantity (String)
  }
],
"customFieldsInsideStyle": [ //List of custom fields that has values in this style
  {
    "name": "Type-Group", //Name of the Custom Field (String)
    "value": "8", //The value selected by the user (String) (Allowed type custom field)
    "userDefinedId": "a8" //Id related to the selected value (String)
  }
],
"styleSampleRequests": [
  {
    "typeOfSampleName": "Photo samples", //Name of the sample type
    "requestedSampleSizeSpecs": [ //List of all possible combinations of color and size + number of samples ordered
      {
        "colorId1": "", // color Id
        "colorId2": "",  // colorId2
        "colorName": "", //When colorName is empty means "Available" color
        "quantity": 1, //Number of ordered samples in the given size and color (integer or decimal )
                       // NOTE: This will be a decimal value if the allowHalfSamples query parameter is toggled.
        "size": "XS" //Name of the size (String)
      }
    ],
    "deadline": "2015-06-09T00:00:00", //Deadline of the sample request (YYYY-MM-DDTHH:MM:SS)
    "etd": "2015-06-24T00:00:00", //Estimated time of arrival (YYYY-MM-DDTHH:MM:SS)
```

```
        "status": "Received" //State of the sample request. Can be of the following states:Planned, Requested, Confirmed, Sent, Received, Commented)
      }
    ],
    "skUs": {
      "sku": [ //List of all the SKU of the style
        {
          "number": "CS121/11/XS-1", //The number of the SKU as defined in Admin (String)
          "active": true, //State of the SKU (Boolean)
          "barcode": "" //Barcode added to the SKU (String)
        }
      ]
    },
    "styleExportPrices": {
      "isApproved": false, //Approval of the supplier cost price (Boolean)
      "stylePrices": [
        {
          "id": 18674, //Internal Delogue Id (Numeric)
          "styleId": 22108, //Internal Delogue Id of the style (Numeric)
          "seasonId": 0, //Internal Delouge Id for the season (Numreic)
          "styleColorId": 37084, //Internal Delogue Id for styleColor
          "styleColorMasterId": 9751, //Internal Delogue ID (Numeric) Only in use when prices is different per color
          "styleColorId1": "35",  // styleColorId1 (User defined Id)
          "styleColorId2": "26",  // styleColorId2 (User defined Id)
          "styleColorName": "912 Red rose - 123", //Name of the style color.
          "styleColorStatus": true, //Is the color active on the style (Boolean)
          "sizeRangeSizeName": "XS", //Name of the Size (String)
          "sizeRangeSizeUserDefinedId": "A11", //sizeRangeSize UserDefinedId
          "sizeRangeSizeId": 6, // Internal Delogue Id for the size
          "sizeRangeSizeStatus": false, //is the size active on the style (Boolean)
          "brandComments": "Price is too high. Please try to reduce", //Brands comments (String)
          "supplierComments": "Price will change before production", //Suppliers comments (String)
          "futurePriceBrandComments": null, //Brands comments on the future price (String)
          "futurePriceSupplierComments": "New price will include bag", //Supplier comments on the future price (String)
          "futureDate": "2021-04-01T00:00:00", //The time when the future price will take effect (YYYY-MM-DDTHH:MM:SS)
          "priceNegotiation": [
            {
              "futurePrice": 129.00000, //Value of future price (Numeric)
              "priceCalculation": {
                "id": 286, //Internal Delogue Id (Numeric)
                "organizationId": 2, //Internal Delogue Id (Numeric)
                "ref": "A", //Row reference in the Admin/Price (String)
                "name": "Brand target price", //Name of the row in the Admin/Price
                "typeOfCalculationId": 0, //Type of calculation 0=Currency, 1=Percentage, 2=Formula, 3=Number, 4=Item Price (Numeric)
                "currency": {
                  "id": 27, //Internal Delogue Id
                  "countryCurrency": "Denmark Krone", //Name of the Currency (String)
                  "currencyCode": "DKK", //Currency code (String)
                  "value": 0.0, //Convertion rate against the Local Currency set per season in Admin/Prices/currency (Numeric)
```

```
                "isActive": false //Is the currency active (Boolean)
              },
              "formula": null, //Formula stored for this row in Admin/Prices (String)
              "defaultValue": 0.00000, //The default value set in Admin/Prices (Numeric)
              "isSupplierCurrency": false, //Is the currency the supplier currency (Boolean)
            },
            "value": 125.00000 //Value of the Row (Numeric)
          }
        ]
      }
    ]
  },
  "styleCustomFieldPerColor": { //Custom fields with different values per color
    "styleCustomFields": [
      {
        "colorValues": [ //list of style colors with values
          {
            "styleColorId": 37085, //Internal Delogue Id (Numeric)
            "styleColorId1": "11", //User defined id of the style color (String)
            "styleColorId2": "12", //style Color Id2
            "customFieldId": 251, //Internal Delogue Id (Numeric)
            "customFieldUserDefinedId": "", //custom Field for UserDefinedId
            "styleColorName": "11 Off-White", //Style color name (String)
            "value": "Blue", //Value of the custom field for the above color (String)
            "customField": {
              "id": 1885, //Internal Delogue Id (Numeric)
              "name": "CF_Color", //Name of the custom field (String)
              "isActive": true, //Is the custom field active (Boolean)
              "userDefinedId": "105", //User defined id of the custom field
              "type": 0, //Type of custom fields: 0=Allowed Values, 1=Date, 2=Text, 4=Numeric
              "maxChar": 12, //Max number of characters allowed for this custom field (Numeric)
              "isMandatory": false, //It is mandatory to fill out this custom field before export (Boolean)
              "parentCustomFieldId": null, //The id of the mother for a Hierarchy custom field (Numeric)
              "parentCustomFieldName": null, //The name of the mother for a Hierarchy custom field (String)
              "allowedDecimals": 0, //The allowed numer of decimals allowed for this custom field (Numeric)
            },
            "customFieldAllowedValueUserDefinedID": "2" //The user defined id of the selected custom field value (String)
          }
        ]
      }
    ]
  },
  "styleCustomFieldPerSize": { //Custom fields with different values per size
    "styleCustomFieldSizeRangeSizes": [
      {
        "sizeValues": [
          {
            "sizeId": 6, //Internal Delogue Id (Numeric)
            "customFieldId": 4800, //Internal Delogue Id (Numeric)
            "customFieldUserDefinedId": "",// user Defined CustomField Id  (String)
```

```
            "sizeName": "XS", //Name of the size (String)
            "userDefinedSizeId": "A11",//user Defined Size Id  (String)
            "value": null, //The value of the custom field for the above size (String)
            "customField": {
              "id": 4800, //Internale Delogue Id
              "name": "test of numeric", //Name of the custom field (String)
              "isActive": true, //Custom field is active (Boolean)
              "userDefinedId": null, //For allowed values custom fields (String)
              "type": 4, //Type of custom fields: 0=Allowed Values, 1=Date, 2=Text, 4=Numeric
              "maxChar": 3, //Max number of characters allowed for this custom field (Numeric)
              "isMandatory": false, //It is mandatory to fill out this custom field before export (Boolean)
              "parentCustomFieldId": null, //The id of the mother for a Hierarchy custom field (Numeric)
              "parentCustomFieldName": null, //The name of the mother for a Hierarchy custom field (String)
              "allowedDecimals": 9 //The allowed numer of decimals allowed for this custom field (Numeric)
            },
            "customFieldAllowedValueUserDefinedID": "" //The user defined id of the selected custom field value (String)
          }
        ]
      }
    ]
},
"relatedStyleNumber": null, //Number of the style selected in the Relation on style from the style header
"relatedStyleName": null, //Name of the style selected in the Relation on style from the style header
        "careInstructions": [ // List of all care instructions
        { // The subtab details that these care instructions belong to in the style
        "subtabId": 2,   // Subtab Id
        "subtabUserDefinedId": "testing",            //Subtab user defined id
        "subtabName": "Main",            // Subtab name
        "exception": {   // Details of exception selected for that subtab
        "id": 4,      // Exception id
        "userDefinedId": "asdasda 123",              // Exception user defined id
        "name": "asdasdsad 123"         // Exception name
        },
        "careInstructionsData": { // Raw care instruction data for this subtab in the given style
        "icons": [   // icon data
                    {
                    "userDefinedId": "adas", // user defined id for the icon
                    "iconText": "asdasd" // text for icon
                    }
        ],
        "instructions": [        // instructions data
                    {
                    "userDefinedId": "101",              // user defined id for the instruction
                    "instructionText": "Instruction_1"               // text for instruction
                    }
        ],
        "composition": [        // compositions data
                    {
                    "userDefinedId": "101", // layer user defined id
                    "layerText": "LLayer 1",             // layer text
```

```
            "materials": [          // material details
            {
            "userDefinedId": "1", // material user defined id
            "materialText": "Materials_2",              // material text
            "percentage": 12.00   // percentage value
            }
            ],
            "unit": {    // weight/unit details for the selected composition
            "userDefinedId": null,// user defined id for the unit
            "name": "NewUnitofnewitem"              // unit name
            },
            "unitValue": 23.00000           // Selected unit value
]
},
"careInstructionsOutputs": [ // List of output care instructions
{
            "id": 85, //Internal Delogue Id (Numeric)
            "name": "fgdgdgdfg", // name of the care instructions output
                        "value": "Instruction_1/\n Instruction_1dfgInstruction_2/\n Instruction_2dfgdf", // Generated value of the
                        care instruction based on the pattern set in care instructions output (String)
            "type": "Instructions", // Type of the care instruction output
            "translations": [ // List of translations for the care instructions output
  {
            "id": "126",            // Internal Id for the care instruction translation (String)
            "name": "Arabic", // Name of the language (String)
            "value": "Instruction_1/\n Instruction_1dfgInstruction_2/\n Instruction_2dfgdf"
                                    // Value of the care instruction translated in the current language (String)
            }
            ]
    },
{
            "id": 83,               //Internal Delogue Id (Numeric)
            "name": "adsasd",    // name of the care instructions output
            "value": "LLayer 112.00Materials_289, UNIT AND WEIGHT/\nShell layer10.00Materials_189, UNIT AND WEIGHT45",
                        // Generated value of the care instruction based on the pattern set in care instructions output (String)
            "type": "Composition", // Type of the care instruction output
            "translations": [       // List of translations for the care instructions output
            {
            "id": "123",            // Internal Id for the care instruction translation (String)
            "name": "Spanish",              // Name of the language (String)
            "value": "12.00material89, UNIT AND WEIGHT/\n10.0089, UNIT AND WEIGHT45"
                                    // Value of the care instruction translated in the current language (String)
            }
]
},
{
            "id": 56,               //Internal Delogue Id (Numeric)
            "name": "gunner",    // name of the care instructions output
            "value": "adas asdasd/,\ndsfsdf sdf/,\nre chaman.",
                        // Generated value of the care instruction based on the pattern set in care instructions output (String)
```

```
                "type": "Icons", // Type of the care instruction output
                "translations": []// List of translations for the care instructions output
            }
            ]
            }
            ]
    }
]
```

3.4 STEP 2: STYLE WEBHOOK SERVICE DETAIL IN DELOGUE

This will be the additional feature where an organization can provide a way to call their REST service from Delogue.
From Delogue UI, when the user clicks the 'PUSH STYLE DATA' button, this action will invoke your webhook service. The style number & export id will be sent to the webhook service. Using the style number & the export id, you can invoke the step1 service to get the style data from Delogue.

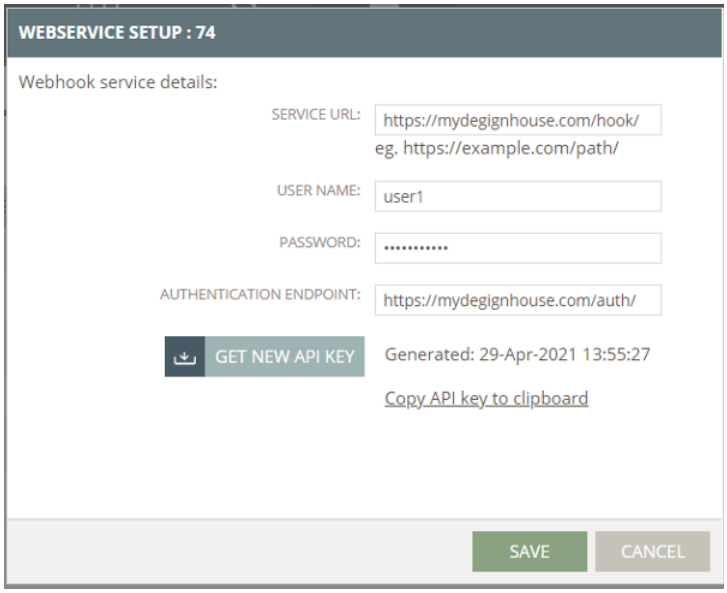
The Step 1 Delogue service needs an authentication API key which can be copied from a custom export service detail popup. The API key for authentication should be passed as a Bearer token (in the header) with input data. This way, organizations can get the style JSON data to update the information on their external system (e.g., ERP)

Webhook service:

You need to create the REST webhook service, which takes the input style number & export id as a parameter.

The service needs to have an authentication endpoint that will return the token after successful authentication. With this token, Delogue will invoke your webhook service.

The webhook service will receive the style number & export id. From the webhook service, you can invoke the Step 1 service to get style JSON data from Delogue.



You can find the SETUP buttons in Delogue under ADMIN→IMPORT/EXPORT→>CUSTOM EXPORT

Webhook service details popup contains:

- URL **(Service URL invocation details are mentioned below)**
- Authentication endpoint **(Authentication Endpoint implementation details are mentioned below)**
- User name
- Password
- Option to generate a new API key
- Copy API key

When a user clicks the **"Push Style Data"** button under styles, Delogue will call your webhook service. The Delogue will send a style number and export id. You can then use this style number to call the Delogue web service with API key authentication (mentioned in Step 1) and get the data for the style & use the style JSON data to update it in ERP.

Your webhook REST service must send a successful response immediately without blocking a Delogue request.

Authentication Endpoint implementation details

For Delogue to call your webhook service (to send Style Number and Export Id), first needs to take an authentication token.

(ie. When pushing style data button from style header). This token will be obtained by calling the authentication endpoint URL.

Request details send to the authentication endpoint url to get the token as follows

```
{ "grant_type": "password", "username": "yourusername", "password": "yourpassword" }
```

for example, call will be made using jquery as

```
$.ajax({
```

```
url: "https://exampleservice.com/auth/token",
type: "POST",
contentType: "application/json; charset=utf-8",
dataType: 'json',
cache: false,
crossDomain: true,
data: { "grant_type": "password", "username": "xxxxx@aaa.com", "password": "yyyyy" },
success: function (data) {
        alert ("Success: " +  data.access_token);
},
error: function (x, y, z) {
alert("Error" + x + '\n' + y + '\n' + z);
}
```

Expected token response from the authentication

```
{
    "access_token": "your token string",
}
```

Testing of authentication endpoint working as per expectation Delogue

Please create a blank HTML page.
Add the following code snippet to the HTML page.
Change 3 details in copied code: url , username password
Save the page and open in the browser
You should get an alert with Suncess : your token value.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
	<title>TestImportApis</title>
</head>
<script src="Scripts/jquery-1.8.2.js"></script>
<script src="Scripts/jquery-1.8.2.min.js"></script>
<script src="Scripts/jquery.validate.min.js"></script>
<script type="text/javascript">
	$(function () {
			$.ajax({
		url: "{your authentication url endpoint}",
		type: "POST",
		contentType: "application/json; charset=utf-8",
		dataType: 'json',
		cache: false,
		crossDomain: true,
		data: { "grant_type": "password", "username": "{your user name}", "password":
"{your password}" },
		success: function (data) {
			alert ("Success: " +  data.access_token);
		},
		error: function (x, y, z) {
		alert("Error: " + x + '\n' + y + '\n' + z);
		}
		});

	});
</script>
<body>
</body>
```

If you get an error message then there is something wrong with you authentication endpoint url implementation

If you get a success message then from Delogue your service will get invoked correctly

How we call your webhook service, i.e. Service Url

We will use the token from the authentication endpoint while calling your service endpoint.

An example of a call using jquery is as follows

```
 function (token) {
        $.ajax({
                url: "{Service URL}",
                type: "POST",
                contentType: "application/json; charset=utf-8",
                dataType: 'json',
data: { "StyleNumber":"00001010", "ExportId": 10 }
                cache: false,
                crossDomain: true,
                beforeSend: function (xhr, settings) { xhr.setRequestHeader('Authorization', 'Bearer '
+ {token received from authentication call}); },
                success: function (data) {
        //Success data contain
                },
                error: function (x, y, z) {
        alert('Error while fetching style json data.\n' + x.responseText);
                }
        });
```

}

3.5 INPUT JSON FORMAT FOR THE SERVICE REQUIRED ARE AS FOLLOWS:

```
{ "StyleNumber":"00001010", "ExportId": 10 }
```

Client Webhook REST service C# example:

```
public class WebhookCustomExportCommand
{
        public string StyleNumber { get; set; }
        public long ExportId { get; set; }
}

[Route("v1.0/customExportwebhooktest/")]
    [HttpPost]
    [Authorize]
    public HttpResponseMessage
WebhookCustomExportTest(WebhookCustomExportCommand command)
    {
        //....
    // Internal implementation to call step1 web service asynchronously. And return the
following
    // response immediately.
    //....
        String inputStyleNo = command.StyleNumber;
        long exportId =  command.ExportId;
        //....
        //....
```

```
                return Request.CreateResponse(HttpStatusCode.OK, "Success",
Configuration.Formatters.XmlFormatter);
            }
```

4 GETTING AN UPDATED STYLE NUMBER LIST BY GIVING A TIMESTAMP FROM DELOGUE

Delogue exposes REST-based services to import/export data into/from Delogue from the external system. To call any API you need to authenticate himself by passing API KEY as a bearer token with all requests to the Delogue system. Details of each request response are explained in the following sections of the document.

4.1 AUTHENTICATION

Exporting any data from Delogue with an API key is a 2 step process

1) Getting authentication API key from custom export - web service details popup UI.
2) Using authentication API key calling the web service

4.2 SERVICE API DETAIL

API URL⇒:

https://my.delogue.com/External/api/CustomExport/Style/Date/{fromDatetime}/{onlyReadyForExport}

Example:

- https://my.delogue.com/External/api/CustomExport/Style/Date/10-20-2020 10:40:00/false
- In this case, if false sent, then all style number lists (regardless of ready for export state) will be returned, which have been updated since the given timestamp.
- https://my.delogue.com/External/api/CustomExport/Style/Date/10-20-2020 10:40:00/true
- In this case, if "true" is sent, then all Ready for Export style number lists will be returned, which have been updated since the given timestamp
- https://my.delogue.com/External/api/CustomExport/Style/Date/10-20-2020 10:40:00
- If you do not send {onlyReadyForExport} parameter, then all style number lists will be returned, which have been updated since the given timestamp.

HTTP method type – GET

The API key from the custom export web service setting popup should be sent as a 'Bearer' key with your

input data. ex. in jquery call -

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

style/date– it fetches style number list which are updated from datetime timestamp given

date/{fromDatetime}- fromDatetime is a UTC timestamp from which you need to get updated style number list data.

This will need to be in a valid accepted datetime format as "MM-dd-yyyy HH:mm:ss" in 24 hour format.

Example:

- for 20 OCT 2020 date & time 03:40 PM it is like ⇒ 10-20-2020 15:40:00
- for 20 OCT 2020 date & time 10:40 AM it is like ⇒ 10-20-2020 10:40:00

Example service invoke call is like: (for timestamp 10-20-2020 10:40:00)

```
$.ajax({
        url: "https://my.delogue.com/External/api/CustomExport/Style/Date/10-20-2020 10:40:00",
        type: "GET",
        contentType: "application/json; charset=utf-8",
        dataType: 'json',
        cache: false,
        crossDomain: true,
        beforeSend: function (xhr, settings) {
        xhr.setRequestHeader('Authorization', 'Bearer ' + api key); }
,
success: function (styleListData) {
        alert("Success : Updated style number list are : \n" + JSON.stringify(styleListData));
        },
        error: function (x, y, z) {
          alert('Error while getting Updated style number list : \n' + x.responseText);
        }
})
```

On success: Call will return in success with information on style number list

On error: If API fails, you get an error message in response text with a bad request status.

4.3 RESPONSE FROM API STYLE JSON DATA OBJECT STRUCTURE

Response JSON like below, list of style numbers

["10000",
"10022",
"10021",
"10001",
"10010",
"10005",
"10011",
"10015"]

The following action will be registered & considered as a change in style for update through API:

Header:

- add/change the style header image
- select another brand
- select another brand contact person
- change style number
- add/change the style name
- add/change the description
- change state
- change ready for export (from style header UI & also if "Reset after Export" is set on the FTP setup)
- add/change supplier
- select another supplier contact person (with both designer & supplier login)
- change season/project
- add/change group
- add/change style categories
- copy style with supplier change
- style state change to WIP/published from sample request update dialog (Set on/off setting of WIP state)
- Import style header picture/pictures logo from Admin
- Change Primary State of the supplier in the style header

- Update above fields from the Custom report tab

Item list:

- add color
- delete color
- deactivate color
- activate color
- add item/items
- import (add/replace) item list
- style item quantity update

Measurement chart:

- add/change size range
- deactivate size
- activate size
- import measurement chart
- Link measurement chart

Custom fields:

- add/change the data: style level
- add/change data: style/color level
- add/change data: style/size level
- update custom field value from Custom Report tab

Prices:

- add/change data
- change price per style / color / size level
- change future prices
- change status approved / not approved
- update price field values from the Custom report tab

5 API FOR GETTING STYLE NUMBERS PER BRAND AND SEASON

5.1 INTRODUCTION

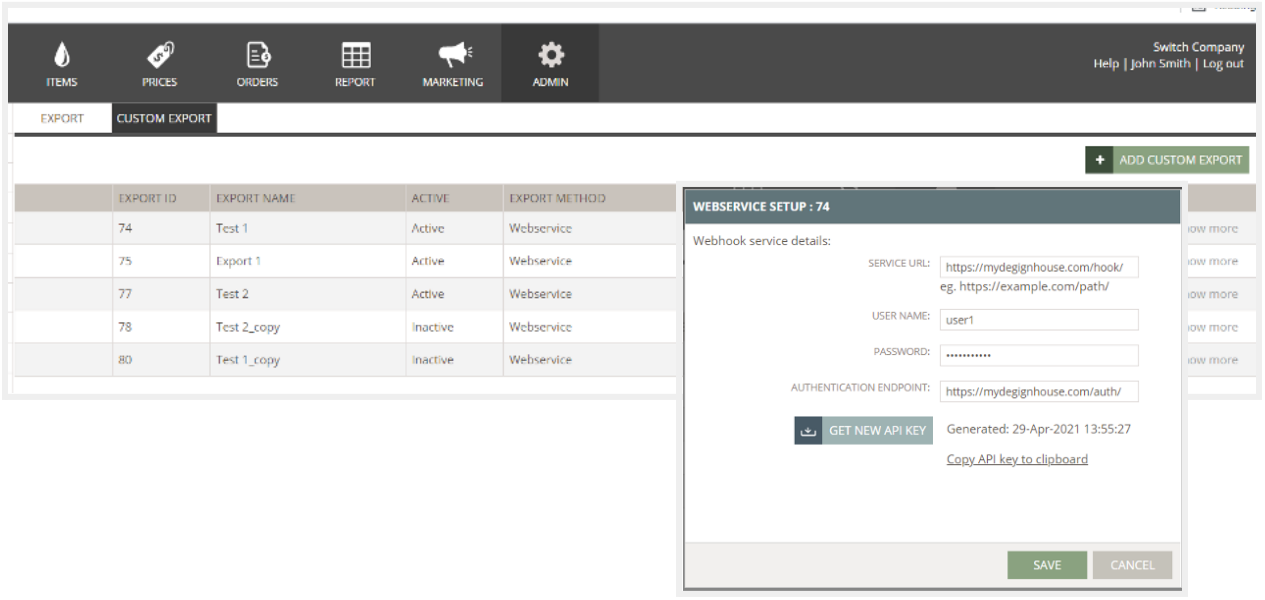
This API takes a list of brand ids and season ids and gives back a list of styles that use the specified brands and seasons. There is also an option to pass a boolean variable to only get the styles set to Ready for Export.

5.2 AUTHENTICATION

It uses API Key authentication. You need to get the API key from a custom export setup and use this API key for authentication with this API.

Getting the API Key

To use the export feature, you must configure your webhook details in Delogue. This configuration can be done in Admin sections under Import/Export settings(ADMIN→IMPORT/EXPORT→>CUSTOM EXPORT).
The webhook configuration pop-up can be opened by clicking on the setup button for the export.



After clicking on the setup button, the webhook settings pop-up will show. Here is how the webhook configuration pop-up looks like:

You will need to copy the API key from this pop-up for authentication purposes. The API key will typically be of the following format.

IAp74oCL4oCL4oCL4oCL4oCLCiJPcmdhbml6YXRpb25JZCI6ICIyIiwKIkV4cG9ydElkIjogIjEiLAoiQXBpS2V5QmFzZSI6ICIyMjItMjIyLTIyMi0yMjIiCn3igIvigIvigIvigIvigIsK

Please save this API key, as this is required to be passed as a header in all the export requests to Delogue.

Note: There is an option to generate the API key again, which will invalidate the existing API key and create a new one. In case the new API key is generated using this, the old API key will stop working for export, and the new one should be used instead.

Using the API Key

The API key generated in the previous step should now be passed as an HTTP authorization header as a bearer token to the export requests in Delogue.

Authorization: Bearer IAp74oCL4oCL4oCL4oCL4oCLCiJPcmdhbml6YXRpb25JZCI6ICIyIiwKIkV4cG9ydElkIjogIjEiLAoiQXBpS2V5QmFzZSI6ICIyMjItMjIyLTIyMi0yMjIiCn3igIvigIvigIvigIvigIsK

5.3 API OVERVIEW

Endpoint URL: my.delogue.com/External/api/CustomExport/Style/BrandsAndSeasons
HTTP Method: POST
Request Body:

```
{
	"SeasonIds": ["AW2020", "1678"],
	"BrandIds": ["aa", "12w"],
	"OnlyIncludeReadyForExport": "true"
}
```

Request Headers

{
	"Content-Type": "application/json",
	"Authorization": "Bearer <API_KEY>"

}
Note: Replace <API_KEY> with your actual API key.

Response: The API returns a list of style numbers, which are represented as strings in an array.
["one", "two"]

Example Request
POST my.delogue.com/External/api/CustomExport/Style/BrandsAndSeasons HTTP/1.1
Content-Type: application/json
Authorization: Bearer <API_KEY>
{
"SeasonIds": ["AW2020", "1678"],
"BrandIds": ["aa", "12w"],
"OnlyIncludeReadyForExport": "true"
}

5.4 MANUAL PUSH OF STYLE NUMBERS PER BRAND AND SEASON

Instead of calling the API for style numbers per brand and season, it is possible to click on the button “MANUAL PUSH” under ADMIN → IMPORT/EXPORT → CUSTOM EXPORT. Now you will see a popup where you can select which brands and seasons you would like to get style numbers from. When you are satisfied with you selection you can click the button “EXPORT NOW” and Delogue will now start to send the style numbers one by one to your Webhook.

6 EXPORTING CUSTOM EXPORT STYLE DATA FROM DELOGUE VIA FTP

For the Import/Export role in the Admin tab, there is a tab of Import/Export. When you select this option, you will get a custom export sub-tab. In this tab, you can set details of custom export and FTP details as explained below:

1. Create a new custom export row.
2. In the custom export row, select the export method as FTP from the dropdown and save.
3. You will now see a new button called ‘Setup’ on the same custom export row. Detail of the button click is mentioned below in point ‘Setting up FTP Server’ of this document.

6.1 MARKING STYLES AS READY FOR EXPORT

Users with an Integrator role can set a style to “Ready for Export” on the styles. When the style is set to “Ready for Export,” it will be exported the next time a scheduled FTP export runs.

6.2 SETTING UP FTP SERVER

Customers can set up an FTP server to export their product data from Delogue. The setup will require:

- FTP TYPE - In current situations, ftp type is ftp only. In the future there can be sFTP added.
- FORMAT - Format is export file format, ie json or XML
- FTP URL (e.g. ftp://example.com:[portno]/path)
- Credentials (username & password),
- Limit of styles per file
- Setting option to make the uncheck ready for export state after FTP export from export now export
- Setting option to make single style FTP export from style header
- Setting option to make the uncheck ready for export state after FTP export from style header export.

6.3 OPTIONS FOR FTP EXPORT

Manually via export now: When a customer clicks on export now then FTP custom export process will start. On FTP location all the ready for export styles will be exported. The status of FTP export will be notified via email. This option will also work if current custom export is in an inactive state.

Manually via style header: From the style header you can click on the PUSH STYLE DATA button and the FTP custom export process will start. The single style data will be exported on the FTP location. The status of FTP export will be notified via email.