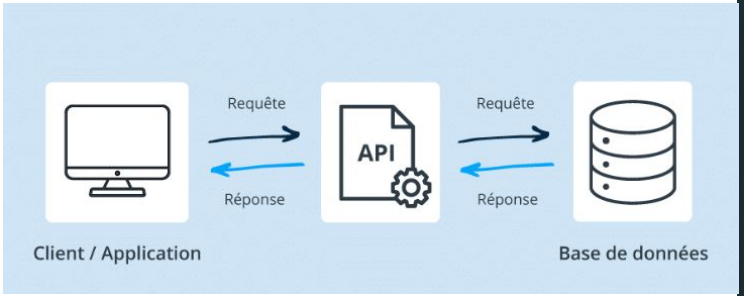
EKL APIs

By: Robin Bredemus

Contents

Qu’est ce qu’une API?

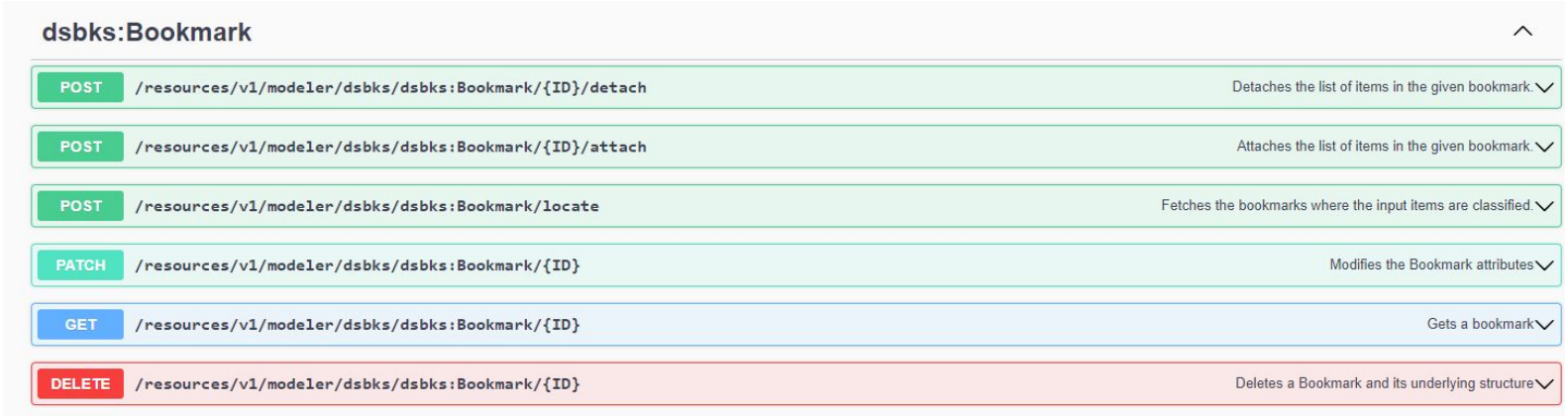


1. Lien entre une base de données et une application
2. Toute requete -> Lecture (**Get**), Manipulation (**Post**), Modification (**Patch/Delete**)
3. Surtout pour client léger, mais peut etre demander pour obtenir des données non natif sur le client lourd.
4. Example, pour les bookmarks, purement des **données web**, donc obligatoire de passer par des APIs pour obtenir les data.
5. Un **CSRF token** est necessaire pour toute requete API de PLM compass de type HTTP. Il est obtenu via un API (A voir plus tard dans la présentation)
6. Dans les requêtes API, les en-têtes de requête **(request headers)** sont utilisés pour fournir des informations supplémentaires permettant à un serveur de traiter une requête API.

Comment les trouver

Disponible sur documentation Dassault, example bookmarks -> https://media.3ds.com/support/documentation/developer/R2023x/en/DSDoc.htm?show=CAABookmarkWS/BookmarkAPI1_v1.htm

Tous les types d'appels possible sur une donnée:



Comment construire un API

1. Il faut déclarer un **fichier JSON** (fichier de transport de data)
2. Pour savoir comment le déclarer, voir la page documentation Dassault de l'API
3. Pour la forme du JSON et comment le construire, voir Comment les déclarer en EKL (2/3)

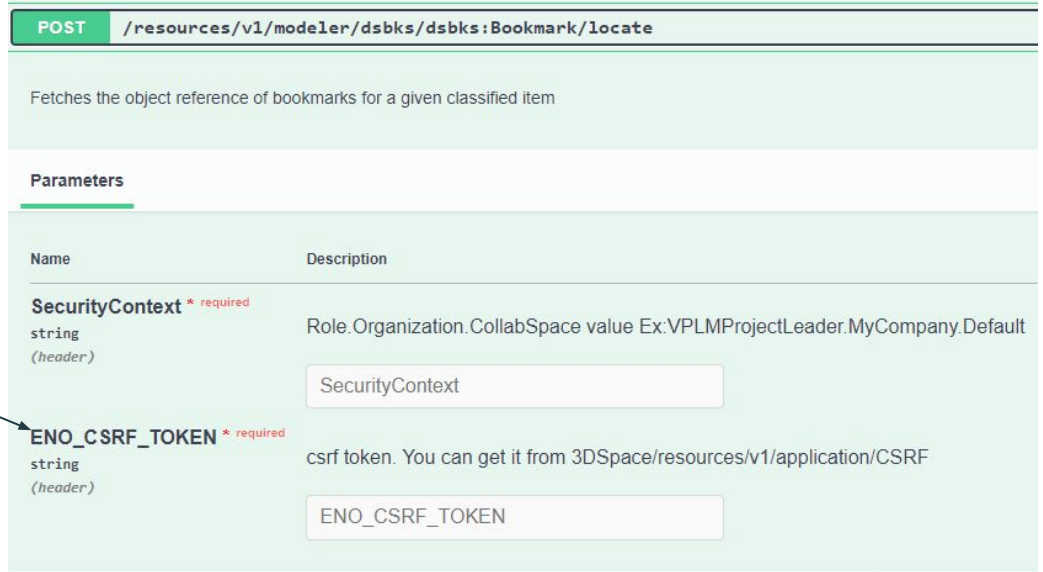
E.g Pour fetch un bookmark, le Json doit avoir la forme de ->

```
jsonBody = "[{"
jsonBody = jsonBody + "\"identifier\": \""+prdPID+"\","
jsonBody = jsonBody + "\"relativePath\": \"/resources/v1/modeler/dseng/dseng:EngItem/"+prdPID+"\","
jsonBody = jsonBody + "\"type\": \""+prdToInstanciate.PrimaryType.Name+"\","
jsonBody = jsonBody + "\"source\": \""+server_url+"\""
jsonBody = jsonBody + "}]"
```

Pre-requis

Pour savoir de quoi on va avoir besoins, referencer la doc Dassault ->

Pour la requête API que l'on veut, il va nous falloir un SecurityContext (c'est bon) mais aussi un CSRF token, obligatoir pour ce genre de requête.



Obtenir token CRSF en EKL (1/2)

First il va falloir obtenir un **token csrf** pour utiliser plus tard

Premièrement il faut obtenir le **serverurl** du client actuel (Tous des strings):

```
server_url = GetSystemInfo("serverurl")
user_name = GetSystemInfo("username")
security_context = GetSystemInfo("securitycontext")
```

Next, créer un **client HTTP** pour préparer la requête

```
let client(HTTPClient)
client = CreateHTTPClient()
```

Définir les **headers du client** comme ci dessous ->

```
client.AddRequestHeaders("Content-Type: application/json")
client.AddRequestHeaders("Accept: application/json")
```

Obtenir token CRSF en EKL (2/2)

Enfin on peut définir le **URI du client** (Identificateur unique permettant la création d'un API specific via un path et un url defini). Il est composé du server_URL et d'un path a retrouver dans la doc Dassault.

```
iURI = server_url +"/resources/v1/application/CSRF"
```

On peut ensuite faire une **requête API** sur le client, le résultat est en forme DataTreeNode

```
let oDTN(DataTreeNode)
let oBuffer(String)
oBuffer = client.Get(iURI, oDTN)
```

On attend donc une **réponse** du client en forme de **ReturnCode**. Si == 0 -> OK sinon KO

```
if (client.ReturnCode == 0)
```

Enfin on accède à nos données dans la réponse **oDTN** via la méthode Access (Voir plus tard dans la présentation)

```
if (client.ReturnCode == 0)
{
    // Get the csrf token with Access method
    dt = oDTN->Access("csrf\value", "DataTreeNode")
    csrf_token = dt.String
}
```

Déclarer une API en EKL (1/3)

Avec le token CRSF obtenu, on va pouvoir faire notre **API**.

First on recréer notre **client**:

```
let client(HTTPClient)
client = CreateHTTPClient()
```

Next, on défini les **headers du client**, plus précis cet fois avec la token CRSF

```
client.AddRequestHeaders("Content-Type: application/json")
client.AddRequestHeaders("Accept: application/json")
client.AddRequestHeaders("SecurityContext: "+security_context)
client.AddRequestHeaders("ENO_CSRF_TOKEN: "+csrf_token)
```

On declare le nouveau URI (Correspond au liens dans la doc Dassault)

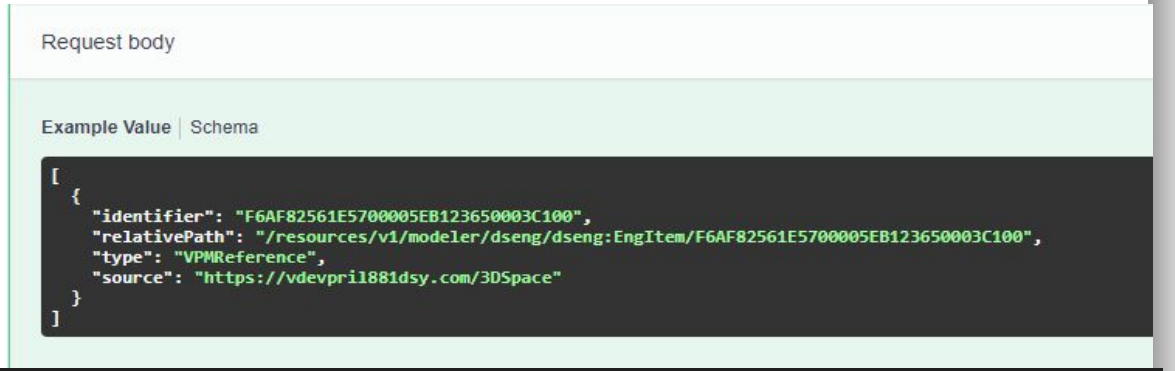
```
iURI = server_url +"/resources/v1/modeler/dsbks/dsbks:Bookmark/locate"
```

Comment les déclarer en EKL (2/3)

Next on va devoir définir le **format du JSON** (Contenu de la requete API.

Pour ca on suit exactement le **format de la doc Dassault**:

Comme on peut le voir, chaque ligne correspond directement au **schéma exemple** du fichier JSON retrouver dans la documentation Dassault. Le prdPID est le physical ID de la pièce dans une db obtenu par une autre API (Voir appendix pour comment l'obtenir).




```
jsonBody = "[{"
jsonBody = jsonBody + "\"identifier\": \""+prdPID+"\","
jsonBody = jsonBody + "\"relativePath\": \"/resources/v1/modeler/dseng/dseng:EngItem/"+prdPID+"\","
jsonBody = jsonBody + "\"type\": \""+prdToInstanciate.PrimaryType.Name+"\","
jsonBody = jsonBody + "\"source\": \""+server_url+"\""
jsonBody = jsonBody + "}]"
```

Déclarer une API en EKL (3/3)

Une fois le Json construit et notre URI déclaré on va pouvoir faire notre requête ->

```
oBuffer = client.Post(iURI, "HTTPDEFINED", jsonBody, oDTN)
```

Si le return code est bon (==0) on va pouvoir ouvrir nos données et les manipuler avec la méthode query

```
if (client.ReturnCode == 0)
{
    // Get the Bks infos with Query/Access method
    dtList = oDTN->Query("DataTreeNode", "x.Name==\"member\"")
```

Comment lire les réponses d'un API (1/2)

Le format des réponses des API sont toute disponible dans la documentation Dassault, par exemple pour la requête que l'on vient de faire on obtient ->

Example Value


```
{
  "totalItems": 1,
  "member": [
    {
      "id": "F6AF82561E5700005EB123650003C100",
      "bookmarks": {
        "totalItems": 1,
        "member": [
          {
            "source": "https://3dexperience.com/3DSpace",
            "type": "dsbks:Bookmark",
            "identifier": "F6AF82561E5700005EB123650003C101",
            "relativePath": "/resources/v1/modeler/dsbks/dsbks:Bookmark/F6AF82561E5700005EB123650003C101"
          }
        ]
      }
    }
  ]
}
```

Comment lire les réponses d'un API (2/2)

Une fois la réponse lu avec la function query dans un format de list de DataTreeNode ->

```
if (client.ReturnCode == 0)
{
    // Get the Bks infos with Query/Access method
    dtList = oDTN->Query("DataTreeNode", "x.Name==\"member\"")
```

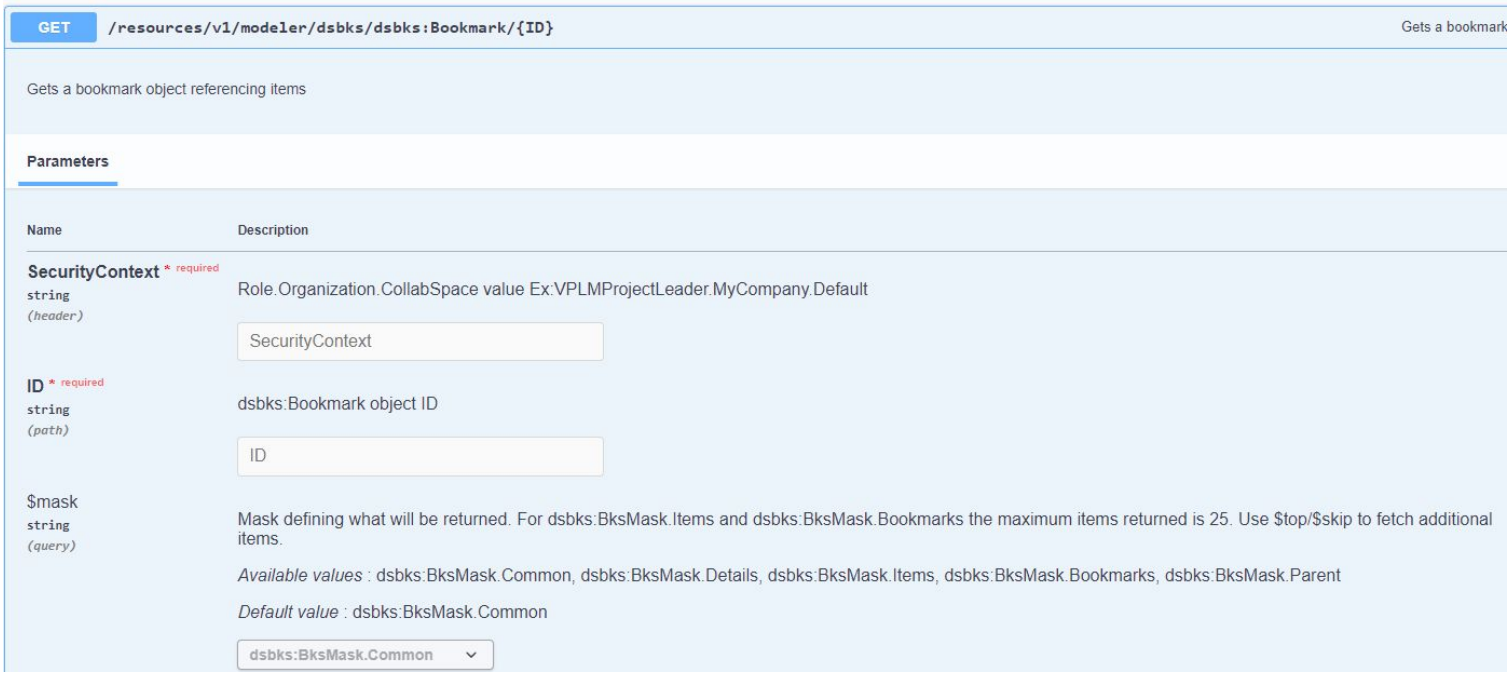
Connaissant le format de l'output, si on veut par exemple l'identité de tous les résultats de notre request, on fait simplement un Access ->

```
if (client.ReturnCode == 0)
{
    // Get the Bks infos with Query/Access method
    dtList = oDTN->Query("DataTreeNode", "x.Name==\"member\"")
    for dt inside dtList
    {
        dtPID = dt->Access("identifier", "DataTreeNode")
        bksPID = dtPID.String
```

Et on a maintenant le ID du bookmark!

Masks (1/3)

Les masks permettent de choisir le format de réponse sur une requête GET d'un API. Avec le ID du bookmark on peut continuer de rechercher des infos dessus ->



Masks (2/3)

Pour cet requête on créer donc le client et les headers comme avant :

```
client = CreateHTTPClient()

// Defines Headers
client.AddRequestHeaders("Content-Type: application/json")
client.AddRequestHeaders("Accept: application/json")
client.AddRequestHeaders("SecurityContext: "+security_context)
```

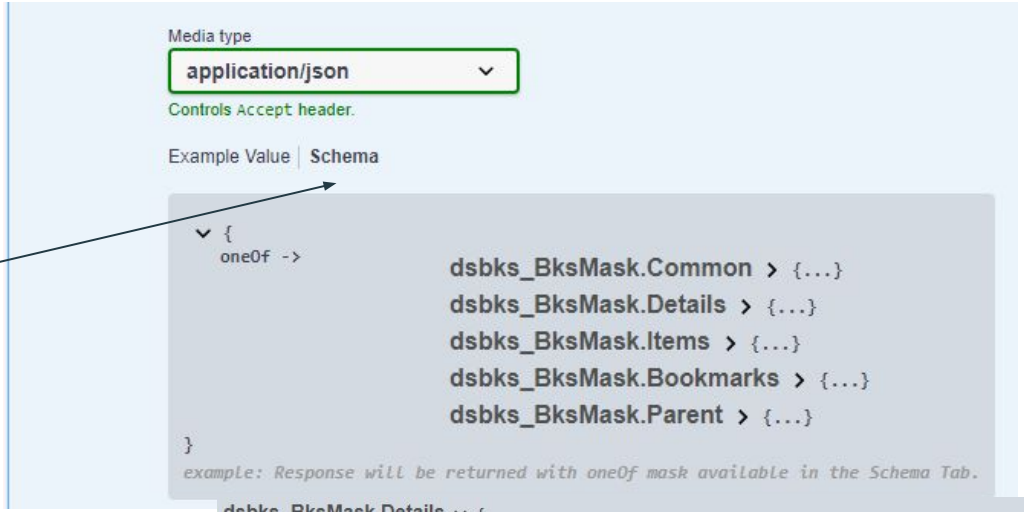
Mais la déclaration du mask se fait au niveau du URI

```
iURI = server_url +"/resources/v1/modeler/dsbks/dsbks:Bookmark/"+ bksPID+"/?$mask=dsbks:BksMask.Details&$fields=dsmveno:CustomerAttributes"
```

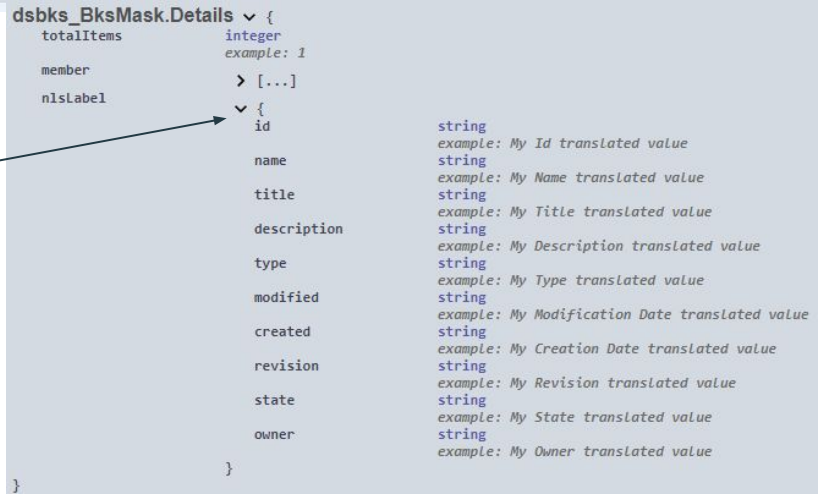
Le field défini quel parti du masque montrer ou updater pour éviter de charger plus que le nécessaire et augmenter les perfos

Masks (3/3)

Pour choisir un mask, on prends la requête API sur lequel on souhaite, on navigue jusqu'au réponses et on clique sur Schema.



En cliquant sur la flèche on peut voir à quoi ressemblerait la réponse avec ce masque.



Appendix

Obtenir le prdID d’un VPM ref:

```
client = CreateHTTPClient()

// Defines Headers
client.AddRequestHeaders("Content-Type: application/json")
client.AddRequestHeaders("Accept: application/json")
client.AddRequestHeaders("SecurityContext: "+security_context)
externalId = prdToInstanciate.PLM_ExternalID
// Defines URI and Body
let strPrdPId (String)

iURI = server_url +"/resources/issues/objects/from-tnr-to-physical-id"

jsonBody = "{"
jsonBody = jsonBody + "\"type\": \""+prdToInstanciate.PrimaryType.Name+"\","
jsonBody = jsonBody + "\"name\": \""+prdToInstanciate.PLM_ExternalID+"\","
jsonBody = jsonBody + "\"revision\": \""+prdToInstanciate.revision+"\""
jsonBody = jsonBody + "}"
Trace(2, "jsonBody: #", jsonBody)

// Example1: Performs a POST request
oBuffer = client.Post(iURI, "HTTPDEFINED", jsonBody, oDTN)
Trace(2, "oBuffer: #", oBuffer)

// Checking if request return code is OK (rc = 0)
if (client.ReturnCode == 0)
{
    // Get the ticket with Access method
    dtPID = oDTN->Access("body\physicalId", "DataTreeNode")
    prdPID = dtPID.String
```