1 First install the bsSDD-example from github and add the DLL's for Microsoft.Identity.Client and Newtonsoft.Json

2 My one-hour-project is „Extract the property-sets for IfcPipeSegment"

2.1 Old style for searching properties of IfcPipeSegment:

```
..\IFC4x3_RC2-psd>dir *PipeSegment*
30.12.2020  17:17             3.873 Pset_PipeSegmentOccurrence.xml
30.12.2020  17:17             2.133 Pset_PipeSegmentPHistory.xml
30.12.2020  17:17            11.866 Pset_PipeSegmentTypeCommon.xml
30.12.2020  17:17             2.277 Pset_PipeSegmentTypeCulvert.xml
30.12.2020  17:17             2.345 Pset_PipeSegmentTypeGutter.xml
```

Inside Pset_PipeSegmentTypeCommon.xml:

<?xml version="1.0"?>
<PropertySetDef xmlns="**http://buildingSMART-tech.org/xml/psd/PSD_IFC4.xsd**" xsi:noNamespaceSchemaLocation="**http://buildingSMART-tech.org/xml/psd/PSD_IFC4.xsd**" templatetype="**PSET_TYPEDRIVENOVERRIDE**" ifdguid="**8bc49580d1f311e1800000215ad4efdf**" xmlns:xsd="**http://www.w3.org/2001/XMLSchema**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**"><IfcVersion version="**IFC4x3_RC2**"/><Name>Pset_PipeSegmentTypeCommon</Name><Definition>Pipe segment type common attributes.</Definition><Applicability/><ApplicableClasses><ClassName>IfcPipeSegment</ClassName></ApplicableClasses><ApplicableTypeValue>IfcPipeSegment</ApplicableTypeValue><PropertyDefs><PropertyDef ifdguid="**9121e000d1f311e1800000215ad4efdf**"><Name>Reference</Name><Definition>Reference ID for this specified type in this project (e.g. type 'A-1').</Definition><PropertyType><TypePropertySingleValue><DataType type="**IfcIdentifier**"/></TypePropertySingleValue></PropertyType><NameAliases><NameAlias lang="**en**">Reference</NameAlias><NameAlias lang="**ja-JP**">参照記号</NameAlias></NameAliases><DefinitionAliases><DefinitionAlias lang="**en**"/><DefinitionAlias lang="**ja-JP**">このプロジェクトにおける参照記号(例：A-1)。分類コードではなく内部で使用されるプロジェクトタイプとして使用されるもの。</DefinitionAlias></DefinitionAliases></PropertyDef><PropertyDef ifdguid="**97b05780d1f311e1800000215ad4efdf**"><Name>Status</Name><Definition>Status of the element, predominately used in renovation or retrofitting projects. The status can be assigned to as "New" - element designed as new addition, "Existing" - element exists and remains, "Demolish" - element existed but is to be demolished, "Temporary" - element will exists only temporary (like a temporary support structure).</Definition><PropertyType><TypePropertyEnumeratedValue>

<EnumList name="**PEnum_ElementStatus**">
<EnumItem>NEW</EnumItem>
<EnumItem>EXISTING</EnumItem>
<EnumItem>DEMOLISH</EnumItem>
<EnumItem>TEMPORARY</EnumItem>
<EnumItem>USERDEFINED</EnumItem>
<EnumItem>NOTKNOWN</EnumItem>
<EnumItem>NEW</EnumItem>
<EnumItem>EXISTING</EnumItem>
<EnumItem>DEMOLISH</EnumItem>
<EnumItem>TEMPORARY</EnumItem>
<EnumItem>OTHER</EnumItem>
<EnumItem>NOTKNOWN</EnumItem>
<EnumItem>UNSET</EnumItem>
</EnumList>
...

2.2 Retrieving the same from bSDD

http://identifier.buildingsmart.org/uri/buildingsmart/ifc-4.3/class/**ifcPipeSegment**
with the same result from
https://bs-dd-api-prototype.azurewebsites.net/api/Classification/v2?namespaceUri=http%3A%2F%2Fidentifier.buildingsmart.org%2Furi%2Fbuildingsmart%2Fifc-4.3%2Fclass%2F**ifcPipeSegment**

Result No.1: No Enums PEnum_ElementStatus

So in the c#-example I changed line 38 to
public static string ApiEndpoint ="https://bs-dd-api-prototype.azurewebsites.net/api/Classification/v2?namespaceUri=http%3A%2F%2Fidentifier.buildingsmart.org%2Furi%2Fbuildingsmart%2Fifc-4.3%2Fclass%2FifcPipeSegment&includeChildClassificationReferences=true";

Result No.2: var searchResult = JsonConvert.DeserializeObject<**SearchResultContract**>(resultText);
The Contract for the new JSON-Format must be changed!

2.3 New start with direct HttpWebRequest (1)

```
using System;using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Net;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;


class bSDD_test {static void Main(string[] args) {


string EntityName="ifcPipeSegment";
string propertySet="Pset_PipeSegmentTypeCommon"; // or null for all Pset's;


string RequestUrl=@"https://bs-dd-api-
prototype.azurewebsites.net/api/Classification/v2?namespaceUri=http%3A%2F%2Fidentifier.buildingsmart.org%2Furi%2Fbuildingsmart%2Fifc-
4.3%2Fclass%2F"+EntityName+"&includeChildClassificationReferences=true";
string Response = new StreamReader(((HttpWebResponse)((HttpWebRequest)WebRequest.Create(RequestUrl)).GetResponse()).GetResponseStream()).ReadToEnd();


// make happy, if it works with the right contract
// dynamic ContractResult= JsonConvert.DeserializeObject<SearchResultContract> (Response);
// Console.WriteLine(ContractResult);


//Console.WriteLine(Response); // dont't make happy:
```

2.3 New start with direct HttpWebRequest (2)

```
// also dont't make happy, but give the first results:
dynamic ResultLevel0= JsonConvert.DeserializeObject(Response); StreamWriter sw=new StreamWriter("test.json");sw.WriteLine(ResultLevel0); sw.Close(); //structured output!
foreach (var ResultLevel1 in ResultLevel0)
  foreach (var ResultLevel2 in ResultLevel1)
    if (ResultLevel2 is JArray)
      if ( ((JArray)ResultLevel2).Count>0)
        foreach (var ResultLevel3 in ResultLevel2)
           if (ResultLevel3 is JObject)
             if ( ((JObject)ResultLevel3).GetValue("propertySet")!=null)
               if ( propertySet==null || ((JObject)ResultLevel3).GetValue("propertySet").ToString()==propertySet )
                  Console.WriteLine(((JObject)ResultLevel3).GetValue("propertySet").ToString()+"."+((JObject)ResultLevel3).GetValue("name")+":"+((JObject)ResultLevel3).GetValue("dataType")+"//"+((JObject)ResultLevel3).GetValue("description"));
}}
```

2.4 Result

```
Pset_PipeSegmentTypeCommon.InnerDiameter:real//The actual inner diameter of the pipe.
Pset_PipeSegmentTypeCommon.NominalDiameter:real//The nominal diameter of the pipe segment.
Pset_PipeSegmentTypeCommon.OuterDiameter:real//The actual outer diameter of the pipe.
Pset_PipeSegmentTypeCommon.PressureRange:real//Allowable maximum and minimum working pressure relative to ambient pressure .
Pset_PipeSegmentTypeCommon.Reference:string//Reference ID for this specified type in this project e.g. type A 1 , provided, if there is no classification reference to a recognized classification system used.
Pset_PipeSegmentTypeCommon.Status:string//Indicates an error code or identifier, whose meaning is specific to the particular automation system. Example values include ConfigurationError , NotConnected , DeviceFailure , SensorFailure , LastKnown, CommunicationsFailure , OutOfService .
Pset_PipeSegmentTypeCommon.TemperatureRange:real//Temperature range within which the air terminal is designed to operate.
Pset_PipeSegmentTypeCommon.WorkingPressure:real//Boiler working pressure.
Drücken Sie eine beliebige Taste . . .
```

3 Conclusion

Yes, it is possible, to access the property-set-templates from bSDD instead of the XML-files.
The advantage is to access the information from a single allways up to date source.
The Pset-XML-Files seem to be not fully included in the bSDD until now.
I offer my help to transform the XML-Psets to bSDD.
A tool, that export back from bSDD to XML-Psets (or similar) could be a way to Check the completeness