Signifikant XML import

Technical instruction

Document history

Date	Version	Description	Author
2015-06-23	1.0	First version	Muhammad Saqib
2015-09-18	1.1	Transforms added	Mattias Löfstrand
2016-10-19	1.2	Import APIs added	Kenneth Jonsson
2017-02-20	1.3	Name change	Mattias Löfstrand

Contents

1 Introduction 4
1.1 Overview 4
2 Configuration 4
2.1 Sample config file 5
3 APIs 6
3.1 GUI for testing APIs 6
3.2 APIs for transform 6
3.3 APIs for import 6
3.4 APIs for publish 6
4 XML format 6
4.1 Id handling 6
4.2 Objects 7
4.3 Attachments 15
4.4 Reference strings 15

Appendices

-

References

[1] Technical instruction - Configuring Web Viewer

1 Introduction

This document is intended for technical staff creating imports to Signifikant Platform and describes the Signifikant XML import format.

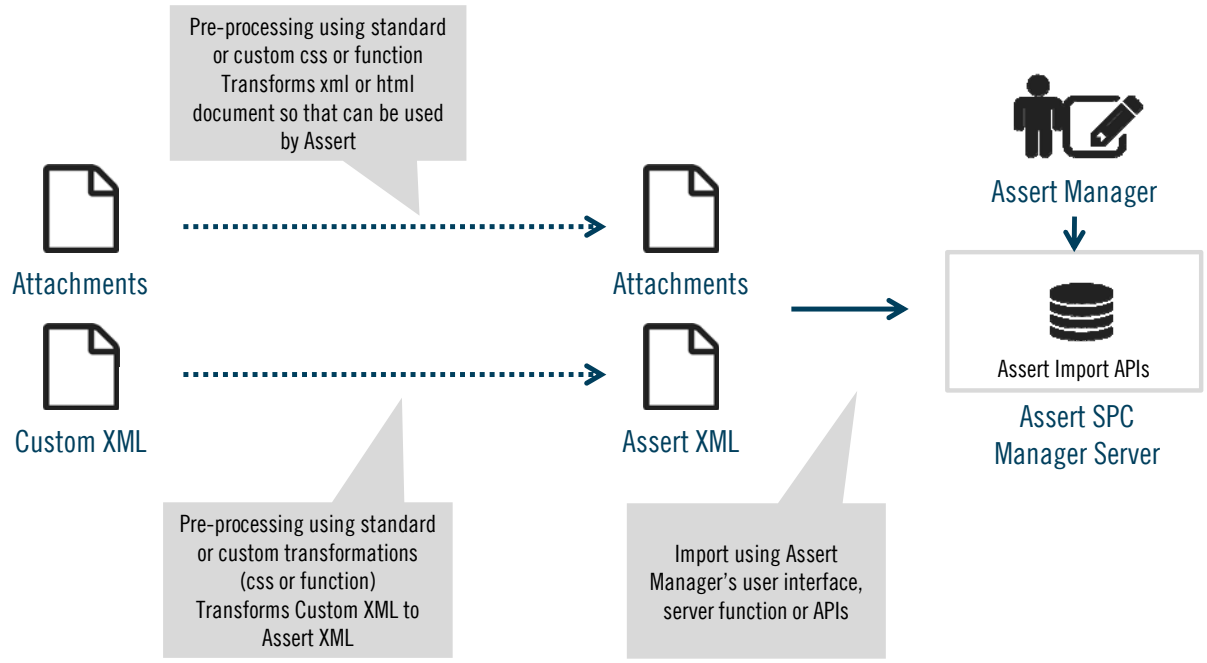
1.1 Overview

Signifikant Platform import process is a two-step process where custom file formats are transformed into Signifikant XML import format and then imported.

Signifikant Platform import function operates on Signifikant XML format described in this document, and has configuration functionality for applying a transformation from any custom xml format. Signifikant Platform import function has the following built in transformations:

- Creo Illustrate xml to Signifikant XML
- IsoDraw xml (companion files) to Signifikant XML
- Windchill xml (requires configuration of Windchill) to Signifikant XML
- Arbortext and Simonsoft CMS to Signifikant XML

Signifikant Platform import function may also apply style sheets to xml-documents during import creating html-versions of the documents.



2 Configuration

Signifikant Platform's standard function is controlled by import.config, which lists the directories to use and the transforms to apply. Each information type will need to reside in their own directory.

E.g. an external application exports parts and pdf-documents. The config file will specify where these exports shall be placed and how Signifikant Platform should recognize these exports. The config file will also specify where the Signifikant XML files shall be placed and where transforms are to be found.

```
D:/Export/accessories_1

D:/Export/pdf-documents_1

D:/Import/Accessories_1

D:/Import/pdf-documents_1
```

When the first step of the import is started, Signifikant Platform will look in the export directory specified in the config file and match the folders found according to the regex in the config file. The information found will be transformed and placed in the import directory along with a .lis file specifying the content.

The second step, the actual import, will look in the import directory and match he folders with the regex in the config file and the actual files listed in the .lis file to determine what to import. When the second step of the import is completed, Signifikant Platform will place a status.xml file in the import directory.

Batch size will split import file in parts to ensure no xml file gets too big. A reasonable size is to set BatchSizeMB to 20.

2.1 Sample config file

```
<?xml version="1.0" encoding="utf-8"?>
<ImportConfiguration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <ExportBasePath>D:\Export</ExportBasePath>
      <ImportBasePath>D:\Import</ImportBasePath>
      <StyleSheetsBasePath>C:\ProgramData\Signifikant\Assert\Customize</StyleSheetsBasePath>
      <ImportSetting>
            <UseCache>false</UseCache>
            <LoadCache>false</LoadCache>
            <AutoAddNewLanguage>true</AutoAddNewLanguage>
            <StopOnError>false</StopOnError>
            <MailEnabled>true</MailEnabled>
            <MailFromManager>true</MailFromManager>
            <MailFromAPI>true</MailFromAPI>
            <MailAtSuccess>true</MailAtSuccess>
            <MailAtFailure>true</MailAtFailure>
            <MailRecipient>malo@signifikant.se</MailRecipient>
            <MailSender>noreply@signifikant.se</MailSender>
      </ImportSetting>
      <TransformerSettingList>
            <TransformerSetting>
                  <Name>catalogues</Name>
                  <FolderRegex>catalogues_</FolderRegex>
                  <SequenceNumber>1</SequenceNumber>
                  <BatchSizeMB>0</BatchSizeMB>
                  <BatchCounter>0</BatchCounter>
            </TransformerSetting>
            <TransformerSetting>
                  <Name>pdf-documents</Name>
                  <FolderRegex>pdf-documents_</FolderRegex>
                  <SequenceNumber>2</SequenceNumber>
                  <BatchSizeMB>20</BatchSizeMB>
                  <BatchCounter>0</BatchCounter>
            </TransformerSetting>
      </TransformerSettingList>
</ImportConfiguration>
```

3 APIs

APIs are normally custom made. The below APIs are provided to control import and publish.

3.1 GUI for testing APIs

Use the below web page to access and test APIs.

http://localhost/AssertServer/PublisherView.cshtml

3.2 APIs for transform

Initiate a transformation and check status by calling

```
http://localhost/AssertServer/api/Publisher/Transform/<folder>/<language>
http://localhost/AssertServer/api/Publisher/Status/<site>/Transform/<folder>
```

3.3 APIs for import

Initiate an import and check status by calling

```
http://localhost/AssertServer/api/Publisher/Import/<site>/Transform/<folder>
http://localhost/AssertServer/api/Publisher/Status/<site>/Import/<folder>
```

3.4 APIs for publish

Initiate publish and check status by calling

```
http://localhost/AssertServer/api/Publisher/Publish/<site>/<publicationlabel>
http://localhost/AssertServer/api/Publisher/Status/<site>/Publish/<publicationlabel>
```

4 XML format

4.1 Id handling

XML document may contain list of objects and objects may have different type of Ids (logical, Business, Persistent). Within XML document two Ids can be used for identification of object for import process.

1. Id
2. Persistent-id

These ids may be used for referring the objects. It is not mandatory for the objects to have Ids for import process, but if objects will be referred then either id (id or persistent-id) is mandatory.

4.1.1 Id

All objects may have an *id* that only exist within the scope of an import to make it easier to refer to the object within the file.

4.1.2 Persistent-id

The use of persistent-id helps to avoid duplication because if there will be an object with persistent-id import will update that object instead of import again. If persistent-id is not

defined then object will be treated as new object and new copy of the object will be created. Persistent-id not applied to all objects it applied to the following objects.

Text, part, part assembly, bulletin, specification type, catalogues, catalogue nodes, illustration, document, brand.

4.1.3 Business-id

Finally, objects of type presentation (part, part assembly, document, catalogue and illustration) may also have a *business-id* which may be used to refer to that object in an easy and readable way, e.g. part number, product id or document number. For object type parts, *number* is used as an alias for *business-id*.

4.2 Objects

Signifikant Platform import format supports the below object types when importing information into Signifikant Platform. Object names are case insensitive to avoid errors. All attributes of objects are also case insensitive.

Object type	Description
<Settings>	It defines the settings in import context.
<Text>	Translated texts.
<SpecificationType>	Defines specifications that may be used on any presentation.
<Specification>	Object defines the specifications, which can contain the different values.
<Part>	A presentation object used to describe parts.
<PartAssembly>	A presentation object used to describe part assemblies.
<Catalogue>	A presentation object used to describe catalogues.
<CatalogueNode>	Each catalogue may contain many nodes and each node may contain many child nodes.
<Illustration>	A presentation object used to describe illustrations.
<Document>	A presentation object used to describe documents.
<Image>	An object to describe an images.

<Bulletin>	An object defines bulletins
<Footnote>	An object defines the footnotes
<Brand>	An object defines the brands
<PartClassification>	An object defines the parts classifications
<Reference>	Reference is used to connect two objects via reference. One object may refer to many different reference identities.

4.2.1 Settings

It defines the settings in import context. It gives the information about import culture to avoid type conversion errors of date formats and decimal separator, xml format and version of format. It also gives information about directory which contains the files; images, documents and documents contents (images in html or xml documents).

```
<Settings>
   <Import Culture="en-US" format="Assert" version="1.0.0"  />
   <Images dir="Images" />
   <Documents dir="Documents" />
   <ContentDirectory src="Graphics"/>
</Settings>
```

4.2.2 Text

Text is used to import a translated text and to refer to a translated text. In the below example, *id* is used as a reference within the scope of the file. The *id* is not actually imported into Signifikant Platform.

```
<Texts>
     <Text id = "1">
          <Translation language="en-GB">English</Translation>
          <Translation language="sv-SE">engelska</Translation>
     </Text>
     <Text id = "2">
          <Translation language="en-GB">Swedish</Translation>
          <Translation language="sv-SE">svenska</Translation>
     </Text>
     <Text id = "3">
          <Translation language="en-GB">Bolt</Translation>
          <Translation language="sv-SE">Bult</Translation>
     </Text>
</Texts>
```

If language is omitted, the specified import language is used. If language is set to empty string, the text is considered language independent.

Other objects may refer to texts in their properties. E.g.

```
<Part id=”1” business-id=”123456”>
     <Name text-ref=”3”/>
```

```
</Part>
```

4.2.3 SpecificationType

SpecificationType is used to define custom properties on any object of type Presentation.

```
<SpecificationTypes>
    <SpecificationType id="1" persistent-id="SpecificationTypePId1"
code="sepciCode">
      <Category text-ref-id="16" />
      <Unit text-ref-id="17" />
      <Name text-ref-id="18" />
    </SpecificationType>
</SpecificationTypes>
```

Quite commonly the specification types will specify a type that need to be translated. In this case, specification type will refer to a text object, and the specification type is then used in e.g. a part.

```
<Texts>
     <Text id = "1">
          <Translation language="en-GB">Material</Translation>
          <Translation language="sv-SE">Material</Translation>
     </Text>
     <Text id = "2">
          <Translation language="en-GB">Length</Translation>
          <Translation language="sv-SE">Längd</Translation>
     </Text>
     <Text id = "3">
          <Translation language="en-GB">Bolt</Translation>
          <Translation language="sv-SE">Bult</Translation>
     </Text>
     <Text id = "4">
          <Translation language="en-GB">Copper</Translation>
          <Translation language="sv-SE">Koppar</Translation>
     </Text>
</Texts>

<Parts>
     <Part id=”1”>
      <Specification id="3" sequence="3">
        <Type Type-ref-id="1" />
        <Value>2050.00</Value>
      </Specification>
     </Part>
</Parts>
```

4.2.4 Specification

All presentation objects may have specifications. Where specification may have value of different types, it can be decimal, string or it can be text.

```
<Specification id="1" sequence="1">
    <Type Type-ref-id="1" />
    <Value>speci string</Value>
</Specification>
<Specification id="2" sequence="2">
    <Type Type-ref-id="1" />
    <Value>48747.98</Value>
</Specification>
```

```
<Specification id="3" sequence="3">
    <Type Type-ref-id="1" />
    <Value>2050.00</Value>
</Specification>
    <Specification id="4" sequence="4">
    <Type Type-ref-id="1" />
    <Value>
      <Text id="51">
         <Translation language="en-GB">specification Text in English</Translation>
         <Translation language="sv-SE">specification Text in Swedish</Translation>
      </Text>
   </Value>
</Specification>
```

4.2.5 Presentation

Presentation is a generic object with a set of properties. Document, Illustration, Catalogue, PartAssembly and Part inherit their properties from Presentation. The properties of Presentation are:

Property	Description	Example
Business-id	Business id, e.g. a part with number “101010”. Note that parts use the alias number for business-id.	`<part business-id=”102030” />`
Name	Presentation name	`<Name text-ref-id="22" />`
Description	Presentation description	`<Description text-ref-id="66" />`

Footnotes	List with additional information	`<Footnote id="1" sequence="1">`
		`<Name text-ref-id="29" />`
		`<Description text-ref-id="30" />`
		`<Note>footnote note</Note>`
		`</Footnote>`

Specifications	List with specifications	`<Specification id="3" sequence="3">`
		`<Type Type-ref-id="1" />`
		`<Value>2050.00</Value>`
		`</Specification>`

References	List with references	`<Reference id="2" sequence="1" PartAssembly-ref-id="8">`
		`<ReferenceIdentity id="2" sequence="5" catalogue-ref-id="7" />`
		`<Description text-ref-id="71" />`
		`</Reference>`

Bulletins	List with bulletins	`<Bulletin id="1" persistent-id="bulletinpid0" fromDate="1/1/2002 12:00:00 PM" toDate="1/1/2012 12:00:00 PM" priority="150">` `<Document document-ref-id="1" />`

```
                                    <Heading text-ref-id="31" />
                                    <Description text-ref-id="32" />
                                </Bulletin>
```

4.2.6 Part

The part object is used to import parts into Signifikant Platform. E.g.

```
<Parts>
    <Part id="5" sequence="0" persistent-id="partPid2" isSellable="True"
supplierNumber="552">
      <Specification id="1" sequence="1">
        <Type Type-ref-id="1" />
        <Value>speci string</Value>
      </Specification>
      <Specification id="2" sequence="2">
        <Type Type-ref-id="1" />
        <Value>48747.98</Value>
      </Specification>
      <Specification id="3" sequence="3">
        <Type Type-ref-id="1" />
        <Value>2050.00</Value>
      </Specification>
      <Specification id="4" sequence="4">
        <Type Type-ref-id="1" />
        <Value>
        </Value>
      </Specification>
      <Footnote id="1" sequence="1">
        <Name text-ref-id="29" />
        <Description text-ref-id="30" />
        <Note> </Note>
      </Footnote>
      <Name text-ref-id="24" />
      <Description text-ref-id="25" />
      <Note />
      <Brand brand-ref-id="2" />
      <PartClassification partclassification-ref-id="2" />
      <QuantityUnit text-ref-id="26" />
    </Part>
</Parts>
```

4.2.7 PartAssembly

The partassembly object is used to import part assemblies into Signifikant Platform. E.g.

```
<PartAssemblies>
    <PartAssembly id="8" isKit="True" sequence="5" persistent-
id="PartAssemblyPid">
      <Specification id="8" sequence="1">
        <Type Type-ref-id="1" />
        <Value>Part Assembely speci string</Value>
      </Specification>
      <Bulletin bulletin-ref-id="3" />
      <AssociatedPart part-ref-id="4" />
      <Name text-ref-id="45" />
      <Description text-ref-id="71" />
```

```
      <Note />
      <Row id="1" sequence="1" Quantity="3.0000000" showPartNumber="True">
        <Part part-ref-id="6" />
        <Note> This is a partMoudleRow Note </Note>
        <QuantityUnit text-ref-id="46" />
        <Remark text-ref-id="47" />
      </Row>
    </PartAssembly>
</PartAssemblies>
```

4.2.8 Catalogue

Catalogues are a set of nodes in a hierarchy, where each node may point to a content of object type Presentation. Each node may contain some properties.

```
<Catalogues>
  <Catalogue id="7" sequence="2">
    <Name text-ref-id="33" />
    <Description text-ref-id="34" />
    <Note> Cat1 Note ......</Note>
    <Image image-ref-id="9" />
      <Bulletin bulletin-ref-id="2" />
      <Footnote id="2" sequence="1">
        <Name text-ref-id="37" />
        <Description text-ref-id="38" />
        <Note>Cat1 Node1 footnote Note</Note>
      </Footnote>
      <Image image-ref-id="10" />
    <CatalogueNode id="1" sequence="2">
    <CatalogueNode id="2" sequence="2">
    <CatalogueNode id="3" sequence="2">
  </Catalogue>
</Catalogues>
```

4.2.9 CatalogueNode

A CatalogueNode is child element of catalogue where catalogue can have many CatalogueNodes and a CatalogueNode can also have many child CatalogueNodes.

```
<CatalogueNode id="1" sequence="2">
   <Bulletin bulletin-ref-id="2" />
   <Footnote id="2" sequence="1">
      <Name text-ref-id="37" />
      <Description text-ref-id="38" />
      </Note>
   </Footnote>
   <Image image-ref-id="10" />
   <CatalogueNode id="2" sequence="1" persistent-id="cateNode1child1Pid">
      <Image image-ref-id="11" />
      <Note>Child node note<Note>
      <Name text-ref-id="40" />
      <Description />
   </CatalogueNode>
   <Note> Parent node note<Note>
   <Name text-ref-id="35" />
   <Description />
</CatalogueNode>
```

4.2.10 Image

An image is an image file with some properties: language, size, etc. The file name always refer to an actual file, which is an attachment to the import.

```
<Images>
    <Image id="2" persistent-id="imgPId2" filename="test3.png" sequence="4"
width="12015" height="14055" category="my Cate" language="en-GB">
      <caption text-ref-id="20" />
   </Image>
</Images>
```

4.2.11 Illustration

Illustrations are a list of images and a hotspot layer. Only one image is shown to the user, and image is selected depending on language or resolution.

```
<Illustrations>
    <illustration id="2" sequence="0" persistent-id="Pid001">
      <Name text-ref-id="21" />
      <Description text-ref-id="67" />
      <Note />
      <Image image-ref-id="16" />
      <Image image-ref-id="17" />
    </illustration>
</Illustrations>
```

4.2.12 Document

A document is an object type Presentation. A document may contain several files for different translations. The file name always refer to an actual file, which is an attachment to the import.

```
<Documents>
    <document id="1" sequence="10" persistent-id="docuPid">
      <Name text-ref-id="22" />
      <Description text-ref-id="66" />
      <Note />
      <File filename="Assert Installer Log.1.txt" sequence="1" language="en-GB"
persistent-id="docuFilePid1" />
      <File filename="Assert Installer Log.2.txt" sequence="2" language="sv-SE"
persistent-id="docuFilePid2" />
    </document>
</Documents>
```

During import, a transformation may be applied to a document. This is useful when importing xml-documents that should be displayed as html. As a method of creating references to a document from a catalogue it is also possible to add information on the document's positon in the xml.

```
<document id="2" sequence="10" persistent-id="123" ApplyTransform="true"
TransformStyle="CMS-document"  AddToCatalogue="true" catalogue-identity="123"
catalogueNode-identity="456">
```

4.2.13 Bulletin

Bulletins is object type which can be defined in many other objects such as all objects which are presentation and also CatalogueNodes.

```
<Bulletins>
```

```
    <Bulletin id="1" persistent-id="bulletinpid0" fromDate="1/1/2002 12:00:00 PM"
toDate="1/1/2012 12:00:00 PM" priority="150">
      <Document document-ref-id="1" />
      <Heading text-ref-id="31" />
      <Description text-ref-id="32" />
    </Bulletin>
</Bulletins>
```

4.2.14 Footnote

Footnotes is object type which can be defined in many other objects such as all objects which are presentation and also CatalogueNodes, PartAssembly Row.

```
<Footnote id="1" sequence="1">
    <Name text-ref-id="29" />
    <Description text-ref-id="30" />
    <Note>Note</Note>
</Footnote>
```

4.2.15 Brand

Parts can have different brands. Brand represents the design, or other features of the product.

```
<Brand id="2" persistent-id="Pid001" name="Name of the Brand">
   <Image image-ref-id="8" />
   <Note> Note ......</Note>
</Brand>
```

4.2.16 PartClassification

Partclassification defines the classification of particular part.

```
<PartClassification id="1" code="code1">
      <Name text-ref-id="19" />
</PartClassification>
```

It can be used in part section as follow.

```
<Part id="4" sequence="0" persistent-id="partpid1" supplierNumber="551">
  <Name text-ref-id="23" />
  <PartClassification partclassification-ref-id="1" />
</Part>
```

4.2.17 Reference

Reference is used to connect two objects via reference. One object may refer to many different reference identities. Objects can be referred using Id, Persistent-id or Technical-id.

```
<References>
   <Reference id="1" sequence="1" catalogueNode-ref-id="3">
      <ReferenceIdentity id="1" sequence="5" part-ref-id="4" />
      <Description text-ref-id="70" />
   </Reference>
   <Reference id="2" sequence="1" PartAssembly-ref-id="8">
      <ReferenceIdentity id="2" sequence="5" catalogue-ref-id="7" />
      <Description text-ref-id="71" />
   </Reference>
</References>
```

4.3 Attachments

The objects image and document always refer to an external file. Image and document files are stored in sub catalogues; one for images and several for the various languages of the documents.

Catalogues for attached files	
/Images	For images
/Documents	Root-catalogue for documents
/en-GB	English documents
/sv-SE	Swedish documents

4.4 Reference strings

All objects have specific reference code (string) which are changeable, case insensitive attributes, used to refer an object. Using reference string will reduce the xml. We can refer an object several times so that we do not need to write xml for whole object again.

Server:

```
Id = "Server-ref-id";
PersistentId = "Server-ref-persistent-id";
TechnicalId = "Server-ref-Technical-id";
```

Site:

```
Id = "Site-ref-id";
PersistentId = "Site-ref-persistent-id";
TechnicalId = "Site-ref-Technical-id";
```

HotSpot:

```
Id = "HotSpot-ref-id";
PersistentId = "HotSpot-ref-persistent-id";
TechnicalId = "HotSpot-ref-Technical-id";
```

Text:

```
Id = "text-ref-id";
PersistentId = "text-ref-persistent-id";
echnicalId = "Text-ref-Technical-id";
```

Part:

```
Id = "part-ref-id";
PersistentId = "part-ref-persistent-id";
TechnicalId = "Part-ref-Technical-id";
```

Brand:

```
Id = "brand-ref-id";
PersistentId = "brand-ref-persistent-id";
TechnicalId = "Brand-ref-Technical-id";
```

SpecificationType:

```
Id = "Type-ref-id";
PersistentId = "Type-ref-persistent-id";
TechnicalId = "SpecificationType-ref-Technical-id";
```

Image:

```
Id = "image-ref-id";
PersistentId = "image-ref-persistent-id";
TechnicalId = "Image-ref-Technical-id";
```

PartClassification:

```
Id = "partclassification-ref-id";
PersistentId = "partclassification-ref-persistent-id";
TechnicalId = "PartClassification-ref-Technical-id";
```

Catalogue:

```
Id = "catalogue-ref-id";
PersistentId = "catalogue-ref-persistent-id";
TechnicalId = "Catalogue-ref-Technical-id";
```

CatalogueNode:

```
Id = "catalogueNode-ref-id";
PersistentId = "catalogueNode-ref-persistent-id";
TechnicalId = "CatalogueNode-ref-Technical-id";
```

Footnote:

```
Id = "footnote-ref-id";
PersistentId = "footnote-ref-persistent-id";
TechnicalId = "Footnote-ref-Technical-id";
```

Document:

```
Id = "document-ref-id";
PersistentId = "document-ref-persistent-id";
TechnicalId = "Document-ref-Technical-id";
```

DocumentFile:

```
Id = "documentfile-ref-id";
PersistentId = "documentfile-ref-persistent-id";
TechnicalId = "DocumentFile-ref-Technical-id";
```

Bulletin:

```
Id = "bulletin-ref-id";
PersistentId = "bulletin-ref-persistent-id";
TechnicalId = "Bulletin-ref-Technical-id";
```

Illustration:

```
Id = "illustration-ref-id";
PersistentId = "illustration-ref-persistent-id";
TechnicalId = "Illustration-ref-Technical-id";
```

PartModule:

```
Id = "PartAssembly-ref-id";
PersistentId = "PartAssembly-ref-persistent-id";
TechnicalId = "PartModule-ref-Technical-id";
```

PartModuleRow:

```
Id = "PartAssemblyRow-ref-id";
PersistentId = "PartAssemblyRow-ref-persistent-id";
TechnicalId = "PartAssemblyRow-ref-Technical-id";
```