IDoc Import Rules

IDocs (**I**nterface **Doc**uments) are the most common way of interchanging data asynchronously with SAP R/3. IDocs consists of control and data records. The control record contains data like IDoc type, sender, receiver, IDoc number and many others. Its purpose is to control the routing and handling of the IDoc. The data records are called segments. Each segment has an 63 byte long header and a 1000 byte data part.

The most simple IDoc is TXTRAW01 containing as data part just a string. The IDoc header contains information like segment type, segment number, parent number, and so forth. The parent number is used to link the segment to other segments building a hierarchical structure. The data part consists of values put in position dependent slots. The information which data attribute is found on which position can be found in the IDoc parse grammar (**WE60** - choose IDoc type - click on **Parser (F9)**) on your SAP system. More information about IDocs can be found in the SAP Library help.
The result of importing TXTRAW01 together with the description of the SAP parser grammar to UML mapping is shown in the class diagram TXTRAW (see figure "Example Of an Imported TXTRAW IDoc").

The modeler must execute the following steps in order to get UML classes of a SAP IDoc:

1. Generate the parse grammar.
2. Generate a WSDL file from the generated SAP IDoc parse grammar.
3. Import the generated WSDL into the UML model.
   You can find more information of how to execute this step on Importing SAP IDoc Meta Data the E2E Builder User's Guide.
4. Model and compile the IDoc transformations.
   You can find more information of how to execute this step on IDoc Handling.

Structure of the Imported Classes

The result of importing SAP IDoc parse grammars will be a set of classes being of stereotype <<SAPIDoc>>. Such classes must be derived from the EDI_DC40 class which holds all attributes found in the control record because the control record is common to all IDocs (see figure "Example Of an Imported TXTRAW IDoc").
Each segment is modeled as class of stereotype <<SAPIDocSegment>> that has an association to the IDoc class or to other segments. The association end name and multiplicity is given by the segment name respectivly the LOOPMIN/LOOPMAX numbers in the SAP parse grammar.

An example of a SAP IDoc parse grammar is:

```
BEGIN_SEGMENT_SECTION
   BEGIN_IDOC          TXTRAW01
     BEGIN_SEGMENT          E2TXTRW
       SEGMENTTYPE         E1TXTRW
       LEVEL          02
       STATUS          MANDATORY
       LOOPMIN          0000000001
       LOOPMAX          0000003600
       BEGIN_FIELDS
         NAME          TLINE
         TEXT          Text line for SAPmail via IDoc interface
         TYPE          CHARACTER
         LENGTH          000072
         FIELD_POS          0001
         BYTE_FIRST          000064
         BYTE_LAST          000135
       END_FIELDS
     END_SEGMENT
   END_IDOC
END_SEGMENT_SECTION
```

> If segment `STATUS` is `MANDATORY` and if `LOOPMIN` is 0, the lower multiplicity is set to 1 to be consistent.

SAP IDoc segment classes have the following tagged values:

Tagged Value	Description

hierarchy Level	corresponds to `LEVEL` in the SAP parse grammar.
type	corresponds to `SEGMENTTYPE` in the SAP parse grammar
group	Segments containing child segments (see class diagram of `CREMAS01`, figure "Fragmentary View of CREMAS01 IDoc Class Structure") are called group-segments. These segments and its children are nested within a `BEGIN_GROUP` section in the SAP parser grammer. The `BEGIN_GROUP` tag is associated with a number which is put into the group tagged value for each group-segment.

Segment attributes can have the following tagged values:

Tagged Value		Description
order		gives the order of the attribute within the IDoc; corresponds to FIELD_POS in the SAP parse grammar
offset		index position in bytes starting at the beginning of the segment; corresponds to BYTE_FIRST in the SAP parse grammar
externalLength		reserved length of the field in the IDoc segment: BYTE_LAST - BYTE_FIRST
internalLength		length in the SAP system: corresponds to LENGTH in the parse grammar
nativeType		basic data type of the segment attribute. The supported types are listed beneath
decimals		number of decimals. Relevant only for numeric types such as DEC

Figure: Example Of an Imported TXTRAW IDoc

«XML»
«SAPStructure»
«FlatFileRecord»
EDI_DC40
{attributeLayout = fixed,
evaluationOrder = 1,
lineNumber = 1}

«XML»
«SAPIDoc»
TXTRAW01
{isOrdered}

E2TXTRW ▼

«E2EAttribute» {externalName = "E1TXTRW",
«XMLElement»   order = 1.0}
+E2TXTRW 1..*

«XML»
«SAPIDocSegment»
E2TXTRW
{hierarchyLevel = "2",
isOrdered,
type = "E1TXTRW"}
+SEGMENT : String = 1
+TLINE : String
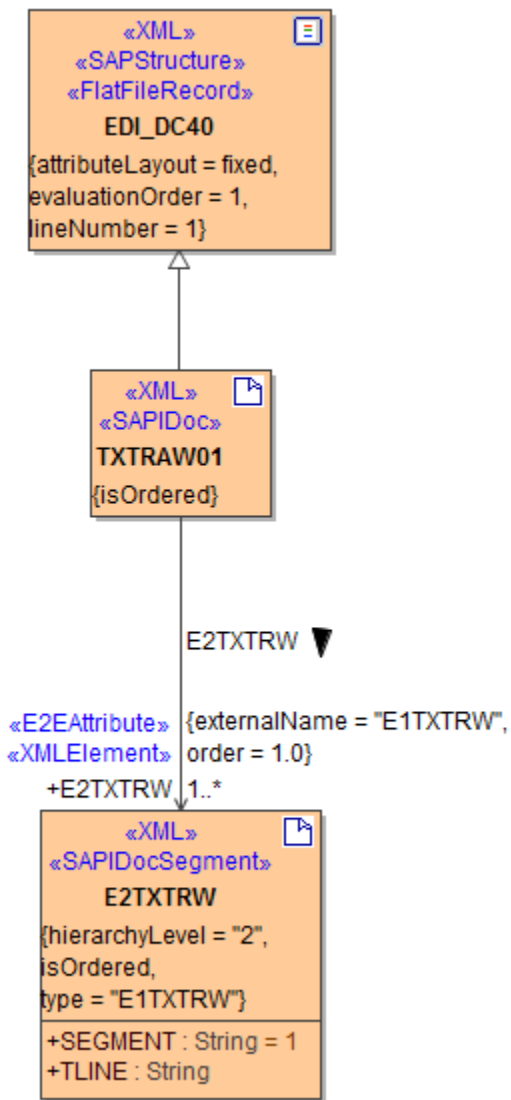
Generating and Importing Meta-information

The base grammar does not contain simple type information – every data element is of type String. Therefore, if it is required to use other base types (such as e.g. numbers, because you want to do some calculations), it is necessary to import additional meta-information of the simple types of the data segment attributes. Without this information it is not possible to convert IDoc segment attribute types automatically to internal types.

This information is typically found in the HTML descriptions of IDocs. The HTML documentation can also be generated within transaction **WE60**. The generated HTML documentation is then used as input of the IDoc Importer, which maps the segment attribute types to internal types.

The mapping is as follows:

Input Type	Output Type
CHAR, LANG, CUKY, STRG, UNIT, ACCP, NUMC	String
QUAN, CURR, DEC, FLTP	Float
RSTR, RAW, PREC, INT1, INT2	Blob
INT4	Integer
TIMS, DATS	DateTime

Figure: Fragmentary View of CREMAS01 IDoc Class Structure

«XMPSG»
CREMAS01
{xmlNamespace:xmlcom:CREMAS01:version6,
isOrdered,
groupsY}

[content 1]
1
+E2LFAHM001

«XMPSGrouphead»
E2LFAHM001
{xmlNamespace:xmlcom:CREMAS01:version6,
isOrdered,
hierarchyLevel=1,
type=E1LFA1M,
groupY}

+FKVRT : String
+BEMV3 : String
+LTSNK : String
+KBVR3F : String
+PLVAL : String
+DUEFL : String
+TLKD : String
+ICHCD : String
+XPISR : String
+LTSNE : String
+QUIRP : String
+KTDIF : String
+BTOIT : String
+REDIYI : String
+ALTDI : String
+KTCDS : String
+KTCDH : String
+IPISP : String
+TNISY : String
+XPISPX : String
+HTSDL : String
+IDMXR : String
+HADSL : String

[content 00]   [content01]   [content02]   [content08]   [content11]
0.3089   0.3089   0.3089   0.3089   0.3089
+E2LFBHM02   +E2HYTHM   +E2LFBHM003   +E2LFBHM003   +E2LFAM

«XMPSGrouphead»   «XMPSGrouphead»   «XMPSGrouphead»   «XMPSGrouphead»   «XMPSGrouphead»
E2LFBHM02   E2HYT1M   E2LFBHM03   E2LFBHM003   E2LFAM
{xmlNamespace:xmlcom {xmlNamespace:xml {xmlNamespace:xml {xmlNamespace:xml {xmlNamespace:xml
:CREMAS01:version6,   com:CREMAS01:ver   com:CREMAS01:ver   com:CREMAS01:ver   com:CREMAS01:ver
isOrdered,   sion6,isOrdered,   sion6,isOrdered,   sion6,isOrdered,   sion6,isOrdered,
hierarchyLevel=2,   hierarchyLevel=2, hierarchyLevel=2, hierarchyLevel=2, hierarchyLevel=2,
type=E1LFBHM}   type=E1HYT1M,   type=E1LFB1M,   type=E1LFB1M,   type=E1LFASM}
                     groupsN}   groupY}   groupN}