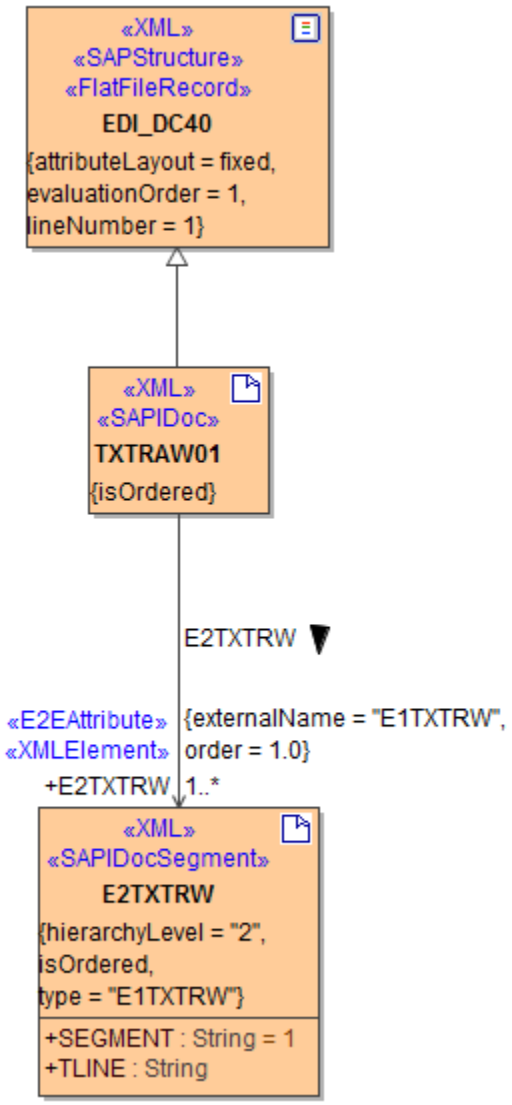
# Parsing IDocs

There are two situations in which you may want to parse an IDoc data buffer:

- You want to map a **String** object containing IDoc data to an UML class, that has the same structure as the data embodied in the IDoc (e.g. **TXTRAW01**, see figure below).
  This is done by an action having the stereotype <<SAPIDocParser>> (see Parsing from String).
- You want to map a SAP **tables** object containing IDoc data (e.g. coming from an RFC call) to an IDoc UML class, that has the same structure as the data embodied in the SAP tables (e.g. **TXTRAW01**, see figure below).
  This is done by an action having the stereotype <<SAPIDocRecordParser>> (see Parsing from SAP Tables).

Both parsers generate an output class of type **Array** having an IDoc UML class as its array element type, e.g. **TXTRAW01**.

*Figure: UML class model of the IDoc TXTRAW01*




After having called the parser, you can catch parsing errors as described in Catching Errors.

**On this Page:**



**Related Pages:**


- Composing IDocs
- Reading IDocs From File
- SAP Adapter Reference

**Example File (Builder project Add-ons/SAP):**

| | |
|---|---|
| | <your example path>\Add-ons\SAP\uml\sapIdoc.xml |

# Parsing from String: <<SAPIDocParser>>

The <<SAPIDocParser> takes an **IDocString** as input and assumes that it contains valid control and data records. This string is parsed to a class that must correspond to the contained IDoc.

The <<SAPIDocParser>> can be used with the following parameters:

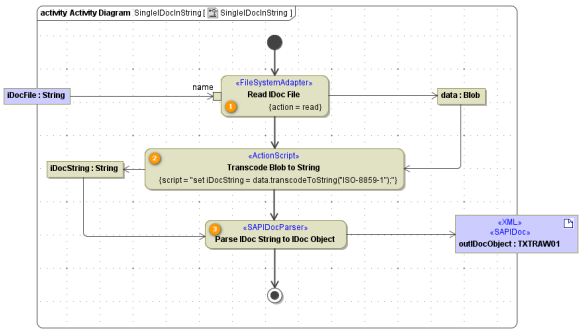
| Name | Type | Direction | Description |
|---|---|---|---|
| **idocString** | **String** | in | String containing IDoc data (e.g. the content of an IDoc file) |
| anyObjectFlow | **Any** | out | Parsed IDoc object(s)<br>The class specifying the type of this parameter must have stereotype <<SAPIDoc>>. |

The <<SAPIDocParser>> returns an **Any** object having the stereotype <<SAPIDoc>>.
The following examples are demonstrating some applications of the <<SAPIDocParser>>.

## Parsing a Single IDoc From String to a Corresponding Class

If you are sure, that the parser input contains only one IDoc, you can get the output directly into an output object of the corresponding IDoc type (e.g. **TXTRAW01**).

*Figure: SAPIDocParser, Single IDoc From String*




Explanation of the activity diagram:

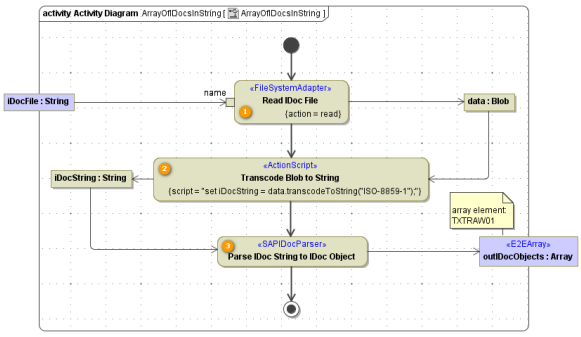
1. Reading a (valid) IDoc from file into a **Blob**.
2. Transcoding the IDoc blob to a **String**.
3. Parsing the IDoc string to an object that is corresponding to the type of the IDoc.

After having called the parser, you can catch parsing errors as described in Catching Errors.

## Parsing Multiple IDocs From String to an Array of Corresponding Class

If the parser input contains multiple IDocs (of the same type), you should get the output in to an array of objects of the corresponding IDoc type (e.g. **TXTRAW01**).

*Figure: SAPIDocParser, IDoc From SAP Tables*




Explanation of the activity diagram:

1. Reading a list of (valid) IDocs from file into a **Blob**.
2. Transcoding the list of IDocs blob to a **String**.
3. Parsing the IDocs string to an array of objects that are corresponding to the type of the IDocs.

After having called the parser, you can catch parsing errors as described in Catching Errors.

# Parsing from SAP Tables: <<SAPIDocRecordParser>>

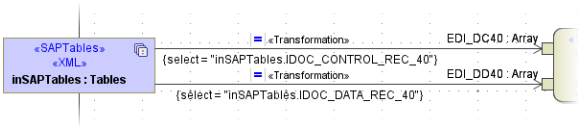
The <<SAPIDocRecordParser> takes two structured arrays as input and assumes that they contains valid control resp. data records. These arrays are parsed to a class that must correspond to the contained IDoc.

The <<SAPIDocRecordParser>> can be used with the following parameters:

| Name | Type | Direction | Description | |
|---|---|---|---|---|
| **EDI_DC40** | **Array** | in | Array of IDoc control record strings | |
| **EDI_DD40** | **Array** | in | Array of IDoc data records strings | |
| anyObjectFlow | **Any** | out | Parsed IDoc object(s)<br>The class specifying the type of this parameter must have stereotype <<SAPIDoc>>. | |

You can get the input arrays from given SAP tables, an object of type **Tables**, e.g. using a transformation as depicted in the following example:

*Figure: Getting IDoc Control and Data Records from Given SAP Tables*

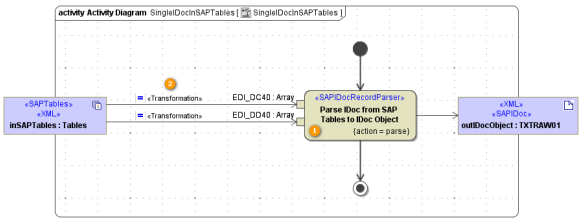



The <<SAPIDocRecordParser>> returns an **Any** object having the stereotype <<SAPIDoc>>.
The following examples are demonstrating some applications of the <<SAPIDocRecordParser>>.

## Parsing a Single IDoc From SAP Tables to a Corresponding Class

If you are sure, that the parser input contains only one IDoc, you can get the output directly into an output object of the corresponding IDoc type (e.g. **TXTRAW01**).

*Figure: SAPIDocRecordParser, Single IDoc From Tables*



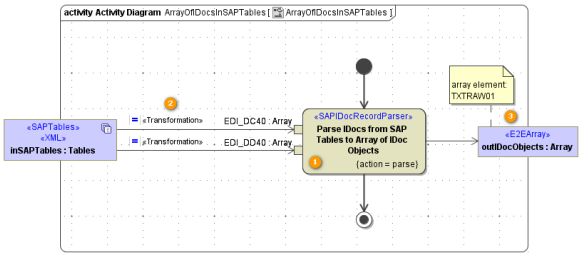

Explanation of the activity diagram:

1. Parsing a single IDoc from SAP tables to an object that is corresponding to the type of the IDoc.
2. The necessary input arrays are coming directly from the SAP tables via a transformation.

## Parsing Multiple IDocs From SAP Tables to a Corresponding Class

If the parser input contains multiple IDocs (of the same type), you should get the output in to an array of objects of the corresponding IDoc type (e.g. **TXTRAW01**).

*Figure: SAPIDocRecordParser, Multiple IDocs From Tables*



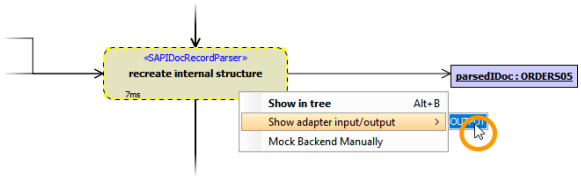

Explanation of the activity diagram:

1. Parsing multiple IDocs from SAP tables to an array of objects that are corresponding to the type of the IDocs.
2. The necessary input arrays are coming directly from the SAP tables via a transformation.
3. The parser output is an array  of IDoc objects having the IDoc type as array element.

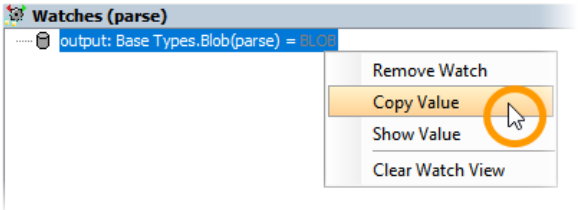
## Inspecting the Parsing Process With the Analyzer

You can inspect the parsing process of the SAP IDoc Record Parser with the Analyzer. To view the parser trace file do the following:

1. Run a testcase with **Full Trace** option.
2. Navigate to the IDoc parser action.
3. Select **Show adapter input/output > OUTPUT** from the context menu. The adapter output will be displayed in the **Watches** section of the Analyzer.




4. Right-click the displayed adapter output **output** and select **Copy Value** from the context menu.




5. Paste the copied content into a text editor of your choice. It displays something like:

```
Start parsing message: recreate_internal_structure...
_965161528|_17_0_3_b2002db_1406276778049_987802_26054.
Start adding IDoc.
Adding IDOC number: 14468604.
Start adjusting CREDAT Type: target type = String.
CREDAT: DateTime converted to String: 20140526.
```

# Parsing from XML: <<SAPXMLIDocParser>>

The <<SAPXMLIDocParser> takes a **Blob** as input and assumes that it contains valid control and data records. This string is parsed to a class that must correspond to the contained IDoc.

The <<SAPXMLIDocParser>> can be used with the following parameters:

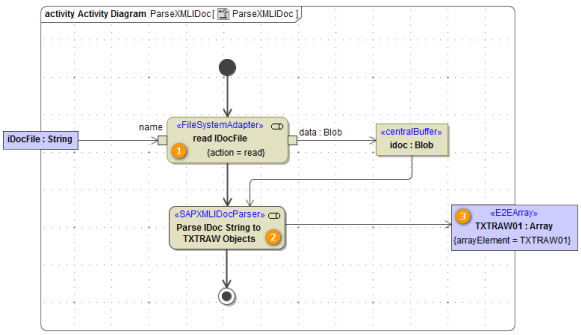
| Name | Type | Direction | Description |
|---|---|---|---|
| **idoc** | **Blob** | in | Blob containing IDoc data (e.g. the content of an IDoc file) |
| anyObjectFlow | **Any** | out | Parsed IDoc object(s)<br>The class specifying the type of this parameter must have stereotype <<SAPIDoc>> and <<XML>>.<br><br>The receiving object must be named exactly after the root element of the IDoc, e.g. TXTRAW01. If not, the model will throw an exception. |

The <<SAPXMLIDocParser>> returns an **Any** object having stereotype <<SAPIDoc>>.
The following example is demonstrating an application of the <<SAPXMLIDocParser>>.

## Parsing Multiple IDocs From XML to a Corresponding Class

If you are sure, that the parser input contains only one IDoc, you can get the output directly into an output object of the corresponding IDoc type (e.g. **TXTRAW01**).
If the parser input contains multiple IDocs (of the same type), you should get the output in to an array of objects of the corresponding IDoc type.

*Figure: SAPXMLIDocParser, Multiple IDocs From XML*




Explanation of the activity diagram:

1. Reading a list of (valid) IDocs from file into a **Blob**.
2. Parsing the IDocs blob to an array of objects that are corresponding to the type of the IDocs.
3. The receiving object must be named exactly after the root element of the IDoc, e.g. TXTRAW01. If not, the model will throw an exception at runtime.

# IDoc Parsing and SAP R/3

The above examples show how to parse IDocs of version 4.x . However, SAP R/3 3.x systems use IDocs, too. These IDocs are handled similar as in version 4.x. The differences between parsing version 4 and version 3 IDocs are as follows :

- IDoc3 Parser Input: EDI_DD and EDI_DC, both string arrays.

- IDoc3 Stereotype: <<SAPIDoc3>>, this is taken into account if the import was done in version 3 mode (see Importing SAP IDoc Meta Data).