

Parsing Flat Files

After [having defined the structure of the flat file](#), you can use the Flat File adapter to parse a flat file into this structure.

Drag the **parse** operation to your diagram as described on [Flat File Adapter](#). Provide the flat file as a blob in parameter **data** or specify a path to a file in the filesystem, and provide an encoding or localization if necessary.

The path and file name of the file you want to access can be given dynamically via input parameter **name**, or statically via the definitions of a File **alias**.

The parsed file is returned as an object of the defined flat file structure.

Name	Type	Direction	Mandatory	Description	Allowed Values	Example
data	Blob	in	✔	Provide the flat file data to be parsed. Alternatively, you can specify a path to a flat file in the file system (see parameter name). Note, that the name parameter takes priority over data .		
encoding	String	in		Provide the encoding of the file to be parsed as specified on Charset Definitions .	any valid encoding (see Charset Definitions) def a ultISO-8859-1 (Latin1)	UTF-8
locale	Numbers Locale	in		Specify how number values will be treated, when parsed from the flat file (decimal point, currency symbol, ...). You can overwrite the system locales here, if the file was written with divergent locales. Refer to Number Formatting for more information.		
name	String	in	✔	Specify a full path to the flat file to be parsed. Alternatively, you can parse the flat file from a Blob object (see parameter data). Note, that the name parameter takes priority over data .		tmp /myFile.txt
anyObject Flow	Any with FlatFile class stereotype	out	✔	The adapter returns a parsed flat file object. The class defining the type of this object should have stereotype FlatFile and should depict the structure of the file.		



If you provide both parameters, **name** and **data**, the Flat File object will be parsed from the file system.

On this Page:

- [Flat File Adapter Parsing Process](#)
- [Inspecting the Parsing Process With the Scheer PAS Analyzer](#)

FlatFileAdapter_ProductExport_Example



Click the icon to download a simple example model that shows the usage of the Flat File adapter in **Scheer PAS Designer**.

Related Pages:

- [Defining a Flat File Data Structure](#)
- [Using Macro Expressions on Parsing or Composing a Flat File](#)
- [Flat File Adapter Reference](#)

Flat File Adapter Parsing Process

The Flat File adapter parse action processes the parsing of a flat file using the following steps:

Step	Description	Example
open	Open file or blob.	
file	Create FlatFile object.	OrderFile
fetch	Fetch first record.	
check	Go through all associations (FlatFileRecord and FlatFileGroup classes) until the first one matches depending on the tagged values evaluationOrder , lineNuber , condition , and pattern .	Order
out	If no association matches, step out of the recursion and go to step check , abort with an error if there is no parent.	
group	If a group matches, create a FlatFileGroup object and go to step check.	OrderLine
record	If a record matches, create a FlatFileRecord object.	OrderLine
attribute	If the FlatFileRecord class has attributes, process attributes using current record data.	OrderID, customer, ...
next	Fetch next record.	
[...]		
close	Close file and end Flat File adapter.	

The following example shows the parsing process for a given class diagram. The names in the figure refer to the actions in the table above.

Then, it shows the pattern to be matched and the record that matches the pattern - if there is one.	<pre> "ORDER 1234 Winter & Partners 20230531 20230601" Found record "orders" (class="urn: Data_Model.FlatFile.Order"): pattern "^ORDER.*" does match </pre>
If the records are nested hierarchically, the trace log will show the nesting as well by indenting the nested records.	<pre> "PRODQNTY 1 AF- 1300 3 0000067.5 USD" Found record "lines" (class="urn: Data_Model.FlatFile.OrderLine"): pattern "^PRODQNTY.*" does match "LOCNQNTY 204 10" Found record "stockInfo" (class="urn: Data_Model.FlatFile.Stock"): pattern "^LOCNQNTY.*" does match "PRODQNTY 2 RC- 0003 1 0000075. USD" Skipped record "stockInfo" (class=" urn:Data_Model.FlatFile.Stock"): pattern "^LOCNQNTY.*" does not match Found record "lines" (class="urn: Data_Model.FlatFile.OrderLine"): pattern "^PRODQNTY.*" does match </pre>



The parsing log file is only available if trace mode is activated. To have the complete log, additionally activate **Full Trace**.
On composing a file, no logs are available.