Mapping of Hierarchical Record Structures

This page explains the **Flat File Adapter** in Bridge context. If you were looking for the same information regarding the PAS Designer, refer to Flat File Adapter in the Designer guide.

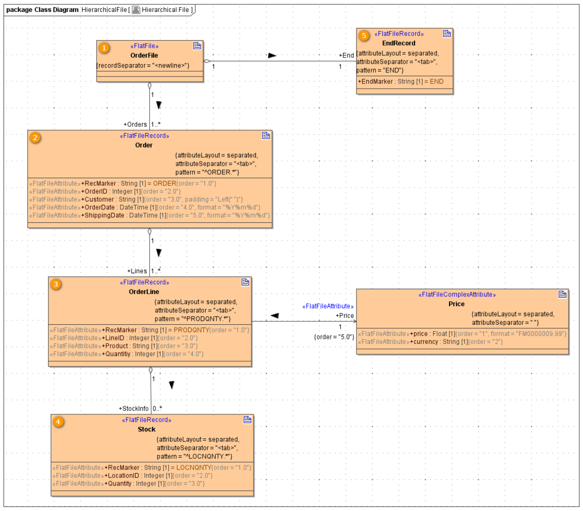
Even flat files can have a hierarchical record structure, means that the file contains different record types. This structure can be directly expressed in class diagrams by associations between <<FlatFileRecord>> classes.

Example File (Builder project Add-ons/FlatFiles):

<your example path>\Add-ons\FlatFile\uml\flatFileHierarchic.xml

See, for example, an order file class structure for a file containing order, order line and stock data:

Figure: Flat File Hierarchical Record Structures


package Class Diagram HierarchicalFile [Hierarchical File]
1
«FlatFile»
OrderFile
{recordSeparator = "<newline>"}
1
+End
1
5
«FlatFileRecord»
EndRecord
{attributeLayout = separated,
attributeSeparator = "<tab>",
pattern = "END"}
+EndMarker : String [1] = END
1
+Orders 1..*
2
«FlatFileRecord»
Order
{attributeLayout = separated,
attributeSeparator = "<tab>",
pattern = "^ORDER.*"}
«FlatFileAttribute»+RecMarker : String [1] = ORDER{order = "1.0"}
«FlatFileAttribute»+OrderID : Integer [1]{order = "2.0"}
«FlatFileAttribute»+Customer : String [1]{order = "3.0", padding = "Left(" ")"}
«FlatFileAttribute»+OrderDate : DateTime [1]{order = "4.0", format = "%Y%m%d"}
«FlatFileAttribute»+ShippingDate : DateTime [1]{order = "5.0", format = "%Y%m%d"}
1
+Lines 1..*
3
«FlatFileRecord»
OrderLine
{attributeLayout = separated,
attributeSeparator = "<tab>",
pattern = "^PRODQNTY.*"}
«FlatFileAttribute»+RecMarker : String [1] = PRODQNTY{order = "1.0"}
«FlatFileAttribute»+LineID : Integer [1]{order = "2.0"}
«FlatFileAttribute»+Product : String [1]{order = "3.0"}
«FlatFileAttribute»+Quantitiy : Integer [1]{order = "4.0"}
«FlatFileAttribute»
+Price
1
{order = "5.0"}
«FlatFileComplexAttribute»
Price
{attributeLayout = separated,
attributeSeparator = " "}
«FlatFileAttribute»+price : Float [1]{order = "1", format = "FM0000009.99"}
«FlatFileAttribute»+currency : String [1]{order = "2"}
1
+StockInfo 0..*
4
«FlatFileRecord»
Stock
{attributeLayout = separated,
attributeSeparator = "<tab>",
pattern = "^LOCNQNTY.*"}
«FlatFileAttribute»+RecMarker : String [1] = LOCNQNTY{order = "1.0"}
«FlatFileAttribute»+LocationID : Integer [1]{order = "2.0"}
«FlatFileAttribute»+Quantity : Integer [1]{order = "3.0"}

The flat file parser identifies each <<FlatFileRecord>> class by a regular expression found in the tagged value **pattern** . This pattern applies to the complete record (data line of the file).
If the classes are placed in a hierarchical order, the flat file records must follow this order to be parsed correctly. If they do not, the parser stops at the first record not matching the expected pattern.

Flat File Record	Pattern	Description
2 **Order**	`^ORDER.*`	All record lines starting with `ORDER` are **Order** records.
3 **OrderLine**	`^PRODQNTY.`	All record lines starting with `PRODQNTY` are **OrderLine** records.
4 **Stock**	`^LOCNQNTY.*`	All record lines starting with `LOCNQNTY` are **Stock** records.
5 **EndRecord**	`END`	All records containing `END` are **EndRecords**.

An example file for this structure could be the following:

```
ORDER    1234567      MYER    19990823    19990825
19990903                4631
PRODQNTY    1    PRD004    6    25.52    EA    C
I
LOCNQNTY    204    6
PRODQNTY    2    PRD001    9    25.52    EA    C
I
LOCNQNTY    202    4
LOCNQNTY    204    3
LOCNQNTY    211    2
PRODQNTY    3    PRD002    10    127.99    EA    C
I
LOCNQNTY    202    2
LOCNQNTY    204    1
LOCNQNTY    207    3
LOCNQNTY    211    4
END
```