How psml data is stored in memory

All psml data is stored in memory inside an indexed array List which is essentially a table with the following 4 columns:

`Node Name`	`Node Type`	`Value`	`Type`
The name of the record as a case sensitive variable length string. - For node type **header**, this will be the name of the element/node. - For **attribute**, this will be the name of the attribute. - For **element**, this will be the name of the nested element/node. - For **free**, this will be a null string. Examples: `“psml”`, `“Name”`, `“”`.	The type of data of the referenced record. Stored as one of the following: - `“H”`: A **header**, The start of a node. - `“A”`: An **attribute**, An stored value saved as an XML attribute. - `“E”`: An **element**, A declaration of a nested XML element. - `“”` **Free**, an empty data cell.	An actual or offset value associated with the record. - For node type **header**, this is the number its attributes plus the number of referenced elements (including the header itself). The sum is the overall size of the element within the array. - For **element** this will be the offset to the header of the current element. This is calculated by subtracting the row index of the record from the row index of its header record. - For **attribute** this will be the stored value of the attribute itself. - For **free** this will be a null string (`“”`). Examples: `“-2”`, `“Alex”`, `“1234”`	The data type of the value or the offset value to find the parent of a header. - For node type **header** this will the offset value to the *parent node header*. E.g `“-5”`. This is calculated by subtracting the row index of the record from the row index of its parent's header. - For **attribute** this will be a specified data type: long, string, boolean or float - For **free** or **element** this will be an empty string. *However*, free elements which are the start of a free Drop will have a reference to the data structure holding Drop addresses. Users have no visibility of this information.

- The complete data structure holding the entire psml file in memory is called a Jellyfish **Ocean**.
- Each node is stored within the Ocean as units called **Drops**.
- Each Drop is made up of rows of information (each holding the Name, Type, Value and Data columns described above). These are called **Molecules**.
- All Drops have a Header Molecule and at least one Element or Attribute Molecule.

Example 1

```
<A>
     <B att=”Alex” />
</A>
```

Index	Node Name	Node Type	Value	Type	Notes	Element being stored
1	“A”	“H”	“2”	“0”	**Value(“2”)** = header(1) + attribute count(0) + nested element count(1) **Type(“0”)** = index of parent header (itself in this case as it has no parent) [1] – current index [1]	A
2	“B”	“E”	“1”	“”	**Value(“1”)** = B header index[3] – current index[2]	A
3	“B”	“H”	“2”	“-2”	**Value(“2”)** = header(1) + attribute count(1) + nested element count(0) **Type(“-2”)** = index of parent header(A)[1] - current index[3]	B
4	“att”	“A”	“Alex”	“String”		B
5	“”	“”	“”	“”		None (free)

Example 2

```
<psml version=”1.23”>
     <Jellyfish1 Name=”123F” />
     <Jellyfish2 Name=”456G” />
</psml>
```

Index	Node Name	Node Type	Value	Type	Notes	Element being stored
1	“psml”	“H”	“4”	“0”	**Value(“4”)** = header(1) + attribute count(1) + nested element count (2) **Type(“0”)** = index of parent header (itself in this case as it has no parent) [1] – current index [1]	psml

2	“version”	“A”	“1.23”	“String”		psml
3	“Jellyfish 1”	“E”	“2”	“”	**Value(“2”)** = Jellyfish1 header (H) index [5] – current index [3]	psml
4	“Jellyfish 2”	“E”	“3”	“”	**Value(“3”)** = Jellyfish2 header (H) index [7] – current index [4]	psml
5	“Jellyfish 1”	“H”	“2”	“-4”	**Value(“2”)** = header(1) + attribute count(1) **Type(“-4”)** = index of parent header (psml) [1] – current index [5]	Jellyfish1
6	“Name”	“A”	“123F”	“String”		Jellyfish1
7	“Jellyfish 2”	“H”	“2”	“-6”	**Value(“2”)** = header(1) + attribute count(1) + nested element count(0) **Type(“-6”)** = index of parent (psml) header [1] - current index [7]	Jellyfish2
8	“Name”	“A”	“456G”	“String”		Jellyfish2
9	“”	“”	“”	“”		None (Free)

Initialise()

Creates a new session handle to lookup Jellyfish attribute and element values from a Jellyfish Ocean in memory. A unique session number is assigned to the referenced iSessionIndex parameter.

Syntax

```
public void Initialise (
      ref int iSessionIndex,
      int iOceanIndex,
      ref int iFlags
);
```

Example Usage

```
int iThisSession;
int iOceanIndex = 2;
int iFlags = 3;

API.Initialise (ref iThisSession, iOceanIndex, ref iFlags);
```

Output

`iThisSession` is set to a unique session number for use in navigating the specified Jellyfish Ocean.

`iFlags` is modified to signify operation success or failure:

- If iFlags = +ve int : SUCCESS
- If iFlags = -ve int : OPERATION FAIL
- If FAIL, the amended iFlags corresponds to an error code (see: Error Codes).

Parameters

Type	Description
`ref int iSessionIndex`	A reference to hold the unique index number of the session to be assigned by `Initialise()`.
`int iOceanIndex`	The index number of the ocean the session handle is to navigate. **Note:** If iOceanIndex == PIXE.OCEAN_NEW (-1), a new ocean is created and the session handle is assigned to that ocean.
`ref int iFlags`	iFlags specify read/write privileges. (TO BE ADDED)

Read()

Reads a Jellyfish attribute or element value from the Jellyfish Ocean and returns it in the appropriate data type (the value for attributes or a boolean for elements).

Syntax

```
public object Read (
      int iSessionIndex,
      string sPath,
      ref int iFlags
);
```

Example Usage

The examples below show both how to read values using relative and absolute paths.

- When providing an absolute path, the attribute or element at the address provided will be searched for.
- When a relative path is provided, the requested attribute or element will be searched for within the current Drop that the session cursor is located at the time of the method call.

Note, the session cursor will move to the read attribute or element at the end of the method call if it is found. If the requested object is not found the cursor will reset to its original location at the start of the method call. In this case, the iFlags variable will be set to the appropriate error code (see: Error Codes).

Reading elements

Using a relative path:

```
int iFlags = PIXE.PSML_READ_ELEMENT;
object oToRead = API.Read(iMySession, “BlackHole”, ref iFlags);
```

Using an absolute path:

```
int iFlags = PIXE.PSML_READ_ELEMENT;
object oToRead = API.Read(iMySession,
“psml://psml/Jellyfish/Appearance/BlackHole”, ref iFlags);
```

Reading attributes

Using a relative path:

```
int iFlags = PIXE.PSML_READ_ATTRIBUTE;
object oToRead = API.Read(iMySession, “Name”, ref iFlags);
```

Using an absolute path:

```
int iFlags = PIXE.PSML_READ_ATTRIBUTE;
object oToRead = API.Read(iMySession, “psml://psml/Jellyfish/Name”, ref iFlags);
```

Returns

Reading elements

A boolean is returned - `true` if the element exists in that location else `false`.

Reading attributes

An object of the appropriate data type holding the requested value (if found).

Elements and attributes that are not found

`iFlags` is modified to signify operation success or failure:

- If iFlags = +ve int : SUCCESS
- If iFlags = -ve int : OPERATION FAIL
- If FAIL, the amended iFlags corresponds to an error code (see: Error Codes).

Parameters

Type	Description
`int iSessionIndex`	The session handle to use in the read.
`string sPath`	The path to the value to lookup: – Passing a **named attribute** (e.g. `“Height”`) will return the value of the matching attribute held at the present node location of the `iSessionIndex` (if it exists). – Passing an **absolute path** (e.g. `“psml://Jellyfish1/Appearance/BlackHole/Width”` will return the value held at this location (if it exists).
`ref int iFlags`	Specifies the type of data to lookup: `PIXE.PSML_READ_ELEMENT`: Read searches for an element matching the sPath parameter. `PIXE.PSML_READ_ATTRIBUTE`: Read searches for an attribute matching the sPath parameter.

Write()

Writes a Jellyfish attribute or element value into memory. Attributes also be overwritten using this method. However, all attribute and element names must be unique. Duplicate names are not allowed within the same Drop.

Syntax

```
public void Write (
      int iSessionIndex,
      string sPath,
      object oValue,
      ref int iFlags
);
```

Example Usage

Writing elements

When writing elements the oValue parameter should be null (any value will be ignored). Note that the session cursor does not move into the created nested element when writing; the method ends with the cursor inside the current Drop being written:

1) Drop before method call:

session cursor position

INDEX	NAME	TYPE	VALUE	TYPE	Notes
16	“Jellyfish”	“H”	“2”	“-15”	Drop header.
17	“Name”	“A”	“PSX1”	“string”	An attribute.
18	“”	“”	“”	“”	An empty Drop Molecule.
19	“”	“”	“”	“”	

2) Call `Write()`:

Using a relative path:

```
int iFlags = PIXE.PSML_WRITE_ELEMENT;
API.Write(iMySession, “Appearance”, null, ref iFlags);
```

Using an absolute path:

```
int iFlags = PIXE.PSML_WRITE_ELEMENT;
API.Write(iMySession, “psml://psml/Jellyfish/Appearance”, null, ref iFlags);
```

1) Drop after method call:

INDEX	NAME	TYPE	VALUE	TYPE	Notes
16	“Jellyfish”	“H”	“2”	“-15”	Drop header.
17	“Name”	“A”	“PSX1”	“string”	An attribute.
18	“Appearance”	“E”	“0”	“”	The newly written element. Note that the VALUE = 0. The nested element does not exist in the ocean until the cursor moves into it for the first time.
19	“”	“”	“”	“”	

Writing attributes

Using a relative path:

```
int iFlags = PIXE.PSML_WRITE_ATTRIBUTE;
API.Write(iMySession, “Status”, “Default”, ref iFlags);
```

Using an absolute path:

```
int iFlags = PIXE.PSML_WRITE_ATTRIBUTE;
API.Write(iMySession, “psml://psml/Jellyfish/Appearance/Status”, “Default”, ref
iFlags)
```

Returns

`iFlags` is modified to signify operation success or failure:

- If iFlags = +ve int : SUCCESS
- If iFlags = -ve int : OPERATION FAIL
- If FAIL, the amended iFlags corresponds to an error code (see: Error Codes).

Parameters

Type	Description
`int iSessionIndex`	The session number being used to navigate the Jellyfish Ocean.
`string sPath`	The path to the value to write: – Passing a **named attribute** (e.g. `“Height”`) will write the value into the matching attribute held at the present node location of the `iSessionIndex` (if it exists). – Passing an **absolute path** (e.g. `“psml://Jellyfish1/Appearance/BlackHole/Width”` will write the value into this location.
`object oValue`	The value to write (in the appropriate data type – int, string, bool, or float).
`ref int iFlags`	Specifies the type of data to write: `PIXE.PSML_WRITE_ELEMENT`: Write a new element. `PIXE.PSML_WRITE_ATTRIBUTE`: Write a new attribute or overwrite an existing attribute.

Move()

Moves the session cursor to the specified location. The cursor can either step up to the parent Drop or step into an attribute or element within the current Drop.

Syntax

```
public bool Move (
      int iSessionIndex,
      string sDestination,
      ref int iFlags
);
```

Example Usage

Moving up to the parent

```
int iFlags = PIXE.PSML_MOVE;
API.Move(iMySession,"..", ref iFlags);
```

Moving into nested elements or Drop attributes

```
int iFlags = PIXE.PSML_MOVE;
API.Move(iMySession,"ElementName", ref iFlags);
```

Returns

A boolean is returned: true if the move successfully occurs, else false.

`iFlags` is also modified to signify the operations success or failure:

- If iFlags = +ve int : SUCCESS
- If iFlags = -ve int : OPERATION FAIL
- If FAIL, the amended iFlags corresponds to an error code (see: Error Codes).

Parameters

Type	Description
`int iSessionIndex`	The session number being used to navigate the Jellyfish Ocean.
`string sDestination`	The requested destination to move the cursor to: – When `".."`, the cursor will move to the header of the parent of the Drop where the cursor is located at the start of the method call. – Passing a named attribute or element will start a search for the element or attribute matching that name.
`ref int iFlags`	This variable holds any error codes if the operation fails.

	Note, this method will not execute if the iFlags parameter is set to an error code at the time of calling. It is therefore recommended that this parameter is initialised to: `PIXE.PSML_MOVE` However, any int greater than 0 will suffice.

freeSession()

Ends the referenced session and frees it for use by another user.

Syntax

```
public void freeSession (
      ref int iSessionIndex,
);
```

Example Usage

```
API.freeSession(ref iMySession);
```

Returns

Void.

Parameters

Type	Description
`ref int iSessionIndex`	The session number to be freed.

Error Codes

All error codes are <= 0. In the event of an operation failure, the iFlags parameter will be amended to one of the following error codes.

Code	Name	Description
0	OP_FAIL	Unspecified operation failure.
-1	OP_FAIL_INVALID_PATH	Provided path to psml attribute or element is invalid. One or more element or attribute could not be found.
-2	OP_FAIL_INVALID_CURSOR_POSITION	The session cursor is pointing to a location outside of the Ocean.
-3	OP_FAIL_DUPLICATE_RECORD	The program has attempted to write an attribute or element with a name that already exists in that Drop.
-4	OP_FAIL_WRITE_ERROR	Unable to write.
-5	OP_FAIL_NO_FREE_SESSION	All sessions are currently in use.
-6	OP_FAIL_MEMORY_ERROR	System does not have enough memory to perform this operation.
-7	OP_FAIL_INVALID_PSML	The psml file provided is not valid.
-8	OP_FAIL_XML_LOAD_ERROR	Unable to load the XML file.