pjs.defineTable()

This method defines an physical file (table) or logical file (view) in the calling scope, defines the file's record formats and fields, and creates a file instance that can be used to perform record I/O operations on the file. This method serves the same purpose as an RPG disk file specification.

Parameters

1. Internal name - A String containing the desired internal file name. A variable of this name will be defined in the calling script and will be populated with a record file instance. I/O operations are then performed by calling file-specific API.
2. External name (optional) - A String specifying the external name of the file object. The name can optionally be qualified with a library. If the library is not specified, then the library list is used to resolve the file. If the external name is not specified, then the internal name is used instead. This parameter can be omitted; in this case, the file configuration is passed for parameter 2. This parameter can also be specified as an Object with the following properties in order to implement dynamic external file names:
    - field - The name of a declared character field that hold the external file object name
    - extDesc (optional) - The name of the file object to use for file discovery before the file is opened and before the field is declared. If omitted, the internal name is used.
3. File configuration (optional) - An Object that sets file options using the property names / values shown below. If the external file name parameter is omitted, then the configuration should be passed as parameter 2.

Element name	Element type	Description
read	Boolean	Set to True to allow records to be retrieved
update	Boolean	Set to True to allow records to be updated
delete	Boolean	Set to True to allow records to be deleted
write	Boolean	Set to True to allow records to be inserted
keyed	Boolean	Set to True to enable keyed record access. Otherwise records will be accessed by relative record number
userOpen	Boolean	Set to True to enable user-controlled file open / close using the file.open() and file.close() methods. Otherwise the file will be opened automatically before this method returns and closed automatically when the calling function ends.
infDS	String	Name of a File Information data structure which will be populated with feedback information after each file method call. The data structure must be defined in the calling scope using pjs.define().
qualified	Boolean	Set to True to qualify field names with the internal file and record format names
include	Array	A list of record formats to define in the calling scope. If omitted, all record format names are included.
ignore	Array	A list of record formats to ignore. If omitted, all record format names are included.
rename	Object	A list of record formats to rename. The record format name given by each object property is set to the name given in the corresponding value.
recno	String	The name of a field that receives the relative record number after a successful input operation. The field must be defined in the calling scope using pjs.define().
likeFile	String	The internal name of another file defined using this method in the calling scope. This defines another instance of the file using the same configuration options.
levelIds	Array	A list of level identifiers for each record format in the file. If specified, record format level checks are performed before opening the file. Otherwise, no level checks are done.
userDefinedData	Any	Specifies user-defined general purpose data associated with the table configuration.

prefix	Object / String	Defines a prefix that will be used to partially rename all the fields of the Rich Display File. This property can be defined in 2 different ways.
		1. As an object of the following format:
		`{`
		`"numberOfCharsToReplace": nn, //numeric value indicating the number of characters, if any, in the existing field names to replace with the prefix`
		`"prefix": "xxxx" //string prefix value. If numberOfCharsToReplace is not defined or is 0, then the prefix is appended to the beginning of the field names`
		`}`
		2. As a string value that will be appended to the beginning of the field names.

driver	String	Optionally specifies the database driver to use for this table. Shipped drivers include "IBMi" and "jsonDB". If this configuration option is not specified, the global dbDriver configuration setting is used.

template	Boolean	If set to True, the the declared file cannot be opened and used for I/O, but its record formats and fields can be used as templates for field definitions by using the options 'like', 'likeRec', and 'extName' with pjs.define().
static	Boolean	If set to True, the file will hold its state across calls to the function. If the file is open when the function, then the file will still be open on the next call to the function.
alias	Boolean	If set to True, the alias (alternate) names will be used, if present, for the fields associated with the file and also for subfield names in data structures defined with the 'likeRec' option.
likeFile	String	Set to the name of a previously defined table which will be used to derive the table definition.

renameFields	String	Object containing field names to rename within the Rich Display File; the field names are given as properties of this object and their new names as values of these properties.
		
		For example:
		
		`{`
		`"field1": "newField1",`
		`"field2": "newField2"`
		`}`
		
		If a prefix is defined, the *renameFields* object properties must be defined with the prefix in place in order for those fields to be renamed.
		
		For example, if the property *prefix* is defined as string "EX", then *renameFields* might look like this:
		
		`{`
		`"EXfield1": "newField1",`
		`"EXfield2": "newField2"`
		`}`

commit	Boolean	Set to true to open the file under commitment control. A commitment control environment must be set up prior to opening the file, using the STRCMTCTL command.

Examples

Define table for reading records

```
pjs.defineTable("productsp", { read: true });
productsp.fetchNext();
```

Define table for reading records with a different internal name

```
pjs.defineTable("read", "PRODUCTSP", { read: true });
read.fetchNext();
```

Define a table with a dynamic file name

```
pjs.defineTable("read", { field: "fileName", extDesc: "PRODUCTSP" }, { read: true, userOpen: true });
pjs.define("fileName", { type: 'char', length: 25, varying: true });
fileName = "PRODUCTSL1";
read.open();
read.fetchNext();
```

Defining a table with userOpen and infDS

```
  pjs.defineTable("productsp", { infDS: 'OPNFBK',  userOpen: true,  read: true,  levelIds: [ '4AFA8C636F188' ] });

  pjs.define("OPNFBK", { type: 'data structure', elements: {
    "FILE": { special: '*file' },
    "OPEN_IND": { type: 'boolean' },
    "ODP_TYPE": { type: 'char', from: 81, to: 82 },
    "FILE_NAME": { type: 'char', from: 83, to: 92 },
    "LIBRARY": { type: 'char', from: 93, to: 102 }
  }});

  pjs.define("data", { type: 'char', length: 24, varying: true });

  productsp.open();
  data = OPNFBK.substr(80, 22) + OPEN_IND.toString();
  productsp.close();
  data += OPEN_IND.toString();
```

Exception Handling

An Error instance will be thrown with the following properties:

- message - The message text.
- error - The message id.
- help - The message help text.

Requirements

This API requires the Profound.js Connector module.