

# **IBM i Programming Tool**

## User Guide

# Contents

<b>Contents .....</b>	<b>2</b>
<b>Introduction.....</b>	<b>4</b>
<b>Contents of the application directory .....</b>	<b>4</b>
<b>Running the application.....</b>	<b>4</b>
<b>Overview .....</b>	<b>5</b>
<b>Application parameters.....</b>	<b>6</b>
<i>User name / IBM i server input fields .....</i>	<i>6</i>
<i>Connect/Reconnect button.....</i>	<i>6</i>
<i>LIB, FILE, MBR input fields .....</i>	<i>7</i>
<i>IBM i source type combo box.....</i>	<i>8</i>
Standard names of source files and their types .....	8
<i>PC charset combo box.....</i>	<i>9</i>
<i>IBM i CCSID combo box .....</i>	<i>9</i>
<i>Source line length.....</i>	<i>9</i>
<i>Complete source record .....</i>	<i>9</i>
<i>Overwrite data.....</i>	<i>10</i>
<i>Windows disks combo box.....</i>	<i>10</i>
<b>File systems as trees.....</b>	<b>11</b>
<i>Expanding nodes .....</i>	<i>11</i>
<i>Left tree – PC .....</i>	<i>11</i>
<i>Right tree – IBM i.....</i>	<i>12</i>
Selecting library objects .....	12
<b>Context menus.....</b>	<b>14</b>
<b>Search in multiple files.....</b>	<b>16</b>
<b>Creating directories and files .....</b>	<b>17</b>
<b>Copy methods.....</b>	<b>17</b>
<b>Copy from IBM i to PC.....</b>	<b>17</b>
<i>Source member → PC file.....</i>	<i>17</i>
<i>IFS stream file → PC file .....</i>	<i>18</i>
<i>Save file → PC file .....</i>	<i>18</i>
<b>Copy from PC to IBM i.....</b>	<b>18</b>
<i>PC file → source member .....</i>	<i>18</i>
<i>PC file → IFS stream file .....</i>	<i>19</i>
<i>PC file → save file .....</i>	<i>19</i>
<b>Copy from IBM i to IBM i .....</b>	<b>20</b>
<i>Source member → source member .....</i>	<i>20</i>
<i>Source member → IFS stream file .....</i>	<i>20</i>
<i>IFS stream file → source member.....</i>	<i>20</i>
<i>IFS stream file → IFS stream file.....</i>	<i>20</i>
<i>Save file in library → IFS.....</i>	<i>21</i>
<i>Save file in IFS → library .....</i>	<i>21</i>
<i>Library → library .....</i>	<i>21</i>
<b>Copy from PC to PC .....</b>	<b>21</b>
<b>Displaying files .....</b>	<b>22</b>
<b>Editing files .....</b>	<b>23</b>

<i>Finding text</i> .....	24
<i>Highlighting blocks</i> .....	25
<i>Shifting selected text</i> .....	26
Horizontal selection .....	26
Vertical selection .....	26
<i>Copy, cut and paste selected text</i> .....	27
Horizontal selection .....	27
Vertical selection .....	27
<b>Help for form-based languages</b> .....	<b>28</b>
<b>Displaying and editing in PC – character sets</b> .....	<b>28</b>
<b>Displaying and editing in IBM i – character sets</b> .....	<b>28</b>
<b>Spooled files</b> .....	<b>29</b>
<b>Compilation</b> .....	<b>31</b>
<i>Source type</i> .....	32
Source members .....	32
IFS stream files .....	32
<i>Compile command</i> .....	33
<i>Change library list</i> .....	34
Library pattern .....	34
Current library .....	34
Creating a user library list .....	34
<i>Compiled object</i> .....	35
Library .....	35
Object .....	35
Library pattern .....	35
<i>Perform command</i> .....	35
<i>Spooled files</i> .....	36
<i>Edit</i> .....	36
<i>Job log</i> .....	36

## Introduction

Some functions of the *System i Navigator* ceased to work in Windows 10. Especially simple transfer of files between IBM i and PC, displaying and editing of files and the like.

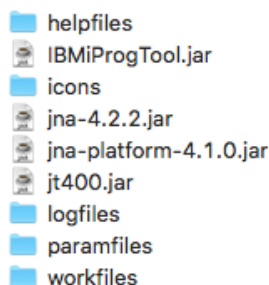
This application provides these functions and, in addition, it enables *editing* and *compiling* programs in the form of source *members* or *stream* files. Compilation messages from listings in *spooled files* may be used for finding and correction of errors.

Programs implementing the application are written in Java and require version *Java SE 8*. They cooperate with classes in *IBM Toolbox for Java* (or JTOpen). The classes require "host servers" running in IBM i and profile QUSER enabled.

The application has been created and tested in systems macOS and Windows 10. Remote connection to the system IBM i, version 7.3 has been used.

## Contents of the application directory

The application does not require an installation. It consists of a directory containing other directories and files:



- The *IBMiProgTool.jar* file is a main, starting file. The application starts after double click with the primary mouse button. A *new version* of the application can be obtained by *replacing this file* by its new version. Other objects may remain unchanged.
- The other *.jar* files are auxiliary programs.
- Directory *helpfiles* contains files for help (this guide) and forms for RPG III, RPG IV, COBOL and DDS.
- Directory *icons* contains icons used in the application windows.
- Directory *logfiles* contains text files *err.txt* a *out.txt*, the redirected output from System.err and System.out files.
- Directory *paramfiles* contains text file *Parameters.txt* with application parameters.
- Directory *workfiles* contains auxiliary files to retain the user library listst, current library name and the recently created spooled file.

Note 1: Files *err.txt* a *out.txt* can be used for revealing the cause of a program error.

Note 2: More files and directories are contained in the open source directory downloaded from internet. They may be deleted without any harm.

## Running the application

The application is started by double clicking on the *IBMiProgTools.jar* file.

Note: The *Help* menu for the application is placed in the menu bar of the main application window (Windows) or in the menu bar of the desktop (macOS).

## Overview

This application allows to

- create files and directories in PC and IBM i,
- remove files and directories in PC and IBM i,
- rename files and directories in PC and IBM i,
- copy files and directories between PC and IBM i,
- display and edit text files in PC and IBM i,
- copy, clear and delete libraries,
- search text in multiple files,
- compile source members and IFS stream files,
- display spooled files.

The application in IBM i works with the following object types:

- Source physical file      type \*FILE, attribute PF,
- Source member            Source physical file member,
- Save file                   type \*FILE, attribute SAVF,
- IFS directory             type \*DIR,
- IFS stream file            type \*STMF,
- Output queue              type \*OUTQ with spooled files inside,
- Library                    type \*LIB.

The *source physical file* is considered a *directory* in the application.

The *source member* is considered a *file* in the application.

Behavior of the application is directed by *parameters* saved in the file "Parameters.txt" in the directory "paramfiles".

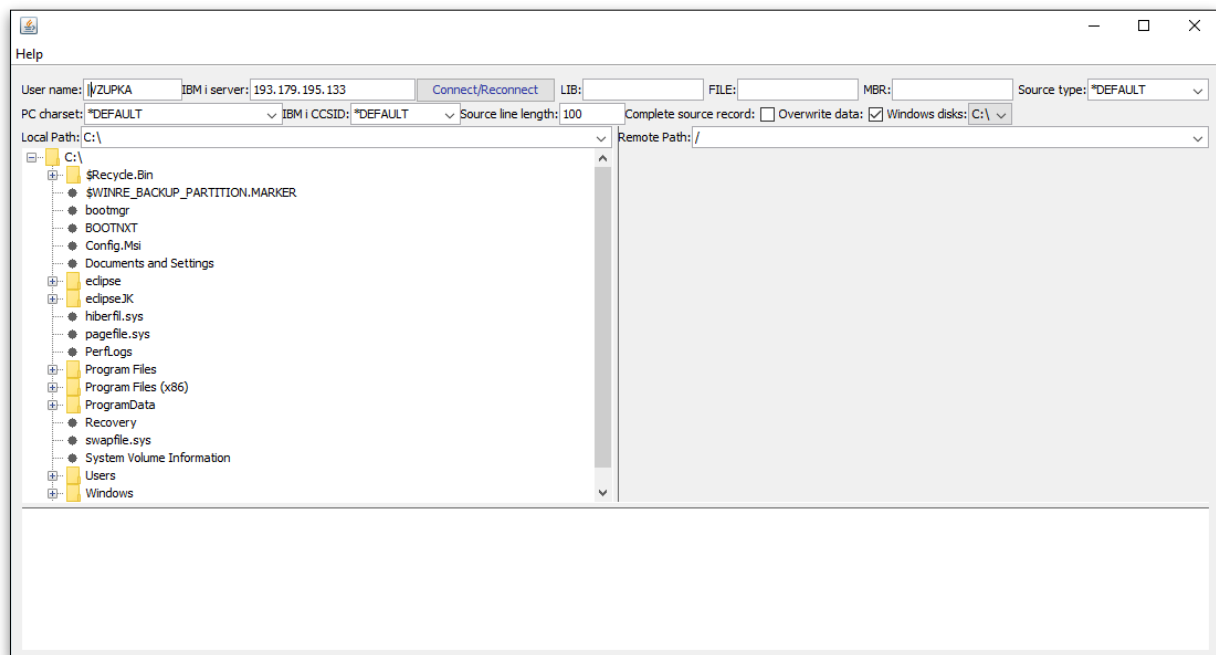
## Application parameters

When the application starts, a window is displayed, where the left side shows the tree representing the PC file system while the right side is empty. The right side is filled after connection to the system IBM i using the button Connect/Reconnect.

The upper part of the window contains components that define application parameters. They have form of input text fields, combo boxes, and check boxes. The button Connect/Reconnect for connection to the IBM i server is also in the upper part.

### *User name / IBM i server input fields*

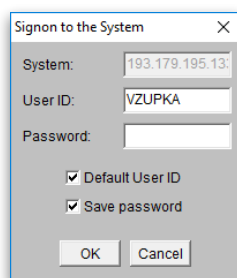
The user name and the address of IBM i server must be entered and then the Connect/Reconnect button must be pressed to connect the server. The other parameters may be adjusted later.



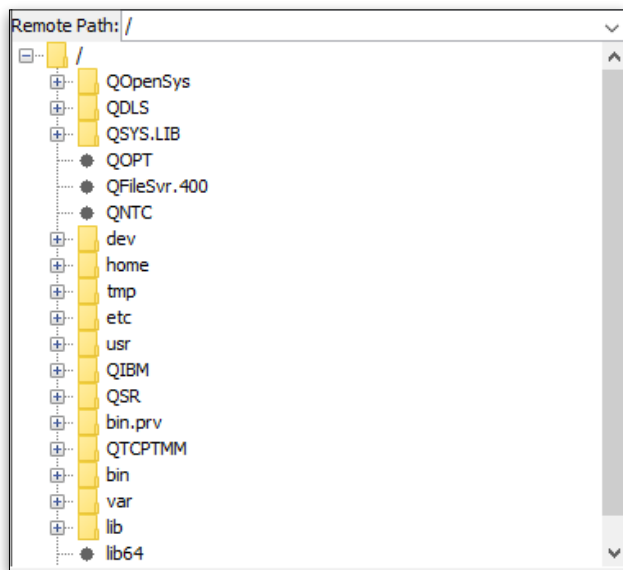
### *Connect/Reconnect button*

When the user connects for the first time, the dialog *Signon to the System* is displayed and the password must be entered. The process of connection may last longer, according to the speed of the communication line. When the button is pressed next, the server is connected again. The new connection may omit the dialog and may last shorter, if the user keeps the server address.

The user can connect another server by changing the address and pressing the Connect/Reconnect button or the *Enter* key.



A tree representing the IBM i file system is shown on the right side of the window after successful connection. All directories of the IFS file system are contained in the tree if the IFS root symbol (/) is entered in the *Remote Path* input field.



### ***LIB, FILE, MBR input fields***

These input fields provide *selection* of objects within the node /QSYS.LIB (the system library). The user may select *libraries*, *files*, or *members* by entering a *search pattern* in the corresponding field. Selection is started by pressing the *Enter* key or clicking the button *Connect/Reconnect*.

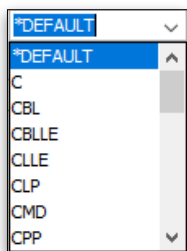
The search pattern may include special characters asterisk (\*) and question mark (?) beside normal characters.

- The \* character matches zero or more characters in the object name.
- The ? character matches one character in a *position* of the object name.

For example, entering the pattern V?T\* in a field selects names starting with V, containing any character in the second position, T in the third position, and any characters in remaining positions.

An empty field selects all names. Entering the exact object name as a pattern selects this specific object.

## IBM i source type combo box



The user selects or enters the source type when a source member is to be copied to PC or to IFS.

If a *new file* is created as a result of the copy operation, it gets a suffix of this type. For example, if the user selects type RPGLE when copying member PROG01.MBR, the newly created file gets name PROG01.RPGLE.

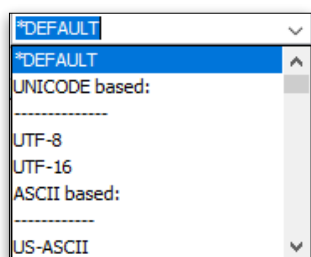
The special value \*DEFAULT assigns the type automatically if the source file has a standard name. For example, if the source file is QRPGLSRC (standard name for ILE/RPG programs), type RPGLE is assigned. If the source file has not a standard name, type TXT is assigned.

### Standard names of source files and their types

QBASSRC	BAS	Basic
QCBLLSRC	CBLLE	ILE/Cobol
QCLSRC	CLLE	CL
QCMDSRC	CMD	Command
QCSRC		C C language
QDDSRC	PF	Physical file
QFTNSRC	FTN	Fortran
QCBLSRC	CBL	Cobol/400
QMAPSRC	CICSMAP	CICS Customer Information Control System
QMENUSRC	MNUDDS	Menu DDS
QMNUSRC	MENU	UIM menu
QPASSRC	PAS	Pascal language
QPLISRC	PLI	PL/I language
QPNLSRC	PNLGRP	Panel group
QREXSRC	REXX	REXX
QRMCSRC	RMC	RM/Cobol-85
QRPGLSRC	RPGLE	ILE/RPG language
QRPGSRC	RPG	RPG/400
QS36PRC	OCL36	System/36 Operator Control Language
QS36SRC	UNS36	S36 unspecified
QSRVSR	BND	Binder source
QTBLSRC	TBL	Table
QTXTSRC	TXT	Text
QUDDSRC	QRY38	Query/38



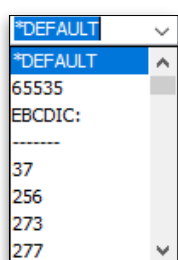
## PC charset combo box



The user can select a character set from the combo box or enter its code in the input text field.

The *PC charset* parameter is applied when copying data and when displaying files in some situations. A special value *\*DEFAULT* is also in the combo box list. It is interpreted differently according to context. See details in following chapters.

Note: Unlike IFS files, PC files have no character set attribute. Therefore the *PC charset* parameter is not applied in creating a new PC file.



## IBM i CCSID combo box

The user can select a CCSID code ((Coded Character Set ID) from the combo box or enter a code in the input text field.

A special value *\*DEFAULT* is also in the combo box list.

The *IBM i CCSID* parameter is applied when copying data and when displaying files. It is applied also in creating a new source physical file, in creating a new IFS stream file, or in displaying and copying spooled files. Details are explained in the next chapters.

## Source line length

This value defines length of the source text line when *creating* a new *source physical file*. For example, for the source physical file QDDSSRC, length 80 is defined as a standard, for the file QRPGLSRC, length 100 is defined, etc. Different length can be entered and confirmed by the *Enter* key.

When copying a PC text file to a source member, the line (or its text part) of the PC file can be longer than the length of the data part in the source member (defined in creating the source file). Then the line is shorten so that it fits in the data part of the source record.

## Complete source record

The check box, if checked, means that a 12-character data is prepended to each output line when copying a source member from IBM i to PC. This data contains the sequential number and the date from the source record.

If the check box is unchecked, only the data part from the source member is copied to PC.

Opposite, when copying a PC text file to a source member, this check box is *not applied*. The application is directed by the first line of the PC file. If the first two 6-character values are whole numbers, they are prepended to the source record.

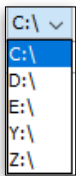
If not, missing data is derived as the computed sequential number (from 1.00 by 1.00) and the date of zero. These numbers are prepended to the line from the PC file and written together as a record to the source member.

Tip: The complete source record option is useful only if the numbers and dates are to be preserved later when restoring the member from the PC backup. The copy operation is rather slow in this case.

### **Overwrite data**

The check box, if checked, means that it is allowed to overwrite data in existing files. If unchecked, data in existing files cannot be overwritten neither at copying nor at editing.

### **Windows disks combo box**



The list of disks is available only in *Windows*. In fact, it is a list of root directories of the Windows file system. The user selects one from the list. Disk C:\ is default.

## File systems as trees

The PC file system on the left and the IBM i file system on the right are presented as trees. The *root* of the tree is placed in the first row.

The root of the file system is the root of the tree on the first start of the application.

- For Windows it is disk C:\.
- For unix type systems it is the forward slash / (called root).
- For IBM i it is the forward slash / like in unix type systems.

The user can change the tree root using the combo box. Gradually, as the user expands individual *nodes* of the tree, paths to corresponding objects are entered to the *list* of the *combo box*. If the user then selects a path from the list (usually a directory), the path becomes the root of the new tree and the new tree is shown.

The user may also enter the path to the object into the input field of the combo box manually and press *Enter* key. Thus the new root is set and a new tree is shown.

The application saves the actual root in its parameters. On the next start of the application, the recently saved root and the corresponding tree is displayed.

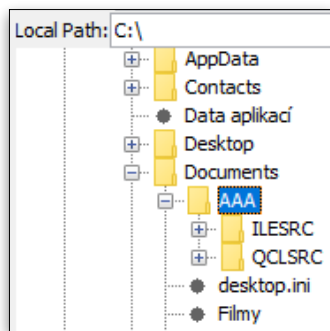
### Expanding nodes

Expand a node by *single click* on its *text* or *icon* with the left mouse button. This is necessary in order to load second level nodes with appropriate information for their further expansion.

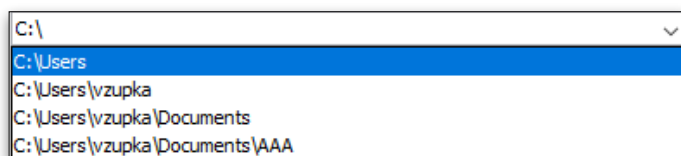
Warning: Do not use *double click* and do not use the *expansion symbol* (+ sign in a box in Windows or a little triangle in macOS)! This would work only after the node was regularly expanded before by clicking on its text or icon. After regular expansion, the node can be collapsed or expanded again.

### Left tree – PC

Clicking on a node (AAA in the picture) with the *left mouse button* reveals its objects (directories ILESRC and QCLSRC).

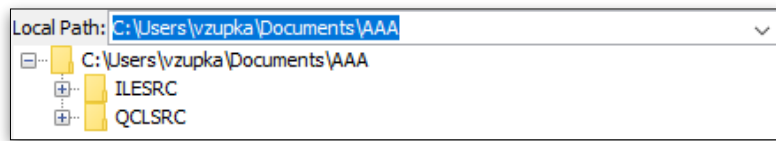


At the same time, the path leading to the node gets entered into the *combo box* list. Expanding the combo box "Local Path", will show all paths leading to the nodes expanded until now:



If you select the last directory from the list, its path gets entered in the input field and a subtree is displayed from the new root:

C:\Users\vizupka\Documents\AAA.



The *right mouse button* on a node is used to display *context menus* with different commands.

### **Right tree – IBM i**

Selecting objects and a root is done the same as in the left tree. Use the "Remote path" *combo box* as described above with "Local path".

Moreover, library objects can be selected using *patterns* in the input fields *LIB*, *FILE*, *MBR*. To start the selection you click on the *Connect/Reconnect* button or press the *Enter key* (when the caret stays in an input field).

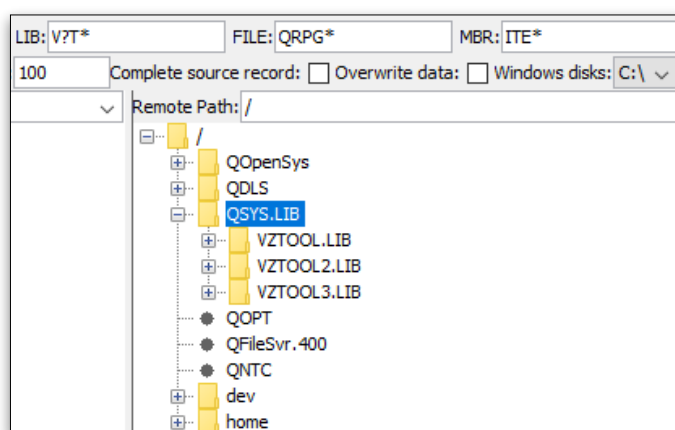
If you select library objects (within the node /QSYS.LIB) using solely the "Right tree" combo box, you get a subtree containing only these library objects. The other IFS objects get lost from view.

If you want to see also the other IFS objects, use input fields *LIB*, *FILE*, *MBR*. Then, if you expand the /QSYS.LIB node, you get a new tree with selected library objects along with the other IFS objects. This method may be useful when copying between library objects and other IFS objects.

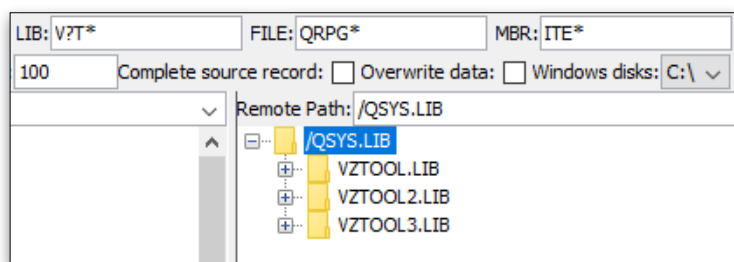
Both methods may be combined.

### **Selecting library objects**

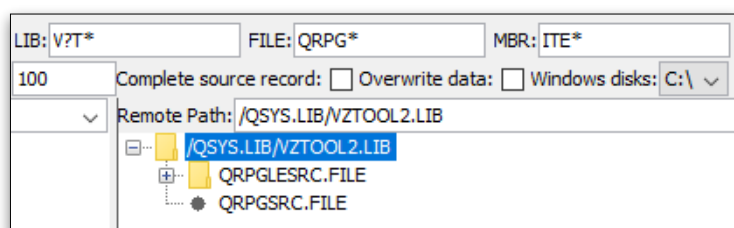
If you select objects using the LIB, FILE, MBR input fields only and having the IFS system root path (/) in the "Remote path" combo box input field, you get a subtree with selected library objects hidden in the node /QSYS.LIB. To see the objects expand this and the next nodes. The other IFS objects are retained in view.



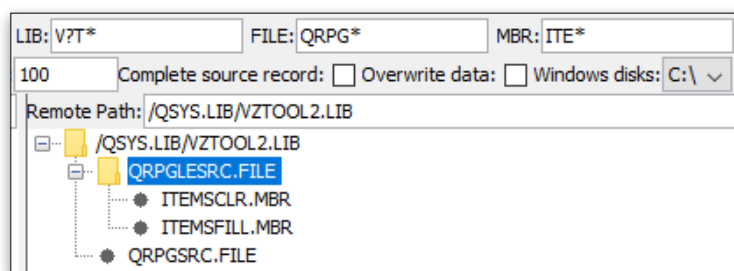
If you enter the *path* /QSYS.LIB into the "Remote path" combo box input field and *patterns* in the LIB, FILE, MBR input fields, you get subtree with the root /QSYS.LIB without other IFS objects:



If you enter the path /QSYS.LIB/VZTOOL2.LIB into the "Remote path" combo box input field and the patterns in the LIB, FILE, MBR input fields, you limit the tree to the single library VZTOOL2 with selected files



and selected members:

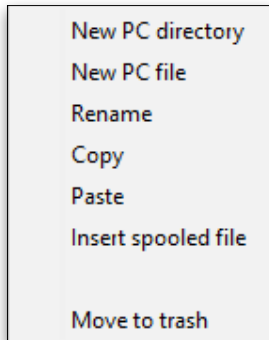


## Context menus

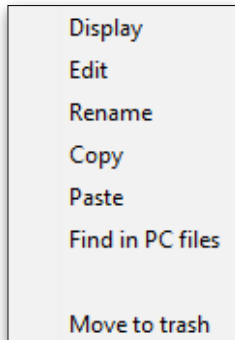
Click by right button of the mouse (right click) on a one or more *selected nodes* invokes a context menu with commands.

On the left side, only two kinds of menus are available.

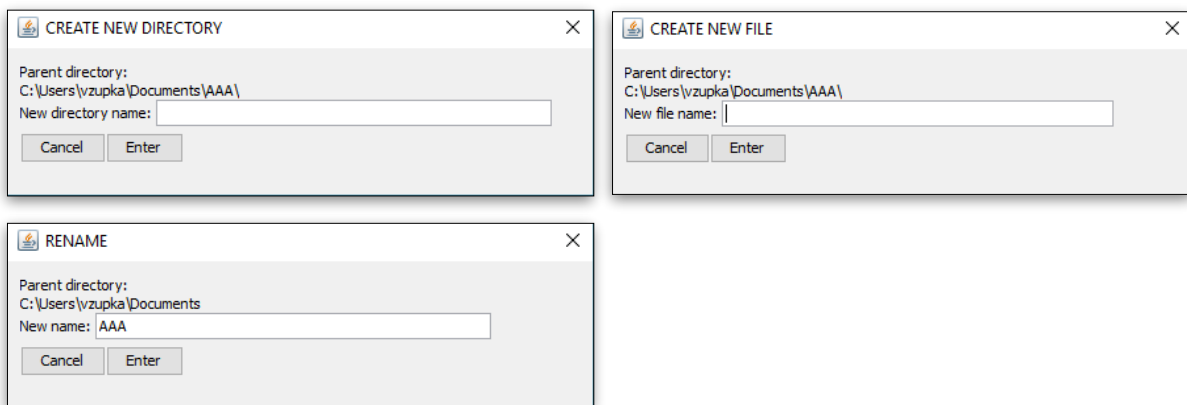
### *PC directory*



### *PC file*



Commands *New PC directory*, *New PC file*, *Rename* invoke a dialog of the unified form. The command *Insert spooled file* is explained in the chapter *Spooled files*.



- *Edit* command invokes the text editor (see chapter *Editing files*). The editor can be invoked also by *double click* on a *PC file node*.
- *Copy* command *remembers* files or directories from *selected nodes* (source nodes).
- *Paste* command *inserts* or *replaces* files or directories in a *target node*.
- *Find in PC files* command on *selected files* invokes a window where a text pattern can be entered. Files that contain the pattern are listed in the window. See chapter *Search in multiple files*.

**Note 1:** Beware of copying to multiple target nodes. Source nodes are copied *to the first* of the target nodes.

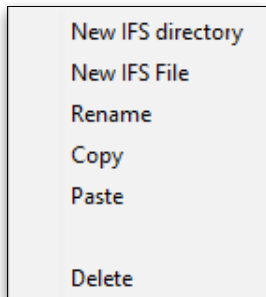
**Note 2:** Use Copy and Paste commands when both the source and target nodes are *not in the same view* and the method *drag and drop* cannot be used.

On the right side, some commands have the same function as on the left side.

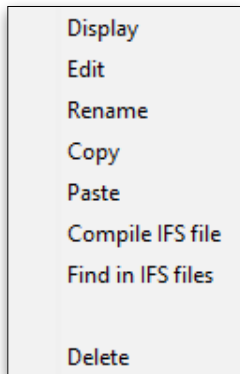
*IFS stream file* and *Source member* have commands *Compile IFS file* and *Compile source member* (see chapter [Compilation](#)) and also commands *Find in source members* and *Find in IFS files*. See chapter [Search in multiple files](#).

Other commands are self explaining.

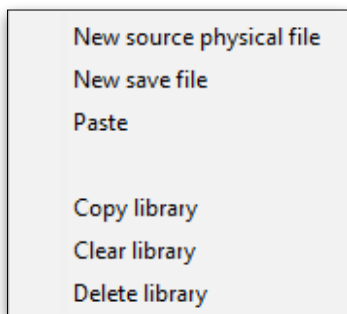
#### ***IFS directory***



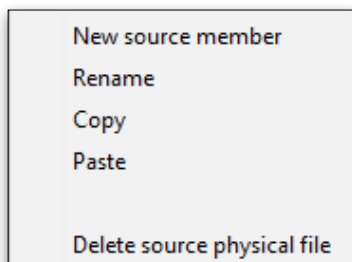
#### ***IFS stream file***



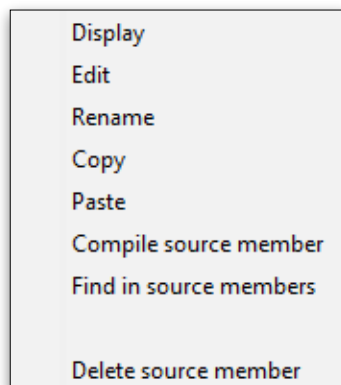
#### ***Library***



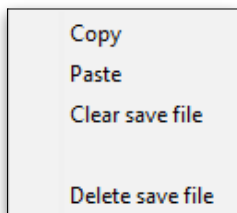
#### ***Source physical file***



#### ***Source member***



#### ***Save file***



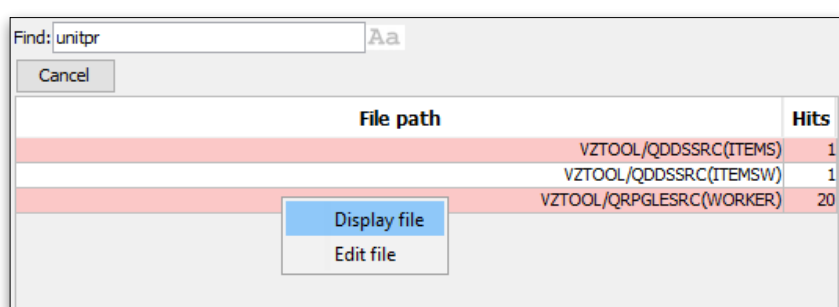
## Search in multiple files

Select one or more files (PC files, source members, IFS files) that may or may not be adjacent. Command *Find in . . .* from the context menu invokes a window where a text pattern is entered and Enter key pressed.

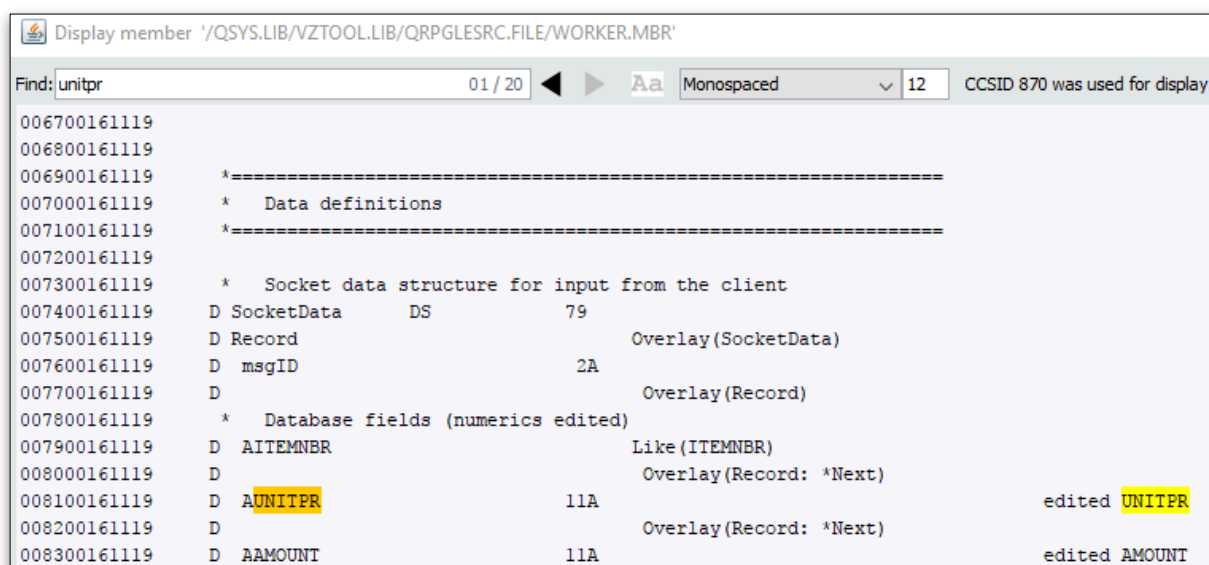
If the "Match case" (**Aa**) icon is light gray, search is done for any letter case. If it is black (**Aa**), search is done for the same letter case.

If one or more files from the selected ones contain the text pattern the files are listed in the window. The user can select one or more of them and make them displayed or edited. Clicking on right button reveals context menu with commands *Display file* and *Edit file*. These commands open display or edit window showing all pattern instances highlighted depending on the Match case button color.

For example, multiple (non adjacent) members were selected and command *Find in source members* was chosen. The user entered pattern *unitpr* in the *Find* field and pressed Enter key. The pattern was found in three source members.



The user selected two rows and chose the command *Display file*. Two members, display file ITEMS and program WORKER are displayed with all instances of the text found highlighted.





## Creating directories and files

Select command *New. . .* from the *context menu* on a selected node.

The new *source physical file* gets its CCSID from the *IBM i CCSID* parameter.

- If the parameter is **\*DEFAULT** it is replaced by CCSID 500 (**EBCDIC** ISO-8859-1, Latin Alphabet No. 1).
- Length of the data in source records is obtained from Source line length parameter. The whole record is 12 bytes longer. A sequence number in the first 6 bytes and a date in the form YYDDMM in the next 6 bytes are prepended to the data.
- The new source member gets its CCSID from its source physical file in which it is created as its attribute. Length of records is obtained from the source physical file "Maximum record length" attribute.

The new *IFS directory* gets the CCSID from the *IBM i CCSID* parameter as its attribute. If the parameter is **\*DEFAULT** it is replaced by CCSID 819 (**ASCII** ISO-8859-1, Latin Alphabet No. 1).

The new *IFS file* gets the CCSID from the *IBM i CCSID* parameter as its attribute. If the parameter is **\*DEFAULT** it is replaced by CCSID 819 (**ASCII** ISO-8859-1, Latin Alphabet No. 1).

Directories and files in the PC do not contain any information about a character set.

## Copy methods

Files and directories can be copied in all directions:

- IBM i ↔ PC
- PC ↔ IBM i
- IBM i ↔ IBM i
- PC ↔ PC

Two copy methods can be used. The first method makes use of commands *Copy* and *Paste* in context menus. The other method is *drag and drop*.

### Copy from IBM i to PC

- source member → PC file
- source member → PC directory
- source physical file → PC directory
- IFS stream file → PC file
- IFS stream file → PC directory
- IFS directory → PC directory
- save file → PC file
- save file → PC directory

#### **Source member → PC file**

Data from the source member is first translated from character set defined in the *IBM i CCSID* parameter to an auxiliary text coded UTF-16 and then translated to the character set defined in the *PC charset* parameter.

- If *\*DEFAULT* is entered in the *IBM i CCSID* parameter, data is translated into the auxiliary text using *CCSID attribute* of the source member.
- If *\*DEFAULT* is entered in the *PC charset* parameter, data from the auxiliary text is translated using character set *ISO-8859-1* (Latin-1).

Note: The source physical file can be created with the *CCSID* attribute 1208. The source member created in this source physical file has the same attribute. Then, if the *IBM i CCSID* parameter or *CCSID* attribute is 1208 (UTF-8 equivalent) it is replaced by value 65535 in translation to the auxiliary text. Thus, data is not translated to UTF-16 as individual bytes but the whole UTF-8 characters.

### ***IFS stream file* → *PC file***

Data from the *IFS stream file* is translated from the character set defined in the *IBM i CCSID* parameter to the character set defined in the *PC charset*.

If *\*DEFAULT* is entered in both parameters, data is transferred without change (binary).

Otherwise, data is translated using the *IBM i CCSID* parameter into an auxiliary text encoded in UTF-16 and then translated using the *PC charset* parameter.

- If *\*DEFAULT* is entered only in the *IBM i CCSID* parameter, data is translated into the auxiliary text from the stream file using the *CCSID* attribute.
- If *\*DEFAULT* is entered only in the *PC charset* parameter, data from the auxiliary text is translated to the PC file using character set *ISO-8859-1* (Latin-1).

### ***Save file* → *PC file***

A save file is placed in a library and has suffix *.FILE*. After copying to PC, the newly created file gets suffix *.savf*. Existing PC file must have suffix *.savf* in order that the save file can be copied into it.

## **Copy from PC to IBM i**

- PC file → source member
- PC file → source physical file
- PC directory → source physical file
- PC directory → library
- PC file → IFS stream file
- PC file → IFS directory
- PC directory → IFS directory
- PC file → save file

### ***PC file* → *source member***

Data from the PC file is translated from the character set defined in *PC charset* parameter to the character set defined by the *CCSID attribute* of the source member. Parameter *IBM i CCSID* is *ignored*. Resulting characters need not be in compliance with the target character set.

Note 1: If an error is reported in copying files, an empty file may be created.

Note 2: Data transfer may last quite long, if character sets differ and the source member is long (3000 rows about 2 minutes). The program must read individual input lines and translate each line before writing.

### ***PC file → IFS stream file***

Data from the PC file is translated from the character set defined in *PC charset* parameter to the character set defined by the *IBM i CCSID* parameter.

If *\*DEFAULT* is entered in both parameters, data is transferred without change (binary).

Otherwise, data is translated using the *PC charset* parameter into an auxiliary text encoded in UTF-16 and then translated using the *IBM i CCSID* parameter.

- For *\*DEFAULT* in the *PC charset* parameter, character set *ISO-8859-1* (ASCII ISO-8859-1, Latin Alphabet No. 1) is assigned.
- For *\*DEFAULT* in the *IBM i CCSID* parameter, value *819* (ASCII ISO-8859-1, Latin Alphabet No. 1) is assigned.
- If character sets correspond on both sides, data is transferred without translation in the following cases:
  - ISO-8859-1 → CCSID 819, CCSID 858
  - windows-1250, Cp1250 → CCSID 1250
  - windows-1251, Cp1251 → CCSID 1251
  - UTF-8 → CCSID 1208
  - UTF-16 → CCSID 1200, CCSID 13488
  - IBM500, Cp500 → CCSID 819
  - IBM870, Cp870 → CCSID 870

### ***PC file → save file***

Any PC file denoted by suffix *.savf* is considered a save file and can be copied to the save file in a library with suffix *.SAVF*. The same PC file can also be copied to an IFS file or directory without changing the suffix, and still retain its content as a save file.

## Copy from IBM i to IBM i

- source member → source member
- source member → source physical file
- source physical file → library
- source member → IFS stream file
- source member → IFS directory
- source physical file → IFS directory
- save file in library → IFS
- save file in IFS → library
- library → library
- IFS stream file → IFS stream file
- IFS stream file → IFS directory
- IFS directory → IFS directory

### **Source member → source member**

Source member data is *translated according to source physical files* in which they reside. That is, from the CCSID attribute of the input source physical file to the CCSID attribute of the output source physical file.

The *IBM i CCSID* parameter is *ignored*.

### **Source member → IFS stream file**

Data from the source member to an *existing* IFS stream file is translated to the CCSID attribute of the IFS stream file.

Data from the source member to an IFS stream file, that is *just being created*, is not translated and the new IFS stream file takes over the CCSID attribute from the *source physical file*, in which the source member resides.

In both cases the *IBM i CCSID* parameter is *ignored*.

### **IFS stream file → source member**

Data from IFS stream file is translated from the CCSID *attribute* of the IFS stream file into the CCSID *attribute* of the *source physical file*, no matter if the source member already exists or is just being created.

The *IBM i CCSID* parameter is *ignored*.

### **IFS stream file → IFS stream file**

If the *IBM i CCSID* parameter is \*DEFAULT or both files have identical CCSID attributes, data is transferred *without change (binary)*.

Otherwise, data of the input IFS file is translated from its CCSID attribute to the CCSID attribute of the output IFS file.

If a *non-existent* output file is *just being created*, the newly created IFS file takes the CCSID attribute of the input IFS file and data is thus transferred *without change (binary)*.

### **Save file in library → IFS**

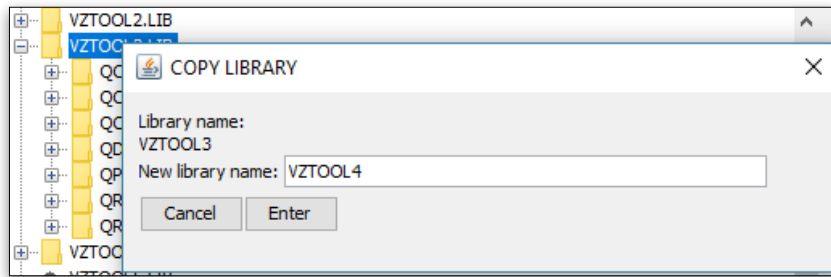
A save file in a library has suffix *.SAVF*. If the output IFS stream file already exists, it must have suffix *.savf*. The newly created output IFS stream file gets suffix *.savf*.

### **Save file in IFS → library**

An IFS directory with suffix *.savf* is considered save file. If the output save file already exists in the library, it has suffix *.FILE*. The newly created save file gets suffix *.FILE*.

### **Library → library**

A library can be copied under a different, user defined name.

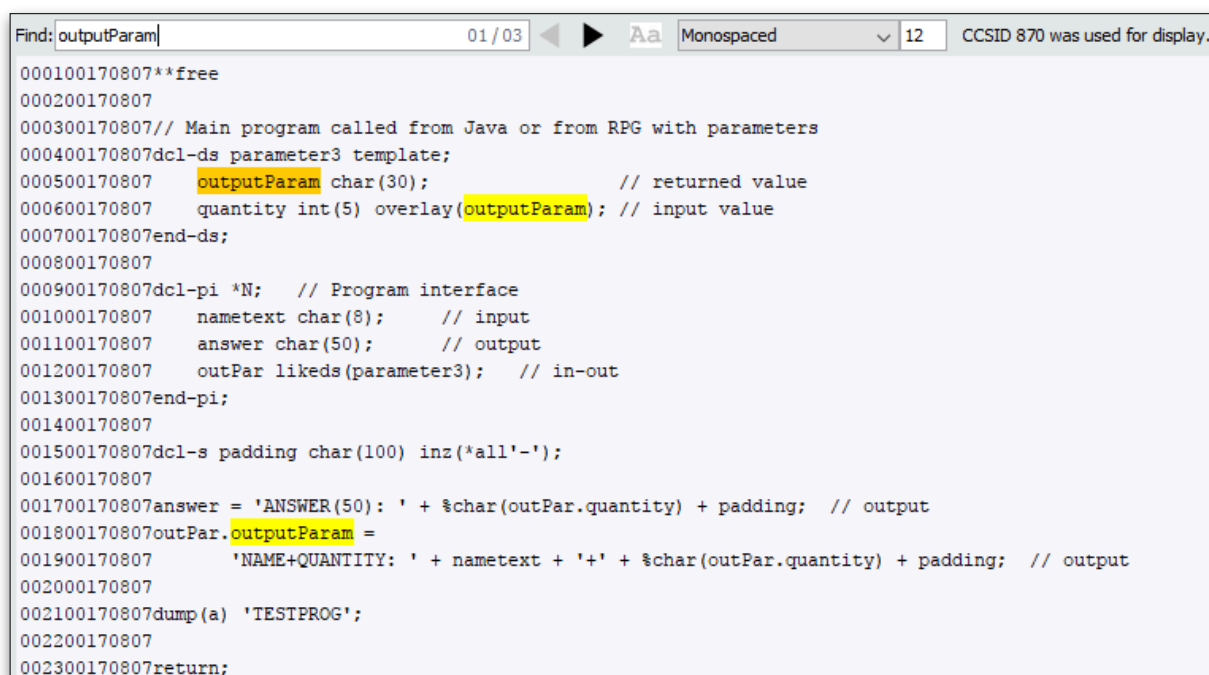


### **Copy from PC to PC**

- PC file → PC file
- PC file → PC directory
- PC directory → PC directory

No conversion is performed in copying data.

## Displaying files



The screenshot shows a window titled "Find: outputParam" with a search bar containing "01/03". The window displays a C program with several lines of code. The search results for "outputParam" are highlighted in yellow. The code includes comments and function calls related to the search term.

```
000100170807**free
000200170807
000300170807// Main program called from Java or from RPG with parameters
000400170807dcl-ds parameter3 template;
000500170807     outputParam char(30);           // returned value
000600170807     quantity int(5) overlay(outputParam); // input value
000700170807end-ds;
000800170807
000900170807dcl-pi *N; // Program interface
001000170807     nametext char(8); // input
001100170807     answer char(50); // output
001200170807     outPar likeds(parameter3); // in-out
001300170807end-pi;
001400170807
001500170807dcl-s padding char(100) inz(*all'-');
001600170807
001700170807answer = 'ANSWER(50): ' + %char(outPar.quantity) + padding; // output
001800170807outPar.outputParam =
001900170807     'NAME+QUANTITY: ' + nametext + '+' + %char(outPar.quantity) + padding; // output
002000170807
002100170807dump(a) 'TESTPROG';
002200170807
002300170807return;
```

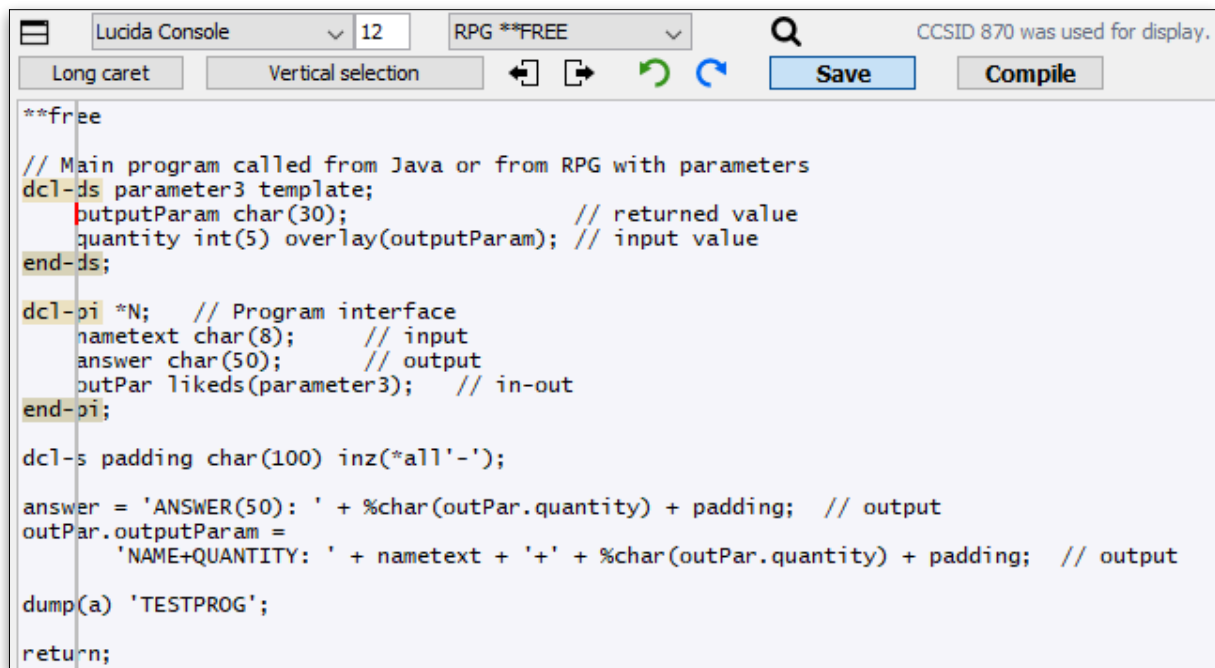
Contents of a file is displayed (on Display command) in a separate window with the information about character set of the text in the file.

- *Find* – entering a text pattern in the field causes finding matching texts. Numbers in the field show sequence number of the current match and number of matches.
- *◀▶ Arrow buttons* – find preceding or following matches in the file. An arrow gets black after clicking on it and indicates direction of searching. The opposite arrow gets gray. The same function is provided by keyboard shortcuts Ctrl ↑ (previous) and Ctrl ↓ (next).
- *Aa/Aa toggle button* – when light gray, the search is case insensitive, when black, the search is case sensitive.
- *Font size* – the input field defines font size of the text in the displayed file.

*ESC* key escapes display and removes the window.

Note: *Cmd* key is used instead of Ctrl in macOS.

## Editing files




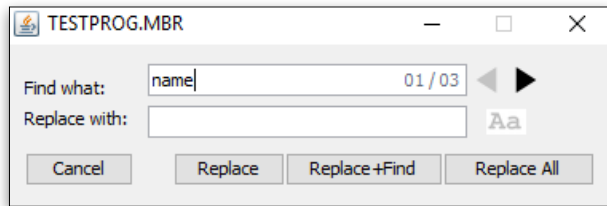
- *Split/unsplit* - toggle split/unsplit editor text vertically by a movable horizontal line into upper and lower area. A copy of the editor text is in the lower area. All text changes made in one area are automatically projected in the other area.
- *Lucida Console etc.* – choose a font and its size from the combo box.
- *RPG \*\*FREE etc.* – choose the programming language suitable to the editor text from the combo box to highlight blocks (compound statements).
- *Find text* – invokes a window to find text. Shortcut Ctrl F may also be used.
- *Long caret/Short caret* – defines the pointer in text as a long vertical line or a short vertical line (standard).
- *Horizontal/Vertical selection* – defines method of selecting text. Horizontal selection is a common method in PC editors. Vertical selection selects a rectangular area in the text.
- *Shift selection* – button shifts selected text one position to the left, button shifts selected text to the right. Keyboard shortcuts Ctrl ← and Ctrl → may also be used.
- *Undo* – remove changes. Keyboard shortcut Ctrl Z may also be used.
- *Redo* – restore changes. Keyboard shortcut Ctrl Y may also be used.
- *Save* – save changes. Keyboard shortcut Ctrl S may also be used.
- *Compile* – shows a window for compilation (see [below](#)) if the edited file is a source member or an IFS file.



*ESC* key escapes editing (without saving) and removes the window.

Note: *Cmd* key is used instead of Ctrl in macOS.

## Finding text

Clicking on the magnifying glass  button or pressing keyboard shortcut Ctrl F (Cmd F in macOS) invokes the following window.

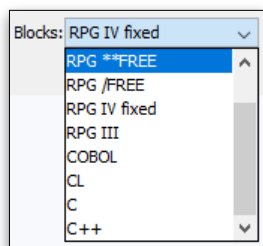


- *Find what* – enter a text pattern to be found in the editor area. Numbers in the field show sequence number and number of matches.
-   *Arrow buttons* – find preceding or following matches in the file. An arrow gets dark on clicking on it and indicates direction of searching and replacing. The opposite arrow gets gray. The same function is provided by keyboard shortcuts Ctrl ↑ (previous) and Ctrl ↓ (next).
- *Replace with* – enter a text replacement.
- **Aa/Aa** *toggle button* – when light gray, the search is case insensitive, when black, the found text must exactly match the the pattern.
- *Replace* – replace the text just found by the replacement
- *Replace+Find* – replace the matching text just found by the replacement and find the *next* matching text.
- *Replace All* – replace *all* matching texts by the replacement text.

Note: Cmd key is used instead of Ctrl in macOS.



## Highlighting blocks



The combo box contains list of programming languages whose compound statements (blocks) should be highlighted. The list contains the following entries:

**\*NONE** – no highlighting,

**\*ALL** – blocks for all languages are highlighted,

**RPG \*\*FREE** – RPG version with completely free statement entries,

**RPG /FREE** – RPG version allowing both fixed and free statements in calculations,

**RPG IV fixed** – RPG version allowing extended fixed form statements,

**RPG III** – RPG version (RPG/400) allowing traditional fixed form statements,

**COBOL** – COBOL language,

**CL** – ILE/CL Control language,

**C** – C language,

**C++** – C++ language,

**SQL** – SQL language script.



Note: An SQL script cannot be run (not to say compiled).

The method of highlighting of compound statements used in this application may have some unwanted effects, because short sequences of letters – namely IF, DO, FOR, may appear also in other places in the text. This might be acceptable for relevant programming languages, however. The lowest efficiency in this sense is for C and C++ languages and for option **\*ALL**.

A block of an RPG program with *RPG /FREE* or *RPG \*\*FREE* option:

```
begsr browseAttributes;
  dou *in03;
    exfmt ATRWIN;
    if *in03;
      exsr endpgm;
    elseif *in12;
      leavesr;
    endif;
  enddo;
endsr; // browseAttributespgm
```

## Shifting selected text

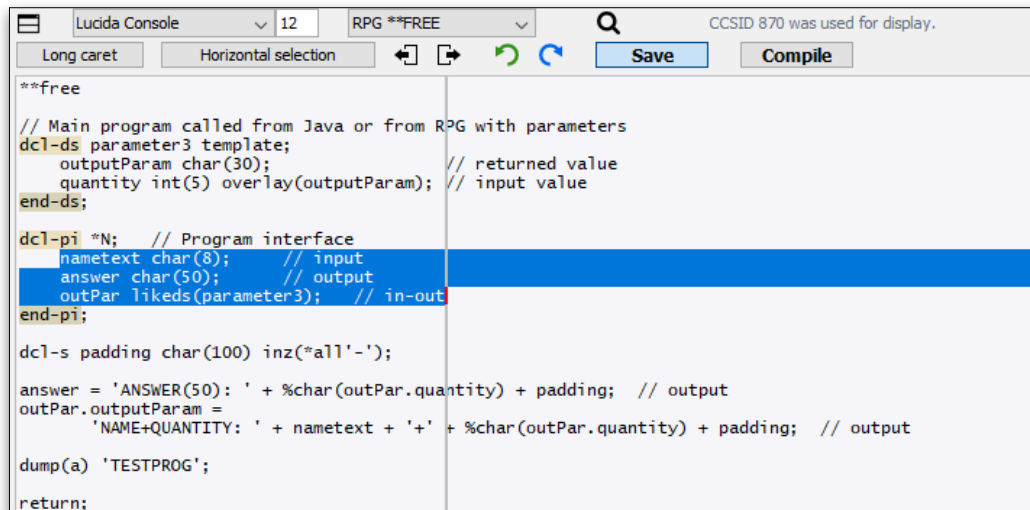
Left  and Right  buttons shift the selected text left or right by one position. The keyboard shortcuts Ctrl ← and Ctrl → do the same functions. In macOS the Cmd key is used instead of Ctrl. Results of shifts may be undone or redone.

### Horizontal selection

The selection shifts *right* along with the unselected rest of the last line.

The selection shifts *left* along with the rest of the lines if it begins at the start of a line and if all lines of the selection contain at least one space at the start.

In the followin picture, the selection was shifted right.



```
Lucida Console 12 RPG **FREE CCSID 870 was used for display.
Long caret Horizontal selection
**free
// Main program called from Java or from RPG with parameters
dcl-ds parameter3 template;
  outputParam char(30); // returned value
  quantity int(5) overlay(outputParam); // input value
end-ds;

dcl-pi *N; // Program interface
  nametext char(8); // input
  answer char(50); // output
  outPar likeds(parameter3); // in-out
end-pi;

dcl-s padding char(100) inz(*all'-');

answer = 'ANSWER(50): ' + %char(outPar.quantity) + padding; // output
outPar.outputParam =
  'NAME+QUANTITY: ' + nametext + '+' + %char(outPar.quantity) + padding; // output

dump(a) 'TESTPROG';

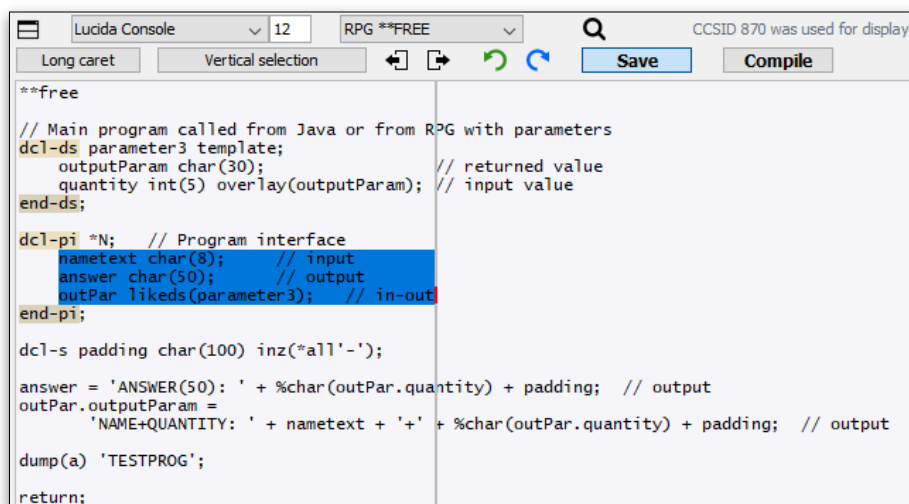
return;
```

### Vertical selection

The selected *rectangle* shifts *left* by one position if there remains at least one column of spaces in the unselected part on the left of the rectangle. The rectangle *overwrites* the columns on the left and leaves a column of spaces behind (on the right).

The selected *rectangle* shifts *right* and leaves a column of spaces behind (on the left). The rectangle is shifted right as long as there is a space on the right.

In the following picture, the rectangle was shifted left.



```
Lucida Console 12 RPG **FREE CCSID 870 was used for display.
Long caret Vertical selection
**free
// Main program called from Java or from RPG with parameters
dcl-ds parameter3 template;
  outputParam char(30); // returned value
  quantity int(5) overlay(outputParam); // input value
end-ds;

dcl-pi *N; // Program interface
  nametext char(8); // input
  answer char(50); // output
  outPar likeds(parameter3); // in-out
end-pi;

dcl-s padding char(100) inz(*all'-');

answer = 'ANSWER(50): ' + %char(outPar.quantity) + padding; // output
outPar.outputParam =
  'NAME+QUANTITY: ' + nametext + '+' + %char(outPar.quantity) + padding; // output

dump(a) 'TESTPROG';

return;
```

## Copy, cut and paste selected text

Common command shortcuts Ctrl C, Ctrl X, Ctrl V are used to copy, cut and paste selected text. In macOS, the Cmd key is used instead of Ctrl.

Copy and Cut operations store the selected text in the *operating system clipboard*.

Paste operation reads data from the clipboard and inserts it to the desired place. This may be in the editor area or elsewhere in PC.

Results of these operations may be undone or redone.

### Horizontal selection

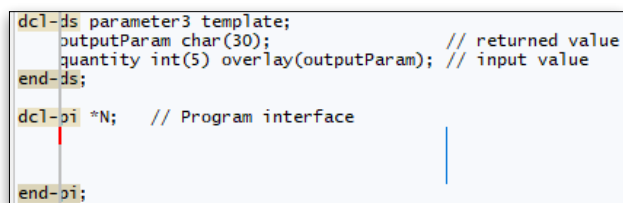
All these operations work as usual in PC.

### Vertical selection

*Copy* operation copies the selection into an internal area and also into the system clipboard.

*Cut* operation copies the selection into an internal area and also into the system clipboard, then *clears* (puts spaces in) *the rectangle area*.

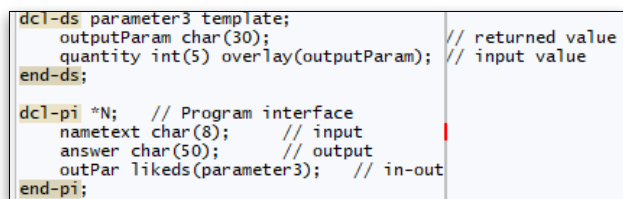
For example, the rectangle from the picture above was cut:



```
dcl-ds parameter3 template;
    outputParam char(30);           // returned value
    quantity int(5) overlay(outputParam); // input value
end-ds;
dcl-pi *N; // Program interface
end-pi;
```

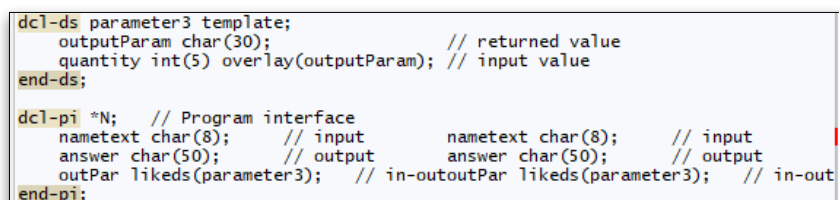
Note the *caret* position, it stands at the *beginning* of the cut rectangle area. The blue line at right denotes the right edge of the rectangle. If desired, a following *paste operation* inserts the erased data back into its original positions.

*Paste* operation replaces the editor area right and down from the *caret position*. It may be the originally copied or cut area.



```
dcl-ds parameter3 template;
    outputParam char(30);           // returned value
    quantity int(5) overlay(outputParam); // input value
end-ds;
dcl-pi *N; // Program interface
    nametext char(8); // input
    answer char(50); // output
    outPar likeds(parameter3); // in-out
end-pi;
```

This time the *caret* stands at the *right top* edge of the rectangle. An immediate following paste operation inserts data on the right of the caret.



```
dcl-ds parameter3 template;
    outputParam char(30);           // returned value
    quantity int(5) overlay(outputParam); // input value
end-ds;
dcl-pi *N; // Program interface
    nametext char(8); // input
    answer char(50); // output
    outPar likeds(parameter3); // in-out
end-pi;
```

If the editor area is shorter than the inserted rectangle, extra empty lines are appended and the pasted data is inserted in the extra lines.

```
dc1-ds parameter3 template;
  outputParam char(30);           // returned value
  quantity int(5) overlay(outputParam); // input value
end-ds;

dc1-pi *N; // Program interface
  nametext char(8); // input
  answer char(50); // output
  outPar likeds(parameter3); // in-out
end-pi;

dc1-s padding char(100) inz(*all'-');

answer = 'ANSWER(50): ' + %char(outPar.quantity) + padding; // output
outPar.outputParam =
  'NAME+QUANTITY: ' + nametext + '+' + %char(outPar.quantity) + padding; // output

dump(a) 'TESTPROG';

return;
nametext char(8); // input
answer char(50); // output
outPar likeds(parameter3); // in-out
```

## Help for form-based languages

Languages RPG III, older versions of RPG IV, COBOL and DDS use forms to enter specifications. *Form headings* are available in *Help menu* which may be copied and inserted in the edited text at suitable places as a comment and lead the programmer to enter program specifications in proper columns. For example, the following RPG IV form heading may be inserted in the program at the beginning of file descriptions:

```
.....F*filename++IPEASFRlen+LKlen+AIDevice+.Keywords++++++Comments++++++
```

## Displaying and editing in PC – character sets

Files are displayed and edited using *PC charset* parameter. If \*DEFAULT is entered, ISO-8859-1 is assigned. If the file contains invalid characters, an error message is reported. The user can change the parameter and try again.

## Displaying and editing in IBM i – character sets

*Source members* are displayed and edited using their *CCSID attributes* without regard to the value of the *IBM i CCSID* parameter. Characters are displayed incorrectly if they do not conform to the CCSID.

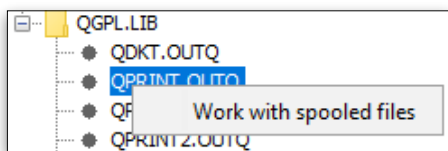
*IFS stream files* are displayed and edited using their *CCSID attributes* without regard to the value of the *IBM i CCSID* parameter. Characters are displayed incorrectly if they do not conform to the CCSID.

*Spooled files* are displayed (but not edited) using the *IBM i CCSID* parameter. If incorrect characters are displayed, using \*DEFAULT or 65535 may help. The program then tries to guess a correct encoding.

## Spooled files

Spooled files can be displayed and can also copied to the PC. There are two ways how to get spooled files.

The first way requires knowledge about *what libraries* contain output queue objects (\*OUTQ type), for example libraries QGPL or QUSRSYS. Click with the right mouse button on the node having suffix .OUTQ, which invokes menu with a single command *Work with spooled files*.



The other way does not require library names. Use the system library QSYS.LIB instead. The system library does not contain any output queues directly but it serves here as a placeholder of the menu with the command *Work with spooled files*. Thus *all* spooled files can be accessed.

The command *Work with spooled files* invokes a window with a table of spooled files.

File name	File num.	Pages	Job name	User	Job num.	Date	Time	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
				QSYS				
				VZUPKA				
File name	File num.	Pages	Job name	User	Job num.	Date	Time	Output queue
QPRINT	1	1	QSLPSVR	QSYS	032508	1170331	134516	QGPL/QPRINT
CENPOR	6	10	QPRTJOB	VZUPKA	033626	1170404	183428	QGPL/QPRINT
TESTPROG	7	7	QPRTJOB	VZUPKA	033626	1170405	102552	QGPL/QPRINT

End work with spooled files by the closing the window by *ESC* key.

To limit range of the table, enter conditions to input fields above the table and press *Enter* key. The text in the field is sought in the corresponding table column. Conditions entered in more fields are evaluated at the same time. An empty field does not limit the table.

The *User* field is a combo box of profiles (user names) to whom the spooled files belong. Selecting a user name limits the table to spooled files belonging to this user.

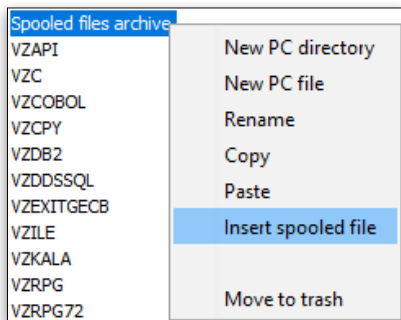
*Double click* on a selected row displays the *single* spooled file *directly* in a separate window (see [Displaying files](#) above).

To further work with spooled files you can select *one* or *multiple* table rows by *left mouse click*. Contiguous rows are selected holding the *Shift* key, non-contiguous rows are selected holding the *Ctrl* key (*Cmd* in macOS). If you *right click* on the selected rows, the following menu with three commands will show:

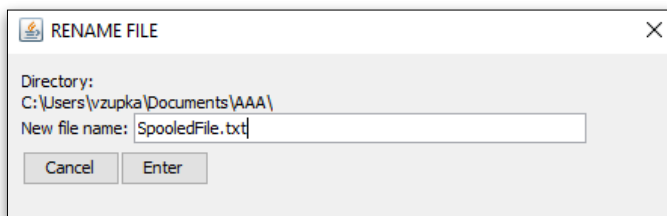
QPJOBLOG	104	4	QPRTJOB	QSYS	1170414	000023	QUSRSYS/QEZJOBLOG	
QPJOBLOG	105	7	QPRTJOB	QSYS	1170414	000025	QUSRSYS/QEZJOBLOG	
QPJOBLOG	20	14	QPRTJOB	VZUPKA	1170414	094333	QUSRSYS/QEZJOBLOG	
QPJOBLOG	106	4	QPRTJOB	QSYS	1170415	000031	QUSRSYS/QEZJOBLOG	
QPJOBLOG	107	7	QPRTJOB	QSYS	1170415	000033	QUSRSYS/QEZJOBLOG	
VEKDS04	21	8	QPRTJOB	VZUPKA	033626	1170415	072849	QGPL/QPRINT

Commands *Display* and *Copy* write the spooled file text into an internal file *SpooledFile.txt* in directory *workfiles*. The *Display* command also displays the text in a separate window (see [Displaying files](#) above).

The internal file can be copied under the same or another name into a PC directory of choice so that it can be further manipulated (e. g. printed or sent by e-mail). Select command *Insert spooled file* from the menu under a PC directory node in the left tree:



A dialog prompting for a file name is shown:

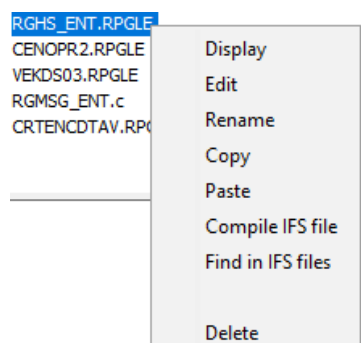


After changing or leaving the name and pressing the Enter key the file is written into the directory.

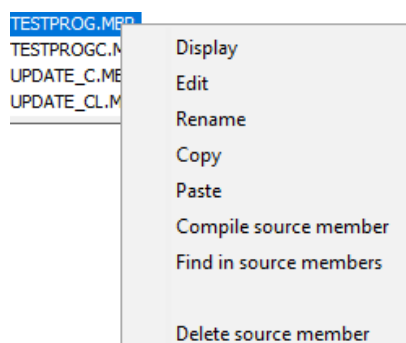
## Compilation

Context menu on some files in IBM i contains a command for compilation. For IFS stream files choose command *Compile IFS file*, for source members choose command *Compile source member*.

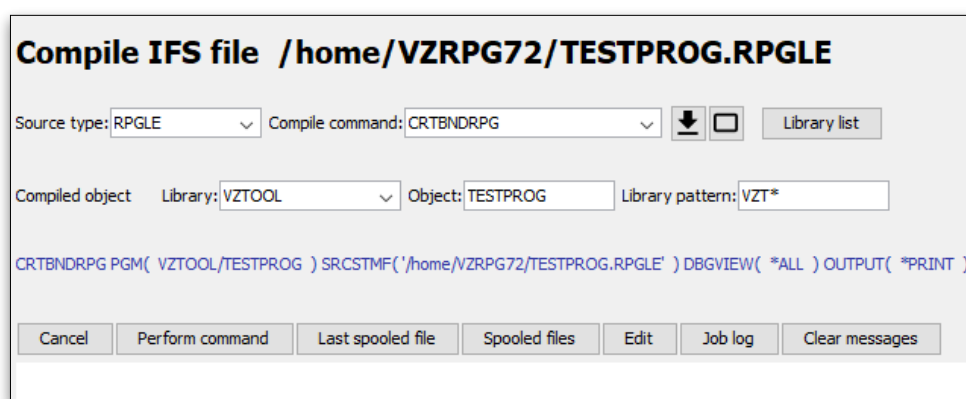
### IFS file



### Source member



Another way to originate compilation is clicking the *Compile* button in the text editor. The compilation window is displayed.



The window is structured into a few rows.

1. The first row shows the *path* to the source file or the IFS stream file.
2. The second row contains parameters necessary to identify *input* for compilation.
3. The third row contains default parameters identifying the *output* of the compilation:
  - Target *Library* name.
  - Target *Object* name.
  - Selection *pattern* for limiting the list of libraries in the combo box – may be entered in the input field.
4. The fourth row shows the *actual* text of the compilation command.
5. The fifth row contains buttons:
  - Cancel* – cancel the work. *ESC* key or closing the window also cancels the work.
  - Perform command* – perform the compilation command.
  - Last spooled file* – displays the most recently produced spooled file for the current user.
  - Spooled files* – invoke table of spooled files for the current user.
  - Edit* – invoke a window with the source text to edit.
  - Job log* – print the actual contents of the job log.
  - Clear messages* – clear all messages from the message area.

## Source type

The *source type* can be selected from the combo box or entered in the input field. Programs and description files of the following source types can be compiled:

CLLE, CLP,  
RPG, RPGLE, SQLRPG, SQLRPGLE,  
CBL, CBLLE, SQLCBL, SQLCBLLE,  
C, CPP, SQLC, SQLCPP,  
CMD,  
DSPF, LF, PF, PRTF,  
TBL

## Source members

For source members, all suffixes are the same: *MBR*. The source type of a member is derived from the name of its source physical file. If the name is standard, the member gets its source type. If the name is not standard, the member gets type *TXT*. *Standard names* of source physical files are as follows:

QCLSRC	CLLE	Control language
QRPGLESRC	RPGLE	ILE/RPG
QRPGSRC	RPG	RPG/400
QCBLLSRC	CBLLE	ILE/Cobol
QCBLSRC	CBL	Cobol/400
QCSRC	C	C language
QCMDSRC	CMD	Command
QDDSSRC	PF	Physical file
QTBLSRC	TBL	Table

## IFS stream files

For IFS stream files, suffixes listed above must be explicitly written, not necessarily with capital letters. IFS stream files of the following source types *cannot be compiled*:

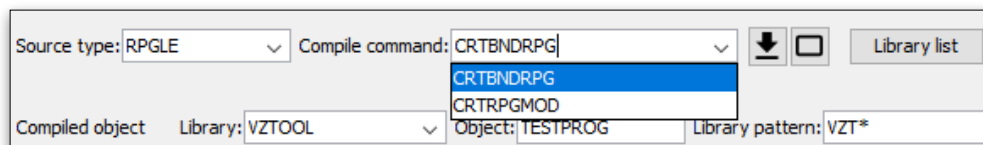
CLLE, CLP, CMD, RPG, CBL, SQLRPG, SQLCBL  
DSPF, LF, PF, PRTF,  
TBL

Note: A file of SQL type cannot be neither run nor compiled.



## Compile command



The actual CL *compilation command* can be left unchanged or it may be selected from the combo box. Changing is sometimes necessary because some source types can be compiled as a *program* or a *module* or a *service program*.



A program or a module can be created from the ILE source types: CLLE, RPGLE, CBLLE, C, CPP, SQLRPGLE, SQLCBLLE, SQLC a SQLCPP.

A service program can be created from source types with SQL statements: SQLRPGLE, SQLCBLLE a SQLC.

Two buttons with icons enable remembering or forgetting *attributes of compilation* (source type, compile command, target library, target object) selected from combo boxes and a text field.

The "Save" button that has two switchable icons  and , defines how the selected values are handled. When the button has the *black* icon, the values are *saved* (assigned) for each compiled file. Thus the *attributes of compilation* (source type, compile command, target library, target object) are fixed and represent a user defined default. They appear in the combo boxes each time the file is chosen for compilation, until the user changes them.

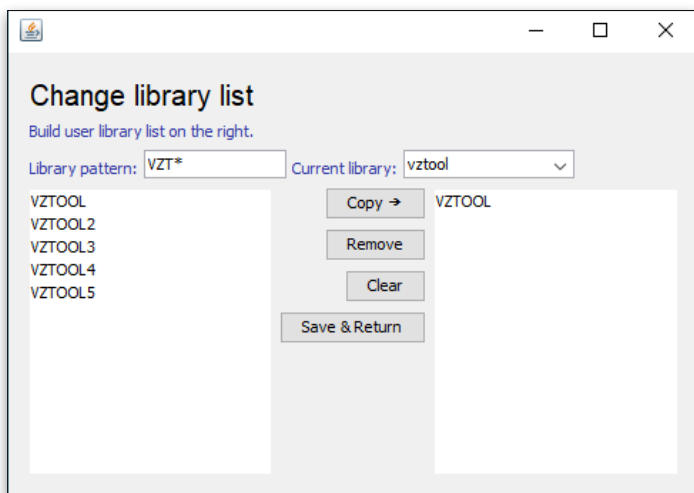
When the button has the *light gray* icon, the values are *not saved*.

The "Clear" button with  icon when clicked, *clears* saved values for all compiled files.

Note: Saved values are written in the file "CompileAttributes.lib" in directory "workfiles".

## Change library list

This button invokes a window with an overview of the *user library list* and the *current library*:



### Library pattern

Entering a *search pattern* (see chapter [LIB, FILE, MBR input fields](#)) in the input field *Library pattern* and pressing the *Enter key* writes a list of libraries whose names conform to the pattern to the *left* frame. If the input field is empty (possibly after clearing the field and pressing *Enter key*), the list of all libraries will be shown.

### Current library

The current library can be selected from the *Current library* combo box or entered in the input field. The symbol *\*CRTDFT* is a special entry which means that the job library list does *not* contain a current library.

### Creating a user library list

A single library or multiple libraries can be selected from the left frame and copied to the right frame either by *drag and drop*, or by **Copy →** button. The right frame represents the *user library list* of the job.

The *Remove* button removes selected libraries from the right frame.

The *Clear* button clears the right frame.

The *Save & return* button saves the changes.

## Compiled object

Compiled object	Library: VZTOOL	Object: TESTPROG	Library pattern: VZTOOL
-----------------	-----------------	------------------	-------------------------

When compiling a *source member* default parameters in this line are derived from the path to the source member. They can be changed and saved by Enter key.

When compiling an *IFS file* it may be necessary to change default parameters target library and object name. Default target library name is taken from the first item of the combo box list. Default target target object name is taken from the *path* and is shorten to 10 characters. The default parameters cannot be directly derived from the path to the IFS file like they can for a source member.

### Library

The library can be selected from the list in the combo box or entered in the input field and press Enter key.

### Object

The object name can be left unchanged or entered in the input field.

### Library pattern

Entering a pattern in the input field and pressing the Enter key writes a list of libraries whose names match the pattern to the combo box for selection. The pattern may contain wildcards \* and ?.

## Perform command

The *Perform command* button *runs the compilation*. One or more messages are reported about the result of the compilation in the low part of the window, e. g.:

RNS9304 \*INFORMATIONAL: Program TESTPROG placed in library QGPL. 00 highest severity.  
Created on 02/08/17 at 14:09:16.

Cause . . . . . : Program TESTPROG was successfully created in library QGPL. The highest message severity that resulted was 00. The program creation date and time are 02/08/17 and 14:09:16.

A compilation protocol (listing) is printed in the print file QPRINT.

## Spooled files

The Spooled files button shows a table of spooled files for the *current user*. Selection, displaying, copying, and deleting is performed the same as in OUTQ object types (see [Spooled files](#) above).

The window contains the *Refresh* button which refreshes the table of the spooled files without using the Spooled files button repeatedly.

File name	File num.	Job name	User	Job num.	Date	Time	
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="v"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Refresh"/>

File name	File num.	Job name	User	Job num.	Date	Time	Output queue
CENY2	1	QPRTJOB	VZUPKA	029937	1170311	094622	QGPL/QPRINT

*ESC* key removes the spooled files window.

## Edit

The *Edit* button displays contents of the text for editing. The text can be changed, saved, and compiled again.

## Job log

The *Job log* button prints the actual contents of the job log in the print file QEZJOBLOG. This file can be found using the Spooled files button and displayed like other text files.