



User Guide

ARCAD-Transformer

RPG

Version 10.06.xx

Publication Date: May, 2016

Prepared by the ARCAD Software Documentation Team



www.arcadsoftware.com



| | | |
|--------------------|---|-------------------------|
| FRANCE (HQ) | 55 Rue Adrastée 74650 Annecy/Chavanod | Tel. +33 4 50 57 83 96 |
| GERMANY | Richardstr. 84 22089 Hamburg | Tel. +49 40 357 09 10-2 |
| SWEDEN | Prostvägen 36 141 43 HUDDINGE | Tel. +46(0) 70-793 6570 |
| USA | 1 Phoenix Mill Lane, Suite 203 Peterborough, NH 03458 | Tel. +1 (603) 371-9074 |
| HONG KONG | Room 22, Smart-Space 3F, 100 Cyberport Road | Tel. +852 3618 6118 |

Copyright © 1992, 2016 by ARCAD Software All rights reserved.

The following terms are names owned by International Business Machines Corporation in the United States, other countries, or both: AS/400®, ClearCase, ClearQuest®, DB2, DB2 Connect™, DB2 Universal Database™, ibm.com, IBM i, iSeries, System i, OS/400, Rational®, SP2, Service Pack, WebSphere. Java and all names based on Java are owned by Oracle Corp. in the United States, other countries, or both. Eclipse is a registered trademark of Eclipse Foundation, Inc. Other names of companies, products or services are the property of their respective owners.

Contact ARCAD Software

arcadsoftware.com

Headquartered in France at the foot of the Alps, ARCAD Software offers global services and has offices on three continents.

| Country | Address | Telephone |
|-------------|--|---|
| FRANCE (HQ) | Arcad Software 55 Rue Adrastée 74650 Annecy/Chavanod | Tel. +33 4 50 57 83 96 Fax +33 4 50 57 52 79 sales-eu@arcadsoftware.com |
| | Arcad Software 67 Rue du Bourbonnais 69009 Lyon | Tel. +33 9 72 42 09 57 |
| GERMANY | ARCAD Software Deutschland GMBH Richardstr. 84 22089 Hamburg | Tel. +49 40 357 09 10-2 sales@arcadsoftware.de |
| SWEDEN | ARCAD Software Nordic AB Prostvägen 36 141 43 HUDDINGE | Tel. +46(0) 70-793 6570 sales-se@arcadsoftware.com |
| USA | ARCAD Software Inc. 1 Phoenix Mill Lane, Suite 203 Peterborough, NH 03458 | Tel. +1 (603) 371-9074 Toll Free +1 (800) 676-4709 sales-usa@arcadsoftware.com |
| ASIA | ARCAD Software Asia Room 22, Smart-Space 3F, Unit 908-915, Level 9, Cyberport 3, 100 Cyberport Road HONG KONG | Tel. +852 3618 6118 sales-asia@arcadsoftware.com |

Table 1: Contact ARCAD Software

ARCAD Software guarantees consultant support 24 hours a day, 5 days a week (24/5) to registered members. Calls received on the numbers for France, USA and Asia are redirected to a central system which, according to the hour, puts you in contact with a consultant in or near your timezone.

| Country | E-mail | Telephone |
|---------|--|-------------------------|
| FRANCE | support-eu@arcadsoftware.com | Tel. +33 4 50 57 28 00 |
| GERMANY | support@arcadsoftware.de | Tel. +49 40 357 09 10-2 |
| SWEDEN | support-se@arcadsoftware.com | Tel. +46(0) 70-793 6570 |
| USA | support-usa@arcadsoftware.com | Tel. +1 (603) 574-4860 |
| ASIA | support-asia@arcadsoftware.com | Tel. +852 8192 5740 |

Table 2: Contact the ARCAD Software support team

Contact an ARCAD Partner

ARCAD partners with leading-edge companies throughout the world to ensure you have the local support you need and best-of-breed contacts for all of your software management solutions. Our global partners are located strategically around the globe to offer full services, close to home.

Visit our website to view the complete list of partners and their local contact information.

arcadsoftware.com/company/partners/

The ARCAD Customer Portal

The [ARCAD Customer Portal](#) is intended for current customers and organizations that have downloaded full or trial versions of ARCAD software. If you already use or are interested in using an ARCAD product, the portal lets you view all of your current licenses and generate your own temporary license keys for most ARCAD products. It grants you access to all of ARCAD's Release Notes and to the ARCAD product knowledge base (FAQ, new releases, fixes, etc.) and ticketing system as well as all of the documentation for each of your products.

Preface

Document Purpose

This document is intended to guide you through configuring and using ARCAD-Transformer RPG.

Reference For more information about installing this product, refer to the *ARCAD-Transformer RPG Installation Guide*.

Intended Audience

This document is intended for all ARCAD-Transformer RPG users.

About the glossary terms

Glossary terms are marked with orange text and defined in footers in this document. The definitions are truncated. For complete definitions of all of ARCAD's terminology, refer to the *ARCAD Glossary*.

Publication Record

Unless stated otherwise, this document is valid for the most current version of ARCAD-Transformer RPG listed as well as every subsequent version.

| Product Version | Document Version | Publication Date | Update Record |
|-----------------|------------------|------------------|---|
| ≥ 10.06.xx | 2.6 | May, 2016 | Modified the CVT_CALL parameter to remove the possibility to prevent the replacement of CALL/CALLB with prototyped calls when the program is an array element variable. |
| | 2.5 | February, 2016 | Updated MAXNOTFREE instructions for "Fully-Free" sources. Revised description of how Conversion Units are generated and used. |
| | 2.4 | November, 2015 | Added License Key Hold option to transfer license to another machine. Added parameters FULLYFREE, MAXNOTFREE and FIRSTCOL. |
| 10.05.04 | 2.3 | September, 2015 | Moved the Conversion Engine installation and update instructions to the <i>ARCAD-Transformer RPG Installation Guide</i> . Updated pagination. |
| 10.05.04 | 2.2 | June, 2015 | No functional changes. Added tables of tables and figures. |

Table 3: ARCAD-Transformer RPG User Guide Publication Record

| Product Version | Document Version | Publication Date | Update Record |
|-----------------|------------------|------------------|--|
| 10.05.00 | 2.1 | June, 2015 | Added multiple language option. Added parameters INDENTCMT, OPCODECASE, BLTFNCCASE and SPCWRDCASE. |
| 10.04.11 | 2.0 | May, 2015 | Initial Publication |

Table 3: ARCAD-Transformer RPG User Guide Publication Record

Related Documentation

| Related Documentation |
|--|
| Release Notes – version 10.06.00.pdf |
| Release Notes – version 10.05.06.pdf |
| Release Notes – version 10.04.10.pdf |
| ARCAD-Glossary |
| ARCAD-Transformer RPG Installation Guide |

Table 4: Related Documentation

Product Information

ARCAD-Transformer RPG 10.06.xx Copyright © 2016 Arcad Software

[Visit the website \(Transformer RPG\)](#)

Contents

| | |
|--|-----------|
| Contact ARCAD Software | 3 |
| Preface | 5 |
| Contents | 7 |
| Tables | 10 |
| Figures | 12 |
| 1 About ARCAD-Transformer RPG | 13 |
| 1.1 Business Context | 13 |
| 1.2 Functional Constraints | 14 |
| 2 About the Conversion Process | 15 |
| 2.1 The Four Conversion Processes | 15 |
| 2.1.1 The Unitary Conversion Process | 15 |
| 2.1.2 The Conversion Process | 15 |
| 2.1.3 The RPG to RPGLE Conversion Process | 16 |
| 2.1.4 Free Form Conversion Process | 16 |
| 2.2 The Conversion Engine | 16 |
| 2.2.1 About the ACVTRPGFRE Command | 17 |
| 2.2.2 About Operation Codes | 18 |
| 2.2.2.1 Opcodes that are Never Transformed | 19 |
| 2.2.2.2 Opcodes that are only Occasionally Transformed | 19 |
| 2.2.2.3 Opcodes with Error Management Using (e) | 19 |
| 2.2.2.4 Opcodes that manage %Found or %Equal | 20 |
| 2.2.3 About the Conversion Operations | 22 |
| 3 Configuring ARCAD-Transformer RPG | 36 |
| 3.1 Managing Activation Keys | 36 |
| 3.1.1 Activating Keys | 37 |
| 3.1.1.1 Viewing Activation Key Details | 37 |
| 3.1.1.2 Registering an Activation Key | 38 |
| 3.2 Transferring Keys to Different Machines | 38 |
| 3.3 Changing the Language | 39 |
| 3.4 Managing the Free Form Conversion Options | 40 |
| 3.5 Managing the Default RPG to RPGLE Conversion Options | 63 |
| 4 Preparing the ARCAD-Transformer RPG Environment | 64 |
| 4.1 Create a Connection | 64 |
| 4.2 Login to a Connection | 65 |
| 4.3 Create a Library for Modernized Source(s) | 65 |
| 4.3.1 Creating a Library from the Command Line | 66 |

| | | |
|-----------|---|-----------|
| 4.3.2 | Creating a Library from the Remote Systems View | 67 |
| 4.4 | Create a Source File | 68 |
| 4.4.1 | Duplicating a Source from the Command Line | 68 |
| 4.4.2 | Duplicating a Source from the Remote Systems View | 69 |
| 4.4.3 | Removing Empty Members from a Duplicate Object | 70 |
| 4.5 | Create a Source Member Filter | 70 |
| 4.6 | Create a Library List | 71 |
| 4.6.1 | Adding a Library Entry from the Command Line | 72 |
| 4.6.2 | Adding a Library Entry from the Remote Systems View | 73 |
| 4.6.3 | Configuring a Permanent Library List | 73 |
| 5 | About the ARCAD-Transformer RPG Perspective | 75 |
| 5.1 | The Conversion List View | 75 |
| 5.2 | The Conversion List Editor | 75 |
| 5.3 | Filtering the Items in a Conversion List | 75 |
| 6 | Selecting a Source Member | 77 |
| 6.1 | Select a Member from the i Projects view | 77 |
| 6.2 | Select a Member from the Remote Systems view | 78 |
| 7 | Launching Single-File Conversions | 80 |
| 7.1 | Executing a Single-File Conversion | 80 |
| 7.2 | Prompting the ACVTRPGFRE Command | 86 |
| 8 | Launching Mass Conversions | 88 |
| 8.1 | Working with Conversion Lists | 88 |
| 8.1.1 | Creating Conversion Lists | 88 |
| 8.1.1.1 | Add Conversion List from Wizard | 89 |
| 8.1.1.2 | Add Conversion List from Conversion List View | 89 |
| 8.1.2 | Populating Conversion Lists | 89 |
| 8.1.3 | Editing Conversion Lists | 90 |
| 8.1.4 | Deleting Conversion Lists | 90 |
| 8.1.5 | Editing Source Members | 91 |
| 8.1.5.1 | Editing Original Source Members | 91 |
| 8.1.5.2 | Editing Converted Source Members | 91 |
| 8.2 | Defining Object Types | 91 |
| 8.3 | Converting Selected Conversion Items | 92 |
| 8.4 | Converting an Entire List | 93 |
| 9 | Understanding Conversion Results | 94 |
| 9.1 | Comparing Original and Modernized Source Members | 94 |
| 9.2 | The Conversion List Editor | 94 |
| 10 | Copybooks in ARCAD-Transformer RPG | 96 |

Troubleshooting97
F.A.Q.107

Tables

| | |
|--|----|
| Table 1: Contact ARCAD Software | 3 |
| Table 2: Contact the ARCAD Software support team | 3 |
| Table 3: ARCAD-Transformer RPG User Guide Publication Record | 5 |
| Table 4: Related Documentation | 6 |
| Table 5: Summary of Conversion Operations | 22 |
| Table 6: Arithmetic Operations Conversion | 23 |
| Table 7: Array Operations Conversion | 23 |
| Table 8: Bit Operations Conversion | 24 |
| Table 9: Branching Operations Conversion | 25 |
| Table 10: Call Operations Conversion | 25 |
| Table 11: Compare Operations Conversion | 25 |
| Table 12: Conversion Operations Conversion | 26 |
| Table 13: Data-Area Operations Conversion | 27 |
| Table 14: Date Operations Conversion | 27 |
| Table 15: Declarative Operations Conversion | 28 |
| Table 16: Error-Handling Operations Conversion | 28 |
| Table 17: File Operations Conversion | 28 |
| Table 18: Indicator-Setting Operations Conversion | 29 |
| Table 19: Information Operations Conversion | 29 |
| Table 20: Initialization Operations Conversion | 30 |
| Table 21: Memory Management Operation Conversion | 30 |
| Table 22: Message Operation Conversion | 31 |
| Table 23: Move Operation Conversion | 31 |
| Table 24: Move Zone Operation Conversion | 32 |
| Table 25: String Operations Conversion | 32 |
| Table 26: Structured Programming Operations Conversion | 33 |
| Table 27: Subroutine Operation Conversion | 34 |
| Table 28: Test Operation Conversion | 35 |
| Table 29: XML Operation Conversion | 35 |
| Table 30: License Status | 37 |
| Table 31: Summary of Conversion Options | 40 |
| Table 32: Replace Existing Member (REPLACE) Parameters | 41 |
| Table 33: Convert Program calls (CVT_CALL) Parameters | 42 |
| Table 34: Convert GoTo (CVT_GOTO) Parameters | 44 |
| Table 35: Convert Key List (CVT_KLIST) Parameters | 49 |
| Table 36: Convert MOVEA (CVT_MOVEA) Parameters | 51 |
| Table 37: Convert Subr. to procedures (CVT_SUBR) Parameters | 54 |
| Table 38: Use %ParmNum (USEPARMNUM) Parameters | 55 |
| Table 39: Indentation Size (char) (INDENT) Parameters | 56 |
| Table 40: Indent Comments (INDENTCMT) Parameters | 57 |
| Table 41: Case for operation codes (OPCODECASE) Parameters | 57 |
| Table 42: Case for the B.i.F. (BLTFNCCASE) Parameters | 58 |
| Table 43: Case for special words (SPCWRDCASE) Parameters | 58 |

| | |
|---|----|
| Table 44: Case for key words (KEYWRDCASE) Parameters | 59 |
| Table 45: Convert to "Fully Free" (FULLYFREE) Parameters | 59 |
| Table 46: Max nbr of not "Free" blocks (MAXNOTFREE) Parameters | 60 |
| Table 47: First Column (Fully Free) (FIRSTCOL) Parameters | 61 |
| Table 48: Pre-compilation Clauses (PRECPL) Parameters | 61 |
| Table 49: Mark the conversion type (FLGCVTTYPE) Parameters | 61 |
| Table 50: Clean Temporary Cross-references (CLRXREF) Parameters | 62 |
| Table 51: Clean Modified Lines (CLRFRMCHG) Parameters | 63 |
| Table 52: The RPG to RPGLE Conversion Options | 63 |
| Table 53: Converted Source Member Properties | 80 |
| Table 54: Source File (SRCFILE) Library Parameters | 81 |
| Table 55: Source Type (SRCTYPE) Parameters | 82 |
| Table 56: Object Type (OBJTYPE) Parameters | 82 |
| Table 57: Convert calculation specs. (CVTCLCSPEC) Parameters | 83 |
| Table 58: Convert declaration specs. (CVTDCLSPEC) Parameters | 83 |
| Table 59: Destination Source File (TOSRCFILE) Parameters | 84 |
| Table 60: Member Name (TOSRCMBR) Parameters | 85 |
| Table 61: Source Line Date (SRCDATE) Parameters | 85 |
| Table 62: Authorize QTEMP in batch (BATCHQTEMP) Parameters | 87 |
| Table 63: The Conversion List Editor | 94 |

Figures

| | |
|---|-----|
| Figure 1: Example of a New Connection (IBM i) | 65 |
| Figure 2: Command Line in the Command Log view | 66 |
| Figure 3: Create Library (CRTLIB) dialog | 67 |
| Figure 4: Confirmation Library was created in Commands Log view | 67 |
| Figure 5: Create duplicate source object dialog | 69 |
| Figure 6: Confirmation Source was created in Commands Log view | 69 |
| Figure 7: Remove Empty Members | 70 |
| Figure 8: New Member Filter dialog page 1 | 71 |
| Figure 9: New Member Filter dialog page 2 | 71 |
| Figure 10: Add Library List Entry dialog | 72 |
| Figure 11: Add Library List to Session | 74 |
| Figure 12: Selecting Source Member from i Projects Navigator view | 78 |
| Figure 13: Selecting Source Member from Remote Systems view | 79 |
| Figure 14: Comparing original and modernized sources | 94 |
| Figure 15: Problem Occurred CPF9801 | 97 |
| Figure 16: Problem Occurred MSG3542 | 98 |
| Figure 17: Problem Occurred MSG3579 | 99 |
| Figure 18: Problem Occurred MSG3866 | 100 |
| Figure 19: Display Error Log view | 101 |
| Figure 20: Error Log entries | 101 |
| Figure 21: Spooled Files View | 102 |
| Figure 22: My spooled files | 102 |
| Figure 23: Spooled File Filter | 102 |
| Figure 24: New Spooled File Filter 1 | 103 |
| Figure 25: New Spooled File Filter 2 | 103 |
| Figure 26: Show the spooled file contents | 103 |
| Figure 27: Spooled File Contents | 104 |
| Figure 28: Server Compatibility Error | 105 |
| Figure 29: Editing the compliant.xml | 106 |

1 About ARCAD-Transformer RPG

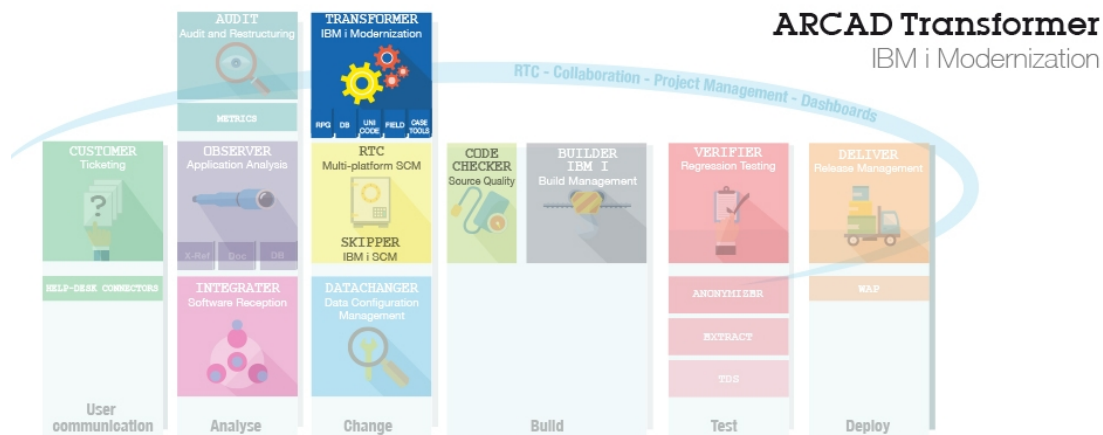
Prepare your RPG code for the new generation

ARCAD-Transformer RPG (aka IBM Rational ARCAD-Converter) accelerates the conversion of your application to free-format coding. It is an optional module offered as an integral part of the ARCAD Pack for Rational. ARCAD-Transformer RPG can also be purchased as an RDi Plug-In.

Reference For more information about ARCAD-Transformer RPG and to request a free trial, visit our website:

[Arcadsoftware.com/resource-items/arcad-transformer-rpg/](https://arcadsoftware.com/resource-items/arcad-transformer-rpg/)

For more information about obtaining permanent licenses, Contact ARCAD Software.



1.1 Business Context

RPG IV has evolved into a modern business language, supporting procedures, data areas, data structures, additional data types and extended file support. Greater interoperability is offered between RPG and Java, XML and SQL. Also, RPG source code is far more readable thanks to Free format, blank lines, and comments.

Free format programs have the same source type and are compiled in exactly the same way as fixed format RPG. The IBM RPG compiler allows the two styles to be mixed freely.

Experienced RPG developers can become proficient in RPG Free Format with just a few days of learning. Free Format brings not only the personal satisfaction of learning a new technology, but also modern language skills that can enhance your IT career for the future.

1.2 Functional Constraints

As a general rule, to achieve maximum conversion (that is, of C specifications, and also H, F, D, P specifications), IBM i V7R1 with Technology Refresh 7 is required. However, if you do not yet have Technology Refresh 7, but you have IBM i V7R1 (or V6R1 or V5R4), conversion with ARCAD-Transformer RPG is still possible, but only C specifications will be converted.

As part of Technology Refresh 7, a standalone PTF (number SI51094) for the RPG compiler is now available from IBM. This PTF enables compilation of full Free Format RPG and also maximum conversion with ARCAD-Transformer RPG.

Note: If you specifically need to compile any SQLRPGLE sources that have been converted to Free Format, you will then require the additional DB2 PTF group SF99701 level 26 (HyperPTF SF99701 in V7R1).

ARCAD-Transformer RPG is "DBCS capable": it is written 100% in Unicode and supports all CCSID sources, including Japanese, Chinese, Korean.

Sources cannot be processed by the `ACVTRPGFRE` command if:

- they are (SQL)RPGLE source stored in IFS files (sources must be stored in a ***FILE** source file).
- the `COPY` clause contains C-specifications (their conversion is possible if they only contain declaration specifications).

For the lines surrounded by conditional compilation directives (`/IF ... /ENDIF`), the conversion is not guaranteed when the conditions are not effective during the compilation.

2 About the Conversion Process

Chapter Summary

| | |
|--|----|
| 2.1 The Four Conversion Processes..... | 15 |
| 2.2 The Conversion Engine..... | 16 |

ARCAD-Transformer RPG provides many different choices of source code conversion. No one method is better than another. It is important to choose a standard conversion method for your company and to understand all of the options and their impact on the resulting modernized source code.

As part of the modernization process, a complete diagnostic compile listing is generated for each program being converted. This listing contains a field cross reference which ARCAD-Transformer RPG relies on in order to successfully convert the source code. This cross reference is an important component in the modernization process. In addition, the initial source code must be completely compilable. Without this, a new modernized source member will not be created.

Note: A diagnostic listing will not create a new program object and it will not replace any existing program objects.

2.1 The Four Conversion Processes

There are four different steps in the conversion process.

Note: A member is an RPG source member if its source type is RPG, RPG38, RPT, RPT38 or SQLRPG.

2.1.1 The Unitary Conversion Process

This process is executed to convert one source member selected from the **i Projects Navigator** or **Remote Systems** view.

- Check for Conversion Engine.
- IF the Conversion Engine is installed THEN check the availability of conversion units.
 - IF enough conversion units remain THEN execute [The Conversion Process](#).
 - ELSE stop the process.
 - END.
- ELSE stop the process.
- END.

2.1.2 The Conversion Process

- IF the source member is an RPG Source Member THEN execute the [The RPG to RPGLE](#)

Conversion Process.

- IF the conversion failed THEN stop the process.
- END
- END
- Execute the [Free Form Conversion Process](#).
 - IF the conversion failed THEN stop the process.
- END

2.1.3 The RPG to RPGLE Conversion Process

- Execute the standard `CVTRPGSRC` command using the preferences defined by the user.
- IF the conversion succeeds THEN use the newly converted RPGLE source member as the input to the Free Form conversion.
- ELSE IF the conversion failed because the target Source File does not exist THEN
 - IF the related preference has been checked THEN create the Target Source File and restart the conversion (this operation is executed only once).
 - END
- ELSE IF the conversion failed because the Target Source Member already exists THEN
 - IF the related preference has been checked THEN remove the Target Source Member and restart the conversion (this operation is executed only once).
 - END
- END

2.1.4 Free Form Conversion Process

- Select the parent library of the `ACVTRPGFRE` command according to the preference.
- Execute the `ACVTRPGFRE` command.

2.2 The Conversion Engine

The transformation of RPGLE to a more modern syntax may be inconvenient, but could also be seen as an advantage.

In calculations and numerical assignments with Free syntax, it is no longer possible to have a result value larger than the capacity of the result variable, nor to have certain invalid values.

Where an old operation allowed the truncation of the result value (whether intended or not by the programmer!), the same operation performed with Free syntax generates an error. We can also consider that this is a way to clean up code, because often these were "hidden bugs".

Reference For more information about installing and maintaining the Conversion Engine, refer to the *ARCAD-Transformer RPG Installation Guide*.

Example 1: Numeric Value Too Large

Before:

```

C      Move(p)  1234      WVar04      defined P(4,0)
C      Z-add    WVar04    WVar03      defined P(3,0)
      (result was 234 in WVar03)

```

After:

```

WVar04 = 1234;
WVar03 = WVar04;

```

(error at execution: MCH1210 Receiver value too small to hold result.)

Example 2: Invalid Numeric Value

Before:

```

C      Move      '45R'    WVarA3      defined A(3)
C      Move      WVarA3    WVar03      defined P(3,0)
      (result was -459 in WVar03)

```

After:

```

WVarA3 = '45R';
WVar03 = %Dec(%XLate(' ': '0':
                  WVarA3):3:0);

```

(error at execution: RNX0105: A character representation of a numeric value is in error.)

2.2.1 About the ACVTRPGFRE Command

The `ACVTRPGFRE` command is intended to be launched either natively or from RDi to prepare and/or perform the conversion of (SQL)RPGLE source to Free syntax.

This command, as a module of ARCAD-Transformer RPG, is licensed according to the number of conversions made (see [Managing Activation Keys on page 36](#)).

Almost all of the H, F, D, P and C specifications can be converted to Free syntax. Regarding the declaration specifications (H, F, D, P), the conversion does not require the source to be compilable independently. The goal is to achieve the same for the calculation specifications (C). However, to do so, it is important that members be written in or already converted to RPG III (RPG/400) or higher before carrying out the RPG Free transformation.

The purpose of this command is to convert ILE RPG to RPG Free. To do so, it carries out two separate actions:

1. the IBM command `CVTRPGSRC`, which converts any RPG III or IV to ILE RPG;
2. the ARCAD command `ACVTRPGFRE`, which converts any ILE RPG to RPG Free.

Important! It is recommended to redesign any RPG II to RPG III or higher. If not, the modernization provided by ARCAD-Transformer RPG will not be possible.

It's easy to convert certain opcodes to Free syntax (e.g. EVAL, ADD, etc.), but others are much more difficult (e.g. MOVE, MOVEL, etc.). In fact, the instruction to generate very often depends on the type, length and dimensions of the fields used for factor 1, factor 2 or the result. This is why the command starts with a complete cross-reference (X-Ref) at the field level so it knows the characteristics of every field in the program.

Important! This X-Ref calculation is based on a compilation (without creating the resulting object) which **must complete successfully**. It is therefore necessary to have libraries containing the sources and files used by the program online (see [Preparing the ARCAD-Transformer RPG Environment on page 64](#)).

Sometimes, compilation attributes are required or it may be necessary to execute commands prior to compilation (especially OVRDBF). In this case, it is recommended that you put these attributes and pre-compilation commands in the source to be analyzed as pre-compilation clause(s).

Note: All field declarations made in C-specifications are moved to D-specifications, unless the field is already declared elsewhere, in a file for example.

Because they can be distracting in the LPEX source editor, any special color attribute characters encountered in the comments in the changed lines are replaced by blanks.

If you specify a source output member, it will contain the new converted source, but you cannot do the conversion of a source member to itself.

Note: The resulting source is normally compilable, but you need to check the use of **%Found** and **%Equal** with the SCAN, CHECK, CHECKR and LOOKUP operation codes (see [About Operation Codes below](#)).

You can also choose to only store the proposals for adding, modifying and deleting lines in the Source Modification file **AARFCHSF1**, keyed on CHS_CAPP = '*NONE', CHS_CENV = '*NONE', CHS_CVER = job number, CHS_JOB = object name, CHS_CTTYPE = RPGLE or SQLRPGLE.

2.2.2 About Operation Codes

All RPG syntax revolves around operation codes. RPG is broken into declaratives and calculation instructions. Declaratives enable you to define: variables; access to files; the input/output parameters; and external prototypes (external programs called inside RPG programs). Calculation instructions are "managed" by an operation code word, with all its factors. The list contains strictly defined operation codes - each line in the list defines one opcode.

2.2.2.1 Opcodes that are Never Transformed

Only the following opcodes (rarely used) remain in classic syntax:

- MHHZO
- MHLZO
- MLHZO
- MLLZO

2.2.2.2 Opcodes that are only Occasionally Transformed

In most cases, the use of these opcodes can be converted, except in certain specific cases where there was no acceptable equivalent found in Free syntax:

- TIME: when the result field is defined with a length of 14.
- SCAN, CHECK, CHECKR: when the result field is an array.
- BITON, BITOFF: when factor 2 is a named-constant.
- POST: when the result field (DS name) is used.
- MOVE, MOVEL: when factor 2 or result field is a variable length field.
- CALL, PARM in some rare cases (see parameter [Convert Program calls \(CVT_CALL\)](#)).
- GOTO, TAG in some rare cases (see parameter [Convert GoTo \(CVT_GOTO\)](#)).
- KLIST, KFLD in some rare cases (see parameter [Convert Key List \(CVT_KLIST\)](#)).
- MOVEA in some rare cases (see parameter [Convert MOVEA \(CVT_MOVEA\)](#)).

2.2.2.3 Opcodes with Error Management Using (e)

Many conventional operations codes allow use of the "(e)" extension to monitor execution errors, often followed by an %Error test. For many of these opcodes (e.g. CHAIN), there is the same possibility in Free with the same operation code.

But for the following opcodes, the Free syntax uses the EVAL operation which does not allow this extension:

- CHECK, CHECKR, SCAN
- OCCUR
- XLATE, SUBST
- ALLOC, REALLOC
- ADDDUR, SUBDUR, EXTRCT

In order to have the equivalent function in Free syntax, MONITOR provides the error handling, but a "ruse" has been used to successfully turn on/off the %Error indicator:

1. Two fields, **TTimeOk** and **TTimeError**, are created in the source.
2. The TEST instruction is used on these 2 fields to obtain the desired value for the %Error indicator.

Two comments containing ***CVTWNG** are also placed in the source to explain this process.

Example: Use of a Monitored ALLOC(e)

Before:

```

C      Alloc(e)  1000000      ptr01
C      If        %Error
C      ...

```

After:

```

// *CVTWNG : set %Error to '0'
  Test(e) TTimeOk;
  Monitor;
    ptr01 = %Alloc(1000000);
    On-Error;
// *CVTWNG : set %Error to '1'
  Test(et) *hms0 TTimeError;
  EndMon;
  If %Error;
  ...

```

2.2.2.4 Opcodes that manage %Found or %Equal

After execution, some conventional operation codes set the %Found or %Equal, indicators that can then be tested.

For many of these opcodes (e.g. SETLL), there is the same possibility in Free syntax with the same opcode, but for the following opcodes, the Free syntax uses EVAL with a BIF %xxxxx(...) which no longer supports these two special indicators.

- CHECK, CHECKR (%Found)
- SCAN (%Found)
- LOOKUP (%Found & %Equal)

The management of the two indicators %Found and/or %Equal is no longer provided in the generated code. A comment containing ***CVTWNG** is added prompting you to verify if these two indicators are subsequently tested and to make any necessary corrections.

This comment is only added to your source if these indicators are used at least once in the program.

Example: SCAN, Followed by a %Found test

Before:

```

C      'B'      Scan      C01
C      If        %Found
C      ...

```

After:

```

If %Scan('B' : C01) > 0;
  // *CVTWNG : %Found is not updated by %Scan
EndIf;

```

```
If %Found;  
...
```

2.2.3 About the Conversion Operations

All of the opcodes included in the following tables were taken using the categories in the RPGLE reference guide. Each one has an explanation of the conversion used in Free.

| Conversion Operations | |
|---------------------------|-----------------------------------|
| Arithmetic Operations | Indicator-Setting Operations |
| Array Operations | Information Operations |
| Bit Operations | Initialization Operations |
| Branching Operations | Memory Management Operations |
| Call Operations | Message Operations |
| Compare Operations | Move Operations |
| Conversion Operations | Move Zone Operations |
| Data-Area Operations | String Operations |
| Date Operations | Structured Programming Operations |
| Declarative Operations | Subroutine Operations |
| Error-Handling Operations | Test Operations |
| File Operations | XML Operations |

Table 5: Summary of Conversion Operations

Arithmetic Operations

If specified in positions 71-76, indicators are set after the converted instruction.

| Operation Code | Free version |
|----------------|---|
| ADD | Simple evaluation of the result value, with the + operator |
| DIV | Simple evaluation of the result value, with the / operator |
| MULT | <p>Simple evaluation of the result value, with the * operator</p> <div style="border: 1px dashed gray; padding: 10px; margin: 10px 0;"> <p>Note: Use of the MULT operation code with one of the following values in factor 1 or 2: 100.0001, 10000.01 or 10000.0001.</p> <p>If the variables on the line have a length of 6 (or 8 for the last case), it is assumed be a method/technique used to invert the format of a date between MmDdYy and YyMmDd (or MmDdYyyy and YyyyMmDd in the last case).</p> <p>For these cases, an equivalent process is done, with Eval, %Dec, %Subst and %EditC.</p> </div> |
| MVR | <p>Evaluation with %Rem(...), but placed before the previous DIV instruction.</p> <div style="border: 1px dashed gray; padding: 10px; margin: 10px 0;"> <p>Note: There is a risk if the result variables of the DIV or MVR are also the DIV operands.</p> </div> |
| SQRT | Simple evaluation of the result value, with %Sqrt(...) |
| SUB | Simple evaluation of the result value, with the - operator |
| Z-ADD | Simple numeric assignment |
| Z-SUB | Simple numeric assignment with negative value |

Table 6: Arithmetic Operations Conversion

Array Operations

| Operation Code | Free version |
|----------------|--|
| LOOKUP | Evaluation using %LookUp, %LookUpXX (or %TLookUp, %TlookUpXX) and set result indicator *INxx if required. In |

Table 7: Array Operations Conversion

| Operation Code | Free version |
|----------------|--|
| | the case of an array with a variable index, the index is set to 1 for a failed lookup. <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> Note: Addition of a comment warning: *CVTWNG : %Equal & %Found are not updated by %LookupXX </div> |
| MOVEA | <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> Reference See parameter Convert MOVEA (CVT_MOVEA). </div> |
| SORTA | Simple conversion to Free syntax for an opcode already using extended factor 2. |
| XFOOT | Evaluation of %XFoot. <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> Note: If specified in positions 71-76, indicators are set afterwards. </div> |

Table 7: Array Operations Conversion

Bit Operations

| Operation Code | Free version |
|----------------|---|
| BITOFF | Evaluation using a combination of %BitAnd and %BitNot. <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> Note: No conversion if factor 2 is a named constant. </div> |
| BITON | Evaluation using a combination of %BitOr. <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> Note: No conversion if factor 2 is a named constant. </div> |
| TESTB | Separate evaluation for each resulting indicator by comparing with a hex value (sometimes with %BitAnd). |

Table 8: Bit Operations Conversion

Branching Operations

| Operation Code | Free version |
|------------------|--|
| CABxx, GOTO, TAG | <i>Reference</i> See parameter Convert GoTo (CVT_GOTO) . |
| ITER, LEAVE | Same opcode without parameters in Free syntax. <i>Note:</i> This can sometimes be replaced by "ATag = '*LEAVE' or '*ITER'", for the management of GOTO (backwards) transformed to structured programming. |
| LEAVESR | Same opcode without parameters in Free syntax. |

Table 9: Branching Operations Conversion

Call Operations

| Operation Code | Free version |
|--------------------------|---|
| CALL, CALLB, PARM, PLIST | <i>Reference</i> See parameter Convert Program calls (CVT_CALL) . |
| CALLP, RETURN | Simple conversion to Free syntax for opcodes already using extended factor 2. |

Table 10: Call Operations Conversion

Compare Operations

| Operation Code | Free version |
|----------------|--|
| ANDxx, ORxx | Addition of "AND" or "OR" then a test of factor 1 and 2, without terminating the instruction if followed by another ANDxx or ORxx. |
| CABxx | If there are indicators, first test factor 1 and factor 2, then manage branch condition. |

Table 11: Compare Operations Conversion

| Operation Code | Free version |
|----------------------------|--|
| | <i>Reference</i> See parameter <code>Convert GoTo (CVT_GOTO)</code> . |
| CASxx | <p>If there are indicators, first test factor 1 and factor 2, then manage subroutine execution according to the condition.</p> <p>Note: However, the conversion of the CASxx, CAS, ENDCS group has 2 possibilities:</p> <ul style="list-style-type: none"> • If at least one of the CASxx statements has an indicator in positions 71-76, it will be converted to "If ...", "Exsr ...", "EndIf". • If none of the CASxx statements has an indicator in positions 71-76, this will be structured better: "Select", "When ...", "Exsr ...", ... "EndSI" |
| COMP | Setting of each indicator, comparing factor 1 and factor 2. |
| DOU, DOW, IF, WHEN | Simple conversion to Free syntax for opcodes already using extended factor 2. |
| DOUxx, DOWxx, IFxx, WHENxx | Use of the opcodes "DoU", "DoW", "If" or "When" with a test of both factor 1 and 2, without terminating the instruction if followed by ANDxx or ORxx. |

Table 11: Compare Operations Conversion

Conversion Operations

| Operation Code | Free version |
|----------------|--|
| MOVE, MOVEL | <i>Reference</i> See Move Operations . |

Table 12: Conversion Operations Conversion

Data-Area Operations

| Operation Code | Free version |
|-----------------|--|
| IN, OUT, UNLOCK | Equivalent code in Free, with additional management of the error indicator if specified in position 73-74. |

Table 13: Data-Area Operations Conversion

Date Operations

| Operation Code | Free version |
|----------------|--|
| ADDDUR | <p>Evaluation of the new Date, Time, TimeStamp fields using %Years, %Months, ... %Hours, etc.</p> <p>Note: Management of operation code extender "(e)" with "Monitor/On-Error/EndMon".</p> |
| EXTRCT | <p>Evaluation of Date, Time, TimeStamp fields using %Subdt; conversion to alpha with %EditC if necessary.</p> <p>Note: Management of operation code extender "(e)" with "Monitor/On-Error/EndMon".</p> |
| MOVE, MOVE1 | <p>When factor 2 and/or the result field contain a Date/Time/TimeStamp field, conversion using %Date, %Time, %TimeStamp, or %Char, %Dec using the date/time format of the field.</p> <p>Reference See Move Operations.</p> |
| SUBDUR | <p>Evaluation according to the operation and the field types for factors 1 and 2 and the result field:</p> <ul style="list-style-type: none"> • either using %Diff • or by subtracting a time/duration using %Years, %Months, ... %Hours, etc. <p>Note: Management of operation code extender "(e)" with "Monitor/On-Error/EndMon".</p> |

Table 14: Date Operations Conversion

Declarative Operations

| Operation Code | Free version |
|----------------|--|
| *DTAARA DEFINE | Transfer of the associated field declaration to a *DTAARA in D-specs, adding the DTAARA(xxxx) keyword. <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> Note: Not done if this declaration is in a COPY clause. </div> |
| KFLD, KLIST | Reference See parameter Convert Key List (CVT_KLIST) . |
| *LIKE DEFINE | Transfer of the field declaration to D-specs. |
| PARM, PLIST | Reference See parameter Convert Program calls (CVT_CALL) . |
| TAG | Reference See parameter Convert GoTo (CVT_GOTO) . |

Table 15: Declarative Operations Conversion

Error-Handling Operations

| Operation Code | Free version |
|---------------------------|---|
| MONITOR, ON-ERROR, ENDMON | Simple conversion to Free syntax for opcodes already using extended factor 2. |

Table 16: Error-Handling Operations Conversion

File Operations

| Operation Code | Free version |
|--|---|
| ACQ, CHAIN, CLOSE, COMMIT, DELETE, EXCEPT, | The equivalent operation code exists in Free for all these file operations with 0, 1 or more parameters. Followed possibly by setting indicators *INxx, if they were specified in positions 71-76, testing the %Error, %Found or %Equal indicators. |

Table 17: File Operations Conversion

| Operation Code | Free version |
|--|---|
| EXFMT, FEOD, FORCE, NEXT, OPEN, READ, READC, READE, READPE, REL, ROLBK, SETGT, SETLL, UNLOCK, UPDATE, WRITE | <div style="border: 1px dashed gray; padding: 5px; display: inline-block;"> Reference See the processing of key lists (Convert Key List (CVT_KLIST)). </div> |
| POST | As above, but not converted when the result field is used - no equivalent in Free syntax. |

Table 17: File Operations Conversion

Indicator-Setting Operations

| Operation Code | Free version |
|----------------|--|
| SETOFF, SETON | Evaluation to '0' or '1' for 1, 2 or 3 indicators, successively. |

Table 18: Indicator-Setting Operations Conversion

Information Operations

| Operation Code | Free version |
|----------------|---|
| DUMP | Equivalent opcode in Free |
| SHTDN | Indicator evaluation *INxx with %ShtDn |
| TIME | Four different cases, according to the type/length of the result field: <ol style="list-style-type: none"> 1. Field type Date/Time/TimeStamp : Evaluation with %Date, %Time, %TimeStamp 2. Numeric field of 6,0 : Evaluation with %Dec(%Time) |

Table 19: Information Operations Conversion

| Operation Code | Free version |
|----------------|---|
| | <ol style="list-style-type: none"> 3. Numeric field of 12,0 : Evaluation with %Dec(%Time) * 1000000 + %Dec(%Date:*JOB RUN) (This would not be OK if the format of the job date was *JUL, but this is without doubt never used). 4. Numeric field of 14,0 : Stays in traditional format, because it is impossible to have a format with the correct mapping of the century according to the format of the job. |

Table 19: Information Operations Conversion

Initialization Operations

| Operation Code | Free version |
|----------------|--|
| CLEAR, RESET | Equivalent opcode that exists in Free, with 1 or 2 parameters. |

Table 20: Initialization Operations Conversion

Memory Management Operations

| Operation Code | Free version |
|----------------|--|
| ALLOC, REALLOC | Evaluation of a pointer with %Alloc or %ReAlloc. <div style="border: 1px dashed gray; padding: 5px; margin: 10px 0;"> Note: Management of the operation code extender "(e)" (or an error indicator in position 73-74) with "Monitor/On-Error/EndMon". </div> |
| DEALLOC | Equivalent opcode that exists in Free, with 1 parameter; then setting of indicator *INxx, if specified as the error indicator. |

Table 21: Memory Management Operation Conversion

Message Operations

| Operation Code | Free version |
|----------------|---|
| DSPLY | Equivalent opcode that exists in Free, with 1, 2 or 3 parameters. |

Table 22: Message Operation Conversion

Move Operations

| Operation Code | Free version |
|----------------|---|
| MOVE, MOVEL | <p>This operation code, widely used in traditional syntax, performs operations for which the behavior depends on the type of the variables and their length; all of the following cases are covered:</p> <p>Figurative constant in factor 2 (*Blank, *Zero, *Hival, *Loval, *ALL'o', *ALL'xxxx').</p> <ul style="list-style-type: none"> • With or without (p) as operation extender • Variable or fixed length field • Factor 2 field with a length less than the result field • Factor 2 field with a length greater than or equal to the result field • Assignment with numeric conversion <-> alpha using %XLate, %Dec, %EditC, and possibly digital shifting by multiplying or dividing by 10, 100, 1000 etc.. • When factor 2 and/or the result field contain a Date/Time/TimeStamp field, conversion using %Date, %Time, %TimeStamp, or %Char, %Dec using the date/time format of the field. <p>If specified in positions 71-76, indicators are set after the converted instruction.</p> <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> <p>Note: The following case is not converted:</p> <ul style="list-style-type: none"> • Assignment between a date/time or numeric field and a variable-length field. </div> |
| MOVEA | <div style="border: 1px dashed gray; padding: 5px;"> <p>Reference See parameter Convert MOVEA (CVT_MOVEA).</p> </div> |

Table 23: Move Operation Conversion

Move Zone Operations

| Operation Code | Free version |
|----------------------------|---|
| MHHZO, MHLZO, MLHZO, MLLZO | These opcodes (rarely used) have no equivalent in Free syntax - they always stay in traditional syntax. |

Table 24: Move Zone Operation Conversion

String Operations

For CHECK, CHECKR, SUBST, XLATE, management of the operation code extender "(e)" (or an error indicator in position 73-74) with "Monitor/On-Error/EndMon".

| Operation Code | Free version |
|----------------|---|
| CAT | <p>About 20 different cases (fairly simple) generating a block of 1 to 20 lines, related to the following variations:</p> <ul style="list-style-type: none"> • with or without "(p)" as operation extender • Number of blanks not specified in factor 2 • Number of blanks is 0 in factor 2 (CAT Var:0) • Number of blanks between 0 and 5 in factor 2 (CAT Var:2) • Number of blanks greater than 5 or variable in factor 2 (..CAT Var:Var2..) <p>In certain cases, a global variable "NCatLen" is added to allow correct management of concatenations.</p> |
| CHECK, CHECKR | <p>Use of %Check or %CheckR.</p> <p>Addition of a comment warning: "// *CVTWNG : %Found is not updated by %Check(r)".</p> <p>Not converted when the result field is an array.</p> |
| SCAN | <p>Use of %Scan.</p> <p>Addition of a comment warning: "// *CVTWNG : %Found is not updated by %Scan".</p> |

Table 25: String Operations Conversion

| Operation Code | Free version |
|----------------|---|
| | Not converted when the result field is an array. |
| SUBST | Evaluation using %Subst with, if necessary, %Subst for the result variable if there is no operation extender "(p)". |
| XLATE | Evaluation using %XLate with, if necessary, %Subst for the result variable if there is no operation extender "(p)". |

Table 25: String Operations Conversion

Structured Programming Operations

| Operation Code | Free version |
|-----------------------------|--|
| ANDxx, ORxx | Addition of "AND" or "OR" then a test of factor 1 and 2, without terminating the instruction if followed by another ANDxx or ORxx. |
| DO | <p>Several possible cases:</p> <ol style="list-style-type: none"> "DO " or "DO 1" are replaced by "DoU '1'". "DO *HIVAL" is replaced by "DoW '1'". Other cases of DO with result field: replaced by "For ...", with the result variable as an index. Other cases of DO without Result field: replaced by "For NForIdxNNNN = ...", where NForIdxNNNN is a local variable created (with NNNN = 0001, 0002, ...) for each loop managed. <p>If an index is specified on the corresponding ENDDO (or END) statement, then this is set to "By xxxx", following the "For ..."</p> |
| DOU, DOW, IF, FOR, WHEN | Simple conversion to Free syntax for opcodes already using extended factor 2. |
| DOUxx, DOWxx, IFxx, WHENxx | Use of the opcodes "DoU", "DoW", "If" or "When" with a test of both factor 1 and 2, without terminating the instruction if followed by ANDxx or ORxx. |
| ELSE, ELSEIF, SELECT, OTHER | Same opcode exists in Free syntax. |

Table 26: Structured Programming Operations Conversion

| Operation Code | Free version |
|----------------|---|
| END, ENDxx | The opcode END is always replaced by "EndIf", "EndDo", "EndFor", "EndMon" or "EndSI" according to the instruction at the start of the group to which it relates. The opcode ENDDO may become "EndFor". |
| ITER, LEAVE | Same opcode without parameters in Free syntax. This can sometimes be replaced by "ATag = '*LEAVE' or '*ITER'", for the management of GOTO (backwards) passed to structured programming. |

Table 26: Structured Programming Operations Conversion

Subroutine Operations

| Operation Code | Free version |
|-----------------------------|---|
| BEGSR, ENDSR, EXSR, LEAVESR | Same opcode in Free syntax. <i>Reference</i> For ENDSR with a label in factor 1, see parameter Convert GoTo (CVT_GOTO) . To convert the subroutines to ILE procedures, see parameter Convert Subr. to procedures (CVT_SUBR) . |
| CASxx, CAS, ENDCS | If there are indicators, first test factor 1 and factor 2, then manage subroutine execution according the condition. However, the conversion of the CASxx, CAS, ENDCS group has 2 possibilities: <ul style="list-style-type: none"> • If at least one of the CASxx statements has an indicator in positions 71-76, it will be converted to "If ...", "Exsr ...", "EndIf". • If none of the CASxx statements has an indicator in positions 71-76, this will be structured better: "Select", "When ...", "Exsr ...", ... "EndSI" |

Table 27: Subroutine Operation Conversion

Test Operations

| Operation Code | Free version |
|----------------|---|
| TEST | Use of the same opcode in Free syntax : "Test(e)...". Then setting of indicator *INxx, if specified in position 73-74. |
| TESTB | Evaluation of each resulting indicator, by comparing with a hex value (sometimes with %BitAnd). |
| TESTN | Evaluation of indicator(s) specified in positions 71-76, more or less complex: with %Checkr('0123456789'...), %Check (X'CoDoFo') or %BitAnd, ... in order to ensure a similar behavior. A comment in the form "// Test xxx : numeric or blank ?" is also added to explain each test. |
| TESTZ | Evaluation of indicator(s) specified in positions 71-76, more or less complex: with %BitAnd, and comparison to allowed hex values (X'Co', X'Do', X'5o', X'6o). |

Table 28: Test Operation Conversion

XML Operations

| Operation Code | Free version |
|-------------------|-----------------------------|
| XML-SAX, XML-INTO | Same opcode in Free syntax. |

Table 29: XML Operation Conversion

3 Configuring ARCAD-Transformer RPG

Chapter Summary

| | |
|---|----|
| 3.1 Managing Activation Keys..... | 36 |
| 3.2 Transferring Keys to Different Machines..... | 38 |
| 3.3 Changing the Language..... | 39 |
| 3.4 Managing the Free Form Conversion Options..... | 40 |
| 3.5 Managing the Default RPG to RPGLE Conversion Options..... | 63 |

Before using ARCAD-Transformer RPG it is necessary to configure the parameters and register your activation key. This chapter describes the different parameters found in ARCAD-Transformer RPG preferences as well as how to manage the product's licenses.

Reference For more information about installing and upgrading the product, refer to the *ARCAD-Transformer RPG Installation Guide*.

Follow the subsequent steps to access the various preference pages.

Step 1 Open the RDi **Preferences** (*Window > Preferences*).

Step 2 Expand the ARCAD-Transformer RPG category.

3.1 Managing Activation Keys

Important The Conversion Engine must be installed before registering an Activation Key.

The number of transformations you can make using ARCAD-Transformer RPG is based on Conversion Units. One unit is consumed each time a source member is successfully converted, regardless of its size and regardless of how many times it has already been converted.

Example

Converting one source costs one Conversion Unit. Each successive conversion of the same source also costs one Conversion Unit.

The Conversion Engine counts Conversion Units based on your license. You can call the Conversion Engine until all the available units in your license are consumed. To continue using the engine after all of your units are consumed, request and activate a new license for ARCAD-Transformer RPG. Licenses are managed by activation keys.

Temporary Activation Keys

Temporary activation keys for ARCAD-Transformer RPG enable you to evaluate the product for free by activating 10 Conversion Units. The temporary ARCAD-Transformer RPG activation key is sent to you by email, following a successful [download from the website](#). This temporary key is valid on the

machine specified (defined by its serial number and LPAR number) for 15 days following the download. It is recommended to activate the key during this period. If you do not activate the key in this period, please contact ARCAD support or the sales team who can re-generate the temporary key for you, granting you an overall total of 10 conversions.

Once the temporary key is active, it is valid for one year. Please note that successive conversions of the same source member do consume conversion units. For a given serial number and LPAR, a used temporary key cannot be extended or re-issued. If you wish to continue to use the product after the 10 free conversions are made, please contact your local ARCAD sales team to purchase a permanent license.

Permanent Activation Keys

Once purchased, a permanent activation key allows you to use the Conversion Units you have paid for, with no expiration date.

To order a permanent key (a license for the product), please [Contact ARCAD Software](#).

You can also contact your local IBM representative with the following product ID: 5725-L13 and part number: D12EWLL.

3.1.1 Activating Keys

Activation keys are managed from the plug-in's **Activation Key** preference page (*Window > Preferences > ARCAD-Transformer RPG > Activation Key*).

ARCAD-Transformer RPG can be installed on several different servers. Select the IBM i connection related to the server you want to manage.

Follow the subsequent steps to select an IBM i connection.

Step 1 Click the  browse icon to the right of the **Connection** field.

Step 2 Select the appropriate connection in the **Available IBM i Connections** dialog.

Step 3 Click **OK**.

Result The connection is automatic and if successful, the **License Status** displays.

| | |
|--------------------------|---|
| Permanent Activation Key | The current, active license is permanent. |
| Temporary Activation Key | The current, active license is temporary. |
| No Valid Activation Key | No active license is found. |

Table 30: License Status

3.1.1.1 Viewing Activation Key Details

Before registering (activating) a key, you can check its contents to be sure it is really what you want to activate. Follow the subsequent steps to view activation key information for ARCAD-Transformer

RPG.

Step 1 Enter the key you wish to check in the **Activation Key** field.

Step 2 Click **Key Information**.

Result All information concerning the key is displayed, including the total number of conversion units in the license, the number used and the number of conversions you can still make.

3.1.1.2 Registering an Activation Key

A key must be registered (activated) for your product before you can use it. Follow the subsequent steps to register an activation key.

Step 1 Enter the key you wish to activate in the **Activation Key** field

Step 2 Click **Activate**.

Result All information concerning the key is displayed, including the total number of conversion units in the license, the number used and the number of conversions you can still make.

3.2 Transferring Keys to Different Machines

You can transfer your ARCAD-Transformer RPG license between two machines in order to use the remaining Conversion Units available on a license key on a second IBM i system. Transferring your license key involves contacting your (ARCAD) Software Provider and entering the information shared with you in the **Preferences** menu.

The idea is to disable a license on one machine ("hold" it) in order to receive a new Activation Key for another machine.

Follow the subsequent steps to hold your current key for an IBM i connection and make the associated Conversion Units available to use on a different machine.

Step 1 Contact your (ARCAD) Software Provider. Ensure that the person you are contacting is able to generate Hold Keys, like the support team.

In your message, give details concerning your request and provide your current machine's serial number and LPAR.

Result Your provider will send you the Hold Key that corresponds to your request.

Step 2 Open the **Hold ARCAD Transformer RPG Key** preference page (*Window > Preferences > ARCAD-Transformer RPG*).

Step 3 Connect to the current IBM i for which you want to deactivate the license in the Preferences menu.

Click the Browse icon to the right of the **Connection** field. Select the appropriate connection in the **Available IBM i Connections** dialog.

Click **OK**.

Step 4 Enter the Hold Key that you received in Step 1 in the first area.

Step 5 Click the **Hold** button.

Result A special Deactivation Key is generated in the second area which includes the number of Conversion Units still available in the initial license.

Step 6 Send the Deactivation Key to your (ARCAD) Software Provider.

This new key *must* be sent back to your Software Provider. They will use this Deactivation Key to analyze the number of Conversion Units your license has and create a new Activation Key for your second system that contains the same amount of units.

Result Your provider will send you the Activation Key that corresponds to your request.

Step 7 Register the new Activation Key for the new system (connection).

Reference For more information about registering keys, refer to [Managing Activation Keys on page 36](#).

3.3 Changing the Language

ARCAD-Transformer RPG is available in three languages: English, French and German.

To manage the language in which the messages and command prompts for ARCAD-Transformer RPG are displayed, open the **Change Language for IBM i product** preference page (*Window > Preferences > ARCAD-Transformer RPG*).

Note: Only one language can be selected at a time and the option is only available when ARCAD-Transformer RPG is used as a standalone product (Library ARCAD_RPG). You cannot change the language when the entire Arcad product is installed.

Some messages or texts may not be translated and therefore are only displayed in English.

Step 1 To change the language for your system, select an IBM i Connection.

The product can be installed on several different servers. Select the IBM i server to manage.

1. Click the Browse icon to the right of the **Connection** field.
2. Select the appropriate connection in the **Available IBM i Connections** dialog.
3. Click **OK**.

Step 2 Select the language from the **Available languages** drop-down list.

Step 3 Click the **Change Language** button to confirm.

3.4 Managing the Free Form Conversion Options

The default status of the Free Form Conversion options are managed from the plug-in's preference page with the same name (*Window > Preferences > ARCAD-Transformer RPG > Free Form Conversion Options*). These parameters enable you to predefine a number of transformation preferences.

When transforming a source member, the same parameters are available to change as needed in the **Converted Source Member Properties** window. However, configuring them in the preferences menu defines the default statuses.

The following table contains a link to a complete description of each parameter and the values allowed for each parameter, with the default setting indicated in bold.

| Conversion Option | Values |
|--|-----------------------------------|
| Replace Existing Member (REPLACE) | [optional] *YES, *NO |
| Convert Program calls (CVT_CALL) | [optional] *YES, *NO |
| Convert GoTo (CVT_GOTO) | [optional] *NO, *BASE, *ADVANCED |
| GOTO Label (TAGFLDNAME) | [optional] Character value, ATag |
| Convert Key List (CVT_KLIST) | [optional] *YES, *NO |
| Convert MOVEA (CVT_MOVEA) | [optional] *NO, *BASE, *ADVANCED |
| Convert Subr. to procedures (CVT_SUBR) | [optional] *YES, *NO |
| Use %ParmNum (USEPARMNUM) | [optional] *YES, *NO |
| Indentation Size (char) (INDENT) | [optional] 0-5, 2 |
| Indent Comments (INDENTCMT) | [optional] *YES, *NO |
| Case for operation codes (OPCODECASE) | [optional] *MIXED, *UPPER, *LOWER |

Table 31: Summary of Conversion Options

| Conversion Option | Values |
|---|-----------------------------------|
| Case for the B.i.F. (BLTFNCCASE) | [optional] *MIXED, *UPPER, *LOWER |
| Case for special words (SPCWRDCASE) | [optional] *MIXED, *UPPER, *LOWER |
| Case for key words (KEYWRDCASE) | [optional] *MIXED, *UPPER, *LOWER |
| Convert to "Fully Free" (FULLYFREE) | [optional] *YES, *NO |
| Max nbr of not "Free" blocks (MAXNOTFREE) | [optional] *NONE, 1-999, *NOMAX |
| First Column (Fully Free) (FIRSTCOL) | [optional] 1, 2-5 |
| Pre-compilation Clauses (PRECPL) | [optional] *ARCAD, *ALDON |
| Mark the conversion type (FLGCVTTYPE) | [optional] *YES, *NO, *KEEP |
| Clean Temporary Cross-references (CLRREF) | [optional] *YES, *NO |
| Clean Modified Lines (CLRFRMCHG) | [optional] *YES, *NO |

Table 31: Summary of Conversion Options

Replace Existing Member (REPLACE)

If you have specified a destination file and source member that already exists, specify here whether the contents should be replaced.

| Parameter | Description |
|-----------|---|
| *YES | The contents of the destination source member are replaced by the converted source. |
| *NO | The conversion is not done if the destination source member already exists. |

Table 32: Replace Existing Member (REPLACE) Parameters

Convert Program calls (CVT_CALL)

Specify if you wish to convert all traditional program calls (CALL) or procedure calls (CALLB), along with the entry/exit parameter declarations via *ENTRY PLIST to Free syntax.

For this the prototypes are created in the D-specs for each different program or procedure call (or if the parameter types/lengths are different); the name of the prototype starts **Pgm_** if it's for a program and **Prc_** if it's for a procedure.

After that, each "traditional" call is replaced by a prototyped call, with the parameters that were specified with PARM.

For the parameters of the source itself (that were on the *ENTRY PLIST), a procedure interface (Pi) is also created; it takes the same name for the variables that were specified in PARM statements. However, there is a special case, when the variable in PARM is a DS: in this case another variable **ds_PiParm_nnn** is defined in the procedure interface, and the DS points to it via pointer "pds_PiParm_nnn" set at the start of the program.

It may be that these variables are already defined in D-specs as well: in which case, their previous definition is deleted (except if they are in a COPY clause, which will cause an error when compiling the new source).

If factors 1 and 2 were used for the PARM, then assignment instructions are added before/after the CALL or at the start and/or end of execution (for *ENTRY PLIST), if the execution is terminated with RETURN instructions assignment instructions will be inserted where necessary.

If PLIST / PARM statements are defined, they are deleted.

| Parameter | Description |
|-----------|--|
| *YES | <p>CALL / CALLB instructions are replaced by prototyped calls. If *ENTRY PLIST is used, the program itself is prototyped.</p> <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> <p>Note: When an indicator "LR" (position 7576) was defined on the CALL/CALLB instruction, it was rarely voluntary: in that case, the following warning is inserted in the source, but the conversion is done, without managing this indicator: // *CVTWNG. The *INxx indicator was defined on column 7576 for the CALL: Removed.</p> </div> |

Table 33: Convert Program calls (CVT_CALL) Parameters

| Parameter | Description |
|-----------|---|
| | Only the following cases prevent the replacement of CALL / CALLB with prototyped calls: <ul style="list-style-type: none"> • Indicator "LR" (position 75-76) is defined on the CALL/CALLB instruction. • A PLIST is used in a COPY clause. • An *ENTRY PLIST contains a field declared with SQLTYPE(BLOB, CLOB, DBCLOB). |
| *NO | All CALL, CALLB, PLIST, PARM instructions remain in traditional syntax. |

Table 33: Convert Program calls (CVT_CALL) Parameters

Convert GoTo (CVT_GOTO)

Traditional branching opcodes GOTO / TAG / CABxx have no exact equivalent in Free syntax; normally it is necessary to restructure the code in order to transform it to structured programming.

However, the ACVTRPGFRE command allows the possibility to remove almost all these branching instructions in certain cases:

- When possible, using LEAVESR, LEAVE, ITER.
- By simulating the branching management in structured programming via the introduction of a variable containing the old label.

Of course, the structure of your program cannot be redesigned automatically, but this has the advantage of converting almost everything to Free syntax but the following cases always stay in traditional syntax, with GOTO/TAG:

- Branching from a subroutine to a label situated in the main body of the program (or procedure).
- Branching not respecting the nesting of structured programming (from outside of a loop or test/compare to within it).

```

Example
      If      Cdt1
      Goto   Act02
      Endif
      ...
      If      Cdt2
    
```

```
Act02   Tag
        ...
        EndIf
```

- In SQLRPGLE, branching made in an SQL instruction "WHENEVER ... GO TO label"
- Branching in the analyzed source to a label defined in a COPY clause.
- Branching from a COPY clause to a label defined in the analyzed source.
- Branching to a label that is not situated at the same control-level indicator level Lo, L1..L9, LR (or later).
- Branching done inner a block of lines surrounded by conditional compilation directives (/IF ... /ENDIF)

| Parameter | Description |
|-----------|--|
| *NO | <p>All GOTO/TAG instructions remain in traditional syntax.</p> <p>ENDSR instructions with a label are split into 2 instructions (a TAG remaining in traditional syntax and an ENDSR transformed to Free syntax).</p> <p>CABxx instructions with a label are split into several instructions (test/comparison, management of any indicators, and a GOTO which remains in traditional syntax).</p> |
| *BASE | <p>Only branching operations that can be done directly in structured programming are modified:</p> <ul style="list-style-type: none"> • Branching to a label situated on an ENDSR instruction: replaced by LEAVESR. • Branching to a label defined by a TAG situated just before the ENDSR instruction: replaced by LEAVESR. • Branching to a label situated just before an unconditioned end-of-loop (ENDDO/ENDFOR): replaced by ITER. • Branching to a label situated just after an end-of-loop (ENDDO/ENDFOR): replaced by LEAVE. • Unnecessary branching (GOTO) instructions to labels that are located immediately afterwards are deleted. • Unused labels (TAG) are deleted. |
| *ADVANCED | <p>In addition to the *BASE case, replacing most of the branching by structured programming tests, with the help of a variable containing the label name, and conditioning lines of code to get to the place where the label was defined.</p> |

Table 34: Convert GoTo (CVT_GOTO) Parameters

| Parameter | Description |
|-----------|---|
| | <p><i>Example 1 - Branching to labels situated later in the code</i></p> <p>When the label is situated earlier in the code, an instruction is inserted allowing it to loop back (DoU).</p> <p>Before:</p> <pre> C If Cdt2 C Goto TRT02 C EndIf C If Cdt1 C Goto Trt01 C EndIf C Eval X = 0 ... C TRT01 Tag C Eval X = 1 ... C TRT02 Tag C Eval X = 2 C ... </pre> <p>After:</p> <pre> If Cdt2; ATag = 'TRT02'; EndIf; If ATag = *Blanks; If Cdt1; ATag = 'TRT01'; EndIf; </pre> |

Table 34: Convert GoTo (CVT_GOTO) Parameters

| Parameter | Description | | | | | | |
|-----------|--|-------|----|------|---|------|-------|
| | <pre data-bbox="646 256 1228 673"> EndIf; If ATag = *Blanks; X = 0; ... EndIf; If ATag = 'TRT01' or ATag = *blanks; ATag = *Blanks; X = 1; ... EndIf; // branch when ATag = 'TRT02' ATag = *Blanks; X = 2; </pre> <p data-bbox="541 714 1276 747"><i>Example 2 - Branching to labels situated previously in the code</i></p> <p data-bbox="541 771 1806 868">For more complex cases (branching and labels situated at different DOxxx/FOR loop levels; existence of LEAVE/ITER instructions at the location where 'DoU' instructions are inserted to loop back in the code; etc.), this management sometimes requires:</p> <ul data-bbox="571 889 1806 1036" style="list-style-type: none"> • the setting of the variable ATag with values '*LEAVE' or '*ITER', • a request to exit from a loop-level with LEAVE, • a comparison/test of values in the loop variable after the end of a loop, to loop-back or exit again to another loop-level. <p data-bbox="541 1052 634 1079">Before:</p> <table data-bbox="604 1101 1213 1161"> <tr> <td>C</td> <td>If</td> <td>Cdt3</td> </tr> <tr> <td>C</td> <td>Goto</td> <td>TRT03</td> </tr> </table> | C | If | Cdt3 | C | Goto | TRT03 |
| C | If | Cdt3 | | | | | |
| C | Goto | TRT03 | | | | | |

Table 34: Convert GoTo (CVT_GOTO) Parameters

| Parameter | Description | | |
|-----------|---------------|-------|-----------------|
| | C | | EndIf |
| | C | TRT00 | Tag |
| | C | | If Cdt1 |
| | C | | Goto Trt01 |
| | C | | EndIf |
| | C | | Eval X = 0 |
| | ... | | |
| | C | TRT01 | Tag |
| | C | | Eval X = 1 |
| | ... | | |
| | C | TRT02 | Tag |
| | C | | Eval X = 2 |
| | C | | If Cdt0 |
| | C | | Goto TRT00 <=== |
| | C | | EndIf |
| | C | | ... |
| | ... | | |
| | C | TRT03 | Tag |
| | C | | Eval X = 3 |
| | C | | If Cdt2 |
| | C | | Goto TRT02 <=== |
| | C | | EndIf |
| | C | | ... |
| | After: | | |
| | | | If Cdt3; |
| | | | ATag = 'TRT03'; |

Table 34: Convert GoTo (CVT_GOTO) Parameters

| Parameter | Description |
|-----------|--|
| | <pre> EndIf; DoU ATag <> 'TRT00' and ATag <> 'TRT01' If ATag = 'TRT00' or ATag = *blanks; ATag = *Blanks; If Cdt1; ATag = 'TRT01'; EndIf; EndIf; EndIf; If ATag = *Blanks; X = 0; ... EndIf; If ATag = 'TRT01' or ATag = *blanks; ATag = *Blanks; X = 1; ... EndIf; If ATag = 'TRT02' or ATag = *blanks; ATag = *Blanks; X = 2; If Cdt0; ATag = 'TRT00'; Iter; <=== EndIf; ... EndIf; // branch when ATag = 'TRT03' ATag = *Blanks; X = 3; </pre> |

Table 34: Convert GoTo (CVT_GOTO) Parameters

| Parameter | Description |
|-----------|---|
| | <pre> 02'; Iter; EndIf; ... EndDo; <=== </pre> |

Table 34: Convert GoTo (CVT_GOTO) Parameters

GOTO Label (TAGFLDNAME)

If you select ***ADVANCED** for the **Convert GoTo (CVT_GOTO)** parameter, specify the name of a new variable that will be created (if necessary) in the source, to memorize and test for the old label name.

Specify the name with lower-case letters, ensuring it is syntactically correct.

If the name indicated (example: **ATag**) is already used in your program, then a name will be generated by adding 2 digits: 'ATag01' or 'ATag02'.

Convert Key List (CVT_KLIST)

Specify if you want to convert the key lists (KLIST / KFLD) defined and used in the source to be converted.

There is no exact equivalent for KLIST / KFLD in Free syntax, but it is possible to indicate several key values directly in the file access instructions (CHAIN, SETLL, READE...).

| Parameter | Description |
|-------------|--|
| *YES | <p>KLIST (+ KFLD) definitions are deleted. In all the lines where KLIST instructions were used, they are replaced by the list of variables that were defined with KFLD (even if these lines were already in Free syntax).</p> <p>Before:</p> <pre> C KORDDT KLIST </pre> |

Table 35: Convert Key List (CVT_KLIST) Parameters

| Parameter | Description |
|-----------|--|
| C | KFLD WOrdHdr |
| C | KFLD a_Line (I) |
| ... | |
| C | KORDDDET CHAIN ORDDETF1 |
| After: | Chain (WOrdHdr : a_Line (I)) ORDDETF1; |
| | Only the following case prevents the deletion of KLIST/KFLD specifications: |
| | <ul style="list-style-type: none"> The KLIST / KFLD is used in at least one COPY clause. |
| *NO | The KLIST (+ KFLD) definitions are left in traditional syntax. They are still used in the file access instructions converted to Free syntax. |

Table 35: Convert Key List (CVT_KLIST) Parameters

Convert MOVEA (CVT_MOVEA)

The classic operation code MOVEA - assignment of values with array variables, considered like global strings which group all the elements - has no exact equivalence in Free form syntax.

However, depending on use cases, it's possible to manage their conversion more or less easily:

- Simple assignment of figurative constant,
- Simple assignment between 2 arrays with same type/length
- Usage of the %SubArr B.i.f.
- Usage of redefinitions of the arrays, as strings, using pointers.

Only the following cases can't be converted and leave in traditional syntax using MOVEA (only when they might be managed using a pointer):

- Arrays which total length of all the elements exceeds the maximum length allowed for a character string (65535 or 16383), when you are at V5R4 OS level.
- Arrays used as MOVEA result factor, but defined in the prototype of the program or procedure, with the CONST keyword.

| Parameter | Description |
|-----------|--|
| *NO | All MOVEA instructions stay in traditional syntax. |
| *BASE | <p>The following cases are done using a simple evaluation:</p> <ul style="list-style-type: none"> • Assignment of a figurative constant (*BLANK, *ZERO, *LOVAL, *HIVAL, *ON, *OFF, *ALL' ') to all the elements of an array • Assignment of a figurative constant (*ALL'xxxx', *ALLX'xxxxxx'), to all the elements of an array, when the length of the literal divides the length of one array element. • Assignment of a numeric value to all the elements of a numeric array • Assignment between an alphanumeric value and an array element which length is upper or equal, without padding blanks for the rest of the array. <p>The following cases are done using a partial evaluation of array(s), with %SubArr(...):</p> <ul style="list-style-type: none"> • Assignment of a figurative constant (*BLANK, *ZERO, *LOVAL, *HIVAL, *ON, *OFF, *ALL' ') beginning from one array element. • Assignment of a figurative constant (*ALL'xxxx', *ALLX'xxxxxx'), beginning from one array element, when the length of the literal divides the length of one array element. • Assignment between 2 arrays which have the same element lengths (for all the elements or just some elements). <p>When MOVEA is used to assign a list of 'o' or '1' values in the array of indicators, from one of its elements and for a maximum of 8 elements.</p> <div data-bbox="527 976 1818 1084" style="border: 1px solid black; padding: 5px;"> <p><i>Example</i></p> <p>MOVEA 'o100' *IN(15) then it is replaced by several instructions: *INxx = 'o'; or *INxx = '1';</p> </div> |
| *ADVANCED | <p>In addition to the *BASE case, replaces most of the MOVEA by an instruction block using redefinition as string(s) via pointer(s), in order to run the operation with a simple %SUBST.</p> <p>To do this, work variables are defined (when needed):</p> |

Table 36: Convert MOVEA (CVT_MOVEA) Parameters

| Parameter | Description |
|-----------|--|
| | <ul style="list-style-type: none"> • AFrmArrStr & pAFrmArrStr: redefinition as a string and pointer for an array used in Factor 2. • NFrmArrStrLen: Variable for the number of characters to be taken into account in the array used in Factor 2. • AToArrStr & pAToArrStr : redefinition as a string and pointer for an array used in result factor. • NToArrStrLen: Variable for the number of characters to be taken into account in the array used in result factor. • NChgArrStrLen: Variable for the number of characters to change in the array used in result factor. • If the variables are Unicode type, they are CFrmArrStr, etc. • If the variables are Graphic type, they are GFrmArrStr, etc. <p>Depending on each case, the corresponding instruction block in Free form may have more or fewer instructions:</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p><i>Example 1: Assignment of array elements to a string which is more long</i></p> <p>Before:</p> <pre> D a_A01 s 1a Dim(100) D B02 s 2a ... C Movea a_A01(4) B02 </pre> <p>After:</p> <pre> pAFrmArrStr = %Addr(a_A01(4)); %Subst(B02:1:2) = %Subst(AFrmArrStr:1:2); </pre> </div> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <p><i>Example 2: Assignment between 2 arrays that have different element lengths, with variables as indexes</i> (this case needs the most instructions)</p> <p>Before:</p> </div> |

Table 36: Convert MOVEA (CVT_MOVEA) Parameters

| Parameter | Description |
|-----------|--|
| | <pre> D a_A01 s 1a Dim(100) D a_B02 s 2a Dim(100) ... C Movea a_A01(X1) a_B02(X2) After: NFrmArrStrLen = (%Elem(a_A01) + 1 - X1) * 1; NToArrStrLen = (%Elem(a_B02) + 1 - X2) * 2; NChgArrStrLen = NFrmArrStrLen; If NChgArrStrLen > NToArrStrLen; NChgArrStrLen = NToArrStrLen; EndIf; pAFrmArrStr = %Addr(a_A01(X1)); pAToArrStr = %Addr(a_B02(X2)); %Subst(AToArrStr:1:NChgArrStrLen) = %Subst(AFrmArrStr:1:NChgArrStrLen); </pre> |

Table 36: Convert MOVEA (CVT_MOVEA) Parameters

Convert Subr. to procedures (CVT_SUBR)

This parameter allows you to specify if you want to convert the subroutines in the main part of the program to ILE procedures.

In the old source, subroutines (and their calls) are converted whether they use the old columned syntax or the new Free syntax.

Example

Before:

```

C          Exsr      SrAmount
C
C          .....
C  SrAmount  BegSr
                    
```

```

C
C          EndSr

After:

    SrAmount ();
    ...
    Dcl-Proc SrAmount;
    ...
    End-Proc SrAmount;
    
```

However, the new ILE procedures have no parameters and no local variables; they continue to use global variables in your source.

You can only request this change if you also convert the calculation specifications ([Convert calculation specs. \(CVTCLCSPEC\)\[*FREE\]](#)) - and the declaration specifications ([Convert declaration specs. \(CVTDCLSPEC\)\[*YES\]](#)) to Free format.

When you don't convert declaration specifications to Free format CVTDCLSPEC(*NO), for each new procedure, it will create DSpecs for the prototype and procedure interfaces and PSpecs for the beginning and end of each new procedure. This will happen when you compile programs for V5R4 or V6R1 OS level.

Warning! When the created object type is a *PGM, in order to use the ILE procedures, before being executed, your program must be genuinely ILE - specifically, it must no longer use the default activation group. If this is not the case, it will be necessary to recompile this source with DFACTGRP(*NO), specifying if necessary a value for parameter ACTGRP(...).

This may change the execution context of your program and you may need to modify the scope of any OVRDBF commands executed in the calling programs.

| Parameter | Description |
|-----------|---|
| *YES | Most subroutines are converted into procedures. However, the following subroutines cannot be converted to ILE procedures and therefore remain as subroutines: |

Table 37: Convert Subr. to procedures (CVT_SUBR) Parameters

| Parameter | Description |
|------------|--|
| | <ul style="list-style-type: none"> • Subroutines already located in an ILE procedure outside the main body of the source; • Subroutines using a GOTO with a "label" (TAG) located outside the subroutine; • Subroutines using a RETURN operation code, in order to exit from the program; • Special subroutines *INZSR and *PSSR; • Subroutines that are defined as file error management subroutines (keyword INFSR). <p>Subroutines using a file with the INFSR keyword (otherwise error message RNF5416 will be issued on compilation). All subroutines that call a subroutine that cannot itself be converted (recursively). If necessary, the new procedures are moved:</p> <ul style="list-style-type: none"> • after the routines that cannot be processed. • after O specifications. <div style="border: 1px dashed gray; padding: 5px; margin-top: 10px;"> <p>Note: In SQLRPGLE, no subroutine is processed if the SQL statement "WHENEVER ... GO TO label" is used.</p> </div> |
| *NO | Subroutines are not converted into procedures. |

Table 37: Convert Subr. to procedures (CVT_SUBR) Parameters

Use %ParmNum (USEPARAMNUM)

In V7R1, a new BiF. %ParmNum(Param_Name) has been introduced, in order to avoid having to hard-code the "parameter number" - compare to %Parms. Additionally, when new parameters are added later, this avoids possible modification errors.

| Parameter | Description |
|-------------|---|
| *YES | <p>When comparing the BiF. %Parms and an integer is found, the integer is replaced by the equivalent BiF. %ParmNum (Param_Name).</p> <ul style="list-style-type: none"> • If the comparison is type "< n", it is treated as "<= n-1" |

Table 38: Use %ParmNum (USEPARAMNUM) Parameters

| Parameter | Description |
|-----------|--|
| | <ul style="list-style-type: none"> If the comparison is type "> n", it is treated as ">= n+1" <p>This modification is only made if you have also requested the conversion of calculation specifications. It is then applied to comparison instructions already coded in Free syntax.</p> <p>This also works when the number of received parameters is retrieved using a variable defined in the SDS in positions 37 to 39 (or with *PARMS).</p> <div style="border: 1px dashed gray; padding: 5px; margin: 10px 0;"> <p>Note: This new syntax option is only available in V7R1Mo. If you do not have this option at your RPGLE or SQLRPGLE compiler level, the modification is not performed.</p> </div> |
| *NO | The comparisons using %Parms are not modified. |

Table 38: Use %ParmNum (USEPARMNUM) Parameters

Indentation Size (char) (INDENT)

Specifies the indentation to be applied to the Free source, according to the programming operation structures used (IF, SELECT, ... ENDxx).

Note: This does not affect the instructions already in Free syntax.

The indentation doesn't change for the instructions that are included after the compilation directives /ELSE, /ELSEIF or in 2 levels of /IF.

| Parameter | Description |
|-----------|--|
| 0 | No indentation is performed. |
| 1-5 | Number of blanks added for each new indentation. |

Table 39: Indentation Size (char) (INDENT) Parameters

Indent Comments (INDENTCMT)

Specifies whether to indent the comments preceding indented instructions.

Note: This does not affect the comments already in Free syntax.

| Parameter | Description |
|-----------|--|
| *YES | If there is enough space, comments are indented and aligned with the following instructions. |
| *NO | Converted comments are not indented - they begin in column 8. |

Table 40: Indent Comments (INDENTCMT) Parameters

Case for operation codes (OPCODECASE)

Choose the case for the calculation operation codes converted to Free syntax (e.g. EvalR, Chain, When...).

Note: This does not affect the instructions already in Free syntax.

| Parameter | Description |
|-----------|--|
| *MIXED | Operation codes are written in mixed case: the first character of each "word" of the operation code is in upper case and the other characters are in lower case. |
| *UPPER | All operation codes are written in upper case. |
| *LOWER | All operation codes are written in lower case. |

Table 41: Case for operation codes (OPCODECASE) Parameters

Case for the B.i.F. (BLTFNCCASE)

Choose the case for the "built-in-function" names on lines converted to Free syntax (e.g. %Subst, %Len, %Scan...).

Note: This does not affect the instructions already in Free syntax.

| Parameter | Description |
|-----------|--|
| *MIXED | Built-in-function names are written in mixed case: the first character of each "word" in the built-in-function name is in upper case and the other characters are in lower case. |
| *UPPER | Built-in-function names are written in upper case. |
| *LOWER | Built-in-function names are written in lower case. |

Table 42: Case for the B.i.F. (BLTFNCASE) Parameters

Case for special words (SPCWRDCASE)

Choose the case for the special words used in calculation specifications converted to Free syntax (e.g. *Blank, *Null, *Zero...).

Note: This does not affect the instructions already in Free syntax.

| Parameter | Description |
|-----------|--|
| *MIXED | Special words are written in mixed case: the first character of each "word" in the operation code is in upper case and the other characters are in lower case. |
| *UPPER | Special words are written in upper case. |
| *LOWER | Special words are written in lower case. |

Table 43: Case for special words (SPCWRDCASE) Parameters

Case for key words (KEYWRDCASE)

Choose the case for the keywords used in declaration specifications converted to Free syntax (e.g. DclDs, Char, Keyed...)

Note: This does not affect the instructions already in Free syntax nor the keywords already in H, F, P or D specifications.

| Parameter | Description |
|-----------|---|
| *MIXED | Keywords are written in mixed case: the first character of each "word" in the operation code is in upper case and the other characters are in lower case. |
| *UPPER | All keywords are written in upper case. |
| *LOWER | All keywords are written in lower case. |

Table 44: Case for key words (KEYWRDCASE) Parameters

Convert to "Fully Free" (FULLYFREE)

Choose to convert the source member to "Fully-Free" which ensures that the source lines can occupy all the columns, starting from the first column, up to the source file record length.

Important! This syntax option is only available from system version V7R2Mo, via the installation of the "Technology Refresh 3" PTF group (IBM i 7.2 TR 3). It is also available in "Technology Refresh 11" (IBM i 7.1 TR 11) V7R1. Normally, the conversion to "Fully-Free" will not be done unless all the source lines can be converted to Free syntax. (see [Max nbr of not "Free" blocks \(MAXNOTFREE\)](#))

If you do not have this for your RPGLE ou SQLRPGLE compiler level, the conversion will not be allowed.

It is also necessary to request the conversion of the declaration specifications.

| Parameter | Description |
|-----------|--|
| *YES | The source is converted to Fully-Free. The special **FREE directive is added as the first line and the lines of code are shifted to the left (see First Column (Fully Free) (FIRSTCOL)). If you want to keep the comments that are in columns 1-5, select *KEEP for the parameter Mark the conversion type (FLGCVTTYPE) . They are then moved to the end of the line. The data lines for the compile time arrays and tables at the end of the compilation source, after the "***" or **CTDATA directives are not affected by these modifications. |
| *NO | The conversion to "Fully-Free" syntax is not done. |

Table 45: Convert to "Fully Free" (FULLYFREE) Parameters

Max nbr of not "Free" blocks (MAXNOTFREE)

If you selected ***YES** for the [Convert to "Fully Free" \(FULLYFREE\)](#) parameter, choose whether or not to combine "fully free" syntax with "traditional" syntax.

In principle, a source that has been converted to "Fully-Free" syntax cannot contain any source line that is in "traditional" syntax. However, ARCAD-Transformer RPG allows the possibility to obtain a source that is almost "Fully-Free", but has some blocks that remain in traditional syntax.

Warning! IBM does not officially support this! It is not recognized in the RDi 9.5 editor but it is supported by the RPGLE compiler.

In order to get a real "Fully-Free" source, editable using RDi, you must modify these blocks of lines either by *rewriting them in Free syntax* (mainly for C-Specs) or by *moving them to /COPY clauses* (mainly for F-, I- or O-Specs).

| Parameter | Description |
|---------------|---|
| *NONE | No source lines in traditional syntax are permitted. If at least one source line cannot be converted to "Free" syntax, then the source is not converted to "Fully-Free" syntax. |
| 1-999 | Enter the maximum number of "source line blocks" not converted to "Free" syntax. They will be enclosed in special directives to allow compilation. <div style="border: 1px solid #0056b3; padding: 5px; margin: 10px 0;"> <p><i>Example</i></p> <pre>**END-FREE C GOTO TAG07 **FREE</pre> </div> |
| *NOMAX | No limit is put on the maximum number of "source line blocks" that remain in traditional syntax. |

Table 46: Max nbr of not "Free" blocks (MAXNOTFREE) Parameters

First Column (Fully Free) (FIRSTCOL)

If you selected ***YES** for the [Convert to "Fully Free" \(FULLYFREE\)](#), choose on which column to start the source line.

Note: When the lines are indented, they are shifted to the right.

| Parameter | Description |
|-----------|--|
| 1 | By default, the non-indented source lines start in column 1 (as opposed to column 8 in Free syntax). |
| 2-5 | The non-indented source lines start in column 2 to 5. |

Table 47: First Column (Fully Free) (FIRSTCOL) Parameters

Pre-compilation Clauses (PRECPL)

Specifies which syntax has been used for the pre-compilation clauses added at the start of the source.

| Parameter | Description |
|-----------|--|
| *ARCAD | By default, the syntax of ARCAD pre-compilation clauses is used. |
| *ALDON | Aldon pre-compilation clause syntax is used. |

Table 48: Pre-compilation Clauses (PRECPL) Parameters

Mark the conversion type (FLGCVTTYPE)

Specify if you wish to place an identifying mark for each type of conversion in columns 1-5 of the converted source.

| Parameter | Description |
|-----------|---|
| *YES | <p>In columns 1-5, a mark is added identifying the conversions as follows:</p> <ul style="list-style-type: none"> Gnnn: conversion of branching instructions (GOTO/TAG) Cnnn: conversion of program calls (CALL/CALLP) Xnnn: conversion of other operation codes Snnn: instruction not converted - no equivalent in Free syntax |

Table 49: Mark the conversion type (FLGCVTTYPE) Parameters

| Parameter | Description |
|--------------|---|
| | <div style="border: 1px dashed gray; padding: 5px;"> <p>Note: This is mainly for the development of conversions, or in case of conversion problems.</p> </div> |
| *NO | Columns 1-5 of the new source are set to blank (for the added or modified lines). |
| *KEEP | <p>The values (comments) in the old source in columns 1-5 are retained for added or modified lines (in most cases). However, any special color attribute characters are replaced by blanks.</p> <div style="border: 1px dashed gray; padding: 5px;"> <p>Note: Keeping these comments in columns 1-5 can be annoying in RDi because they can appear in the "active" columns of the source, if we make a shift to the right using the "RRn" LPEX source editor command.</p> </div> |

Table 49: Mark the conversion type (FLGCVTTYPE) Parameters

Clean Temporary Cross-references (CLRREF)

Specifies whether to delete X-Ref data generated at the start of the process and used for conversion of the RPGLE.

| Parameter | Description |
|-------------|---|
| *YES | The temporary X-Ref data is not conserved. |
| *NO | The temporary X-Ref data is conserved at the end of processing. They are therefore never cleared. |

Table 50: Clean Temporary Cross-references (CLRREF) Parameters

Clean Modified Lines (CLRFRMCHG)

Specify if you want to delete the converted source lines added to Arcad file AARFCHSF1.

| Parameter | Description |
|-----------|--|
| *YES | The lines of converted source put in file <i>AARFCHSF1</i> are not retained. This is the recommended value if you have specified a destination source member. |
| *NO | The lines of converted source are retained at the end of the process in file <i>AARFCHSF1</i> . These lines will never be purged. This is the required value if you specified *NONE for the destination source member. |

Table 51: Clean Modified Lines (CLRFRMCHG) Parameters

3.5 Managing the Default RPG to RPGLE Conversion Options

If your source code is in RPG form (not RPGLE), you must first convert it to RPGLE.

RPG to RPGLE conversion parameters are managed from the plug-in's preferences (*Window > Preferences > ARCAD-Transformer RPG > RPG to RPGLE Conversion Options*).

| Parameter | Description |
|----------------------------------|---|
| Temporary Library | The name of the library that contains the source file where the RPGLE converted source member will be generated (a location to store temporary objects, for example). |
| Source File | The source file where the RPGLE converted source member will be generated. This is the version that will later be converted to free. |
| Extended Parameters | Additional values that will be added "as is" to the <code>CVTRPGSRC</code> command before its execution. |
| Create the Target Source File | If this option is checked, the source file defined in the Source File field will be automatically created if it does not exist. |
| Replace the Target Source Member | If this option is checked, any existing Target Source Member with the same name as the current member created during the RPG to RPGLE conversion will be deleted (replaced) during the conversion. This is most likely the case if the conversion has already been performed in the past. |

Table 52: The RPG to RPGLE Conversion Options

4 Preparing the ARCAD-Transformer RPG Environment

Chapter Summary

| | |
|--|----|
| 4.1 Create a Connection..... | 64 |
| 4.2 Login to a Connection..... | 65 |
| 4.3 Create a Library for Modernized Source(s)..... | 65 |
| 4.4 Create a Source File..... | 68 |
| 4.5 Create a Source Member Filter..... | 70 |
| 4.6 Create a Library List..... | 71 |

This chapter provides instructions for creating and connecting to a new IBM i connection for ARCAD-Transformer RPG as well as for creating a connection, a new development library to contain modernized source code and a new source member.

Important You are not required to create a new profile or a new target library to use ARCAD-Transformer RPG, however you are required to use libraries and library lists to successfully transform data.

If you do create a new library, it is your responsibility to ensure the objects mentioned in the following example(s) do not already exist on your system. If they are already present you will need to substitute values.



Reference For more information about why library lists are necessary, see [About the ACVTRPGFRE Command on page 17](#).

The ARCAD-supplied library called **ARCAD_SMPL** is used in the following example(s).

4.1 Create a Connection

A connection in RDi is roughly equivalent to a green screen session in IBM i Access.

Follow the subsequent steps to create a connection to your IBM i after installing and configuring ARCAD-Transformer RPG.

- Step 1** Open the  **Remote Systems** view (*Window > Show View > Remote Systems > Remote Systems*)
- Step 2** Right-click anywhere in the menu and select **New >  Connection**.
- Step 3** Select **IBM i**, then click **Next**.
- Step 4** Enter the required connection information in the **New Connection** dialog.

In the example below, the IBM i **Host name** is *10.100.10.190*, yours will be different. The **Description** field is optional.

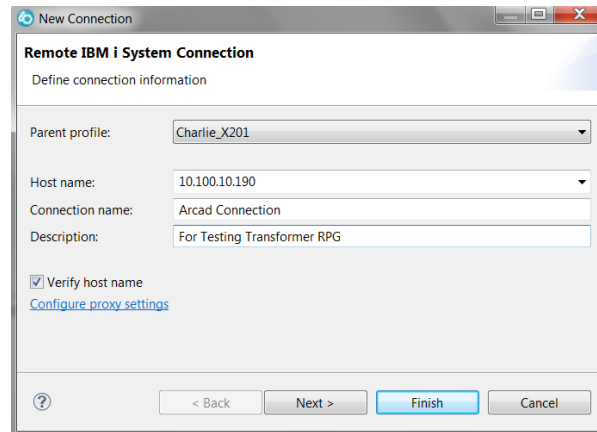


Figure 1: Example of a New Connection (IBM i)


Step 5 Click **Finish**.

4.2 Login to a Connection

When the connection created appears in the list of remote connections (**Remote Systems** view), you can connect to it.

Step 1 Right click on the connection and select **Connect**.

Step 2 Enter your user ID and password in the log in screen.

Result When the connection is established a green arrow appears next to the connection name (). This indicates an active connection.

Note: You can have multiple active connections to the same physical system.

4.3 Create a Library for Modernized Source(s)

It is necessary to have libraries containing sources and files used by the program online.

Important! You are not required to create a new library to use ARCAD-Transformer RPG, however you are required to use libraries to successfully transform data.

Reference For more information about why library lists are necessary, see [About the ACVTRPGFRE Command on page 17](#).

The examples and instructions below are for illustration purposes.

There are two recommended ways to create a new library for ARCAD-Transformer RPG. You can use the `CRTLIB` command in the command line, or use the connection's menu in the **Remote Systems** view.

4.3.1 Creating a Library from the Command Line

Step 1 Open the **Commands Log** view (*Window > Show View > IBM i > Commands Log*)

Step 2 Ensure the command will be executed in the **Normal** mode.

Note: If the Commands Log view is empty, click the drop-down prompt at the top right of the view and select **Show Log**, then your connection name.

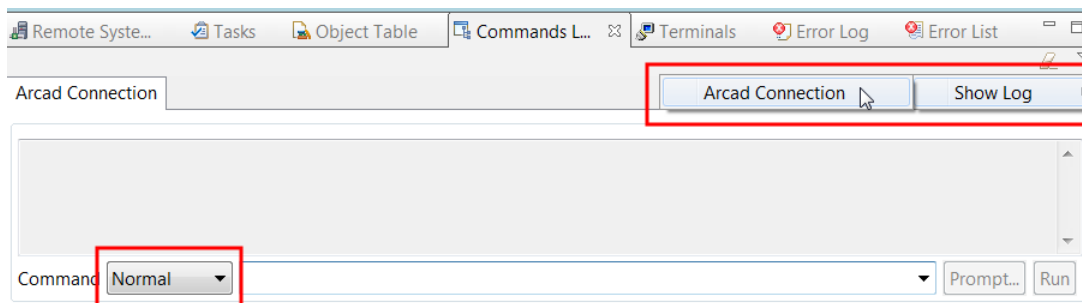


Figure 2: Command Line in the Command Log view

Step 3 Enter `CRTLIB` in the **Command** field and click **Prompt...**

Result The **Create Library (CRTLIB)** dialog displays.

Step 4 Enter the required details for the library, then click **OK**.

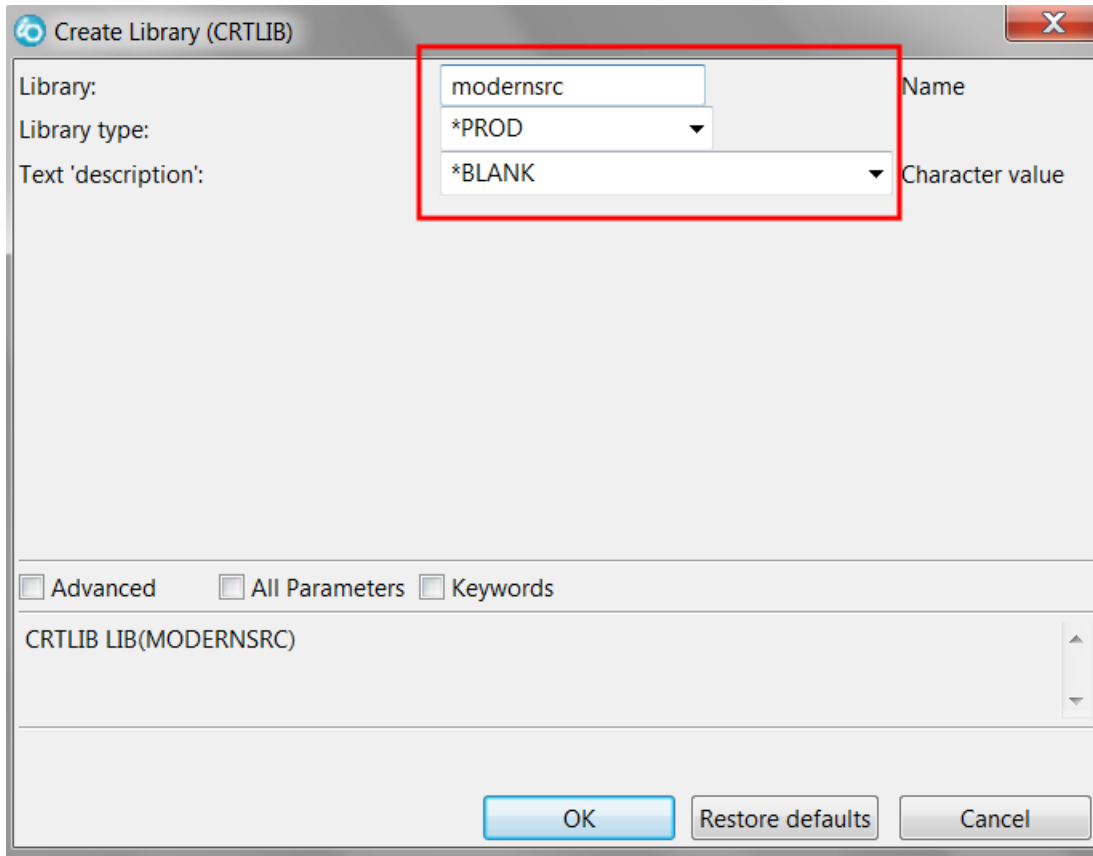


Figure 3: Create Library (CRTLIB) dialog

Result Confirmation is displayed in the **Command Log** view and the library appears in the list of **User libraries** (*Objects > User libraries*).

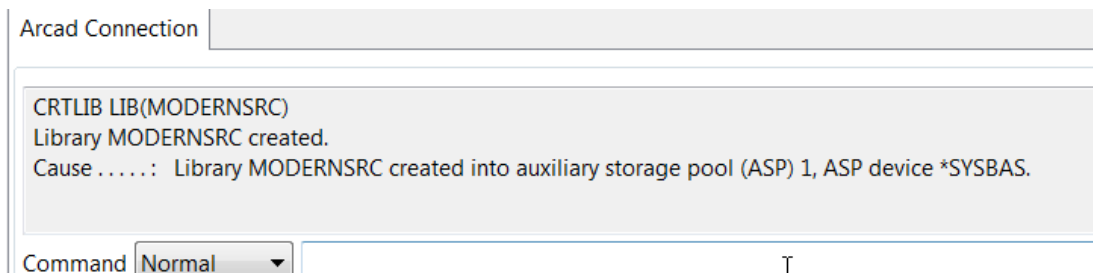


Figure 4: Confirmation Library was created in Commands Log view

4.3.2 Creating a Library from the Remote Systems View

- Step 1** From the **Remote Systems** view, expand the connection for which you want to create a library.
- Step 2** Expand the **Commands** menu.

Step 3 Double-click on **Create library**.

Result The **Create Library (CRTLIB)** dialog displays.

Step 4 Enter the required details for the library, then click **OK**.

Refer to [Figure 3: Create Library \(CRTLIB\) dialog](#).

Result The new library is created and appears in the list under **Create library** and in the list of **User libraries (Objects > User libraries)**.

4.4 Create a Source File

In the following example, a new physical source file is created based on an existing one from the **ARCAD_SMPL** library. This is the source file that will contain the modernized content.

There are two recommended ways to create a new, duplicate source file for ARCAD-Transformer RPG. You can use the `CRTDUPOBJ` command in the command line, or use the connection's menu in the **Remote Systems** view.

4.4.1 Duplicating a Source from the Command Line

Step 1 Open the **Commands Log** view (*Window > Show View > IBM i > Commands Log*)

Step 2 Ensure the command will be executed in the **Normal** mode.

Step 3 Enter `CRTDUPOBJ` in the **Command** field and click **Prompt...**

Result The **Create Duplicate Object (CRTDUPOBJ)** dialog displays.

Step 4 Enter the required details for the source object, then click **OK**.

In the example below, the new source file is connected to the library created in [Create a Library for Modernized Source\(s\)](#) on page 65.

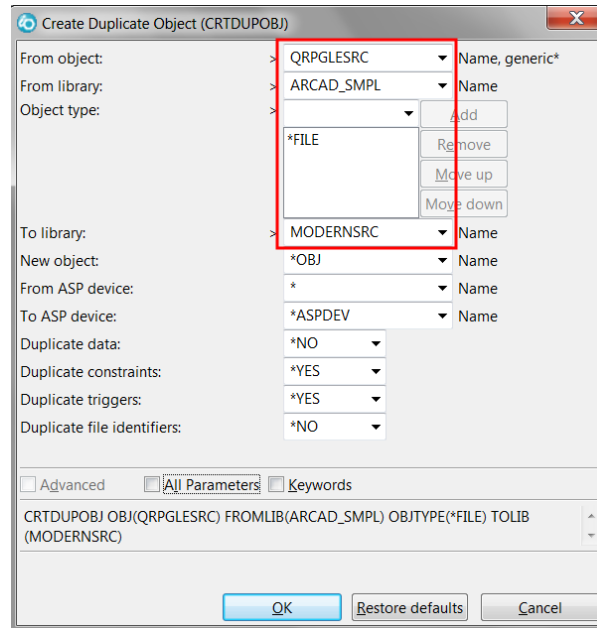


Figure 5: Create duplicate source object dialog

Result Confirmation is displayed in the **Command Log** view and the source appears in the list under the library defined in the **To Library** field in the dialog (*Objects > User libraries > MODERNSRC*).

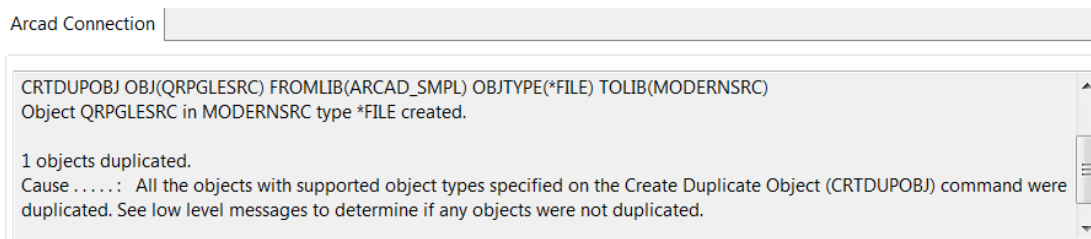


Figure 6: Confirmation Source was created in Commands Log view

4.4.2 Duplicating a Source from the Remote Systems View

Step 1 From the **Remote Systems** view, expand the connection for which you want to create a new source member.

Step 2 Expand the **Commands** menu.

Step 3 Double-click on **Create duplicate object**.

Result The **Create Duplicate Object (CRTDUPOBJ)** dialog displays.

Step 4 Enter the required details for the source object, then click **OK**.

Refer to [Figure 5: Create duplicate source object dialog](#).

Result The new source object is created and appears in the list under the library defined in the **To Library** field in the dialog (*Objects > User libraries > MODERNSRC*).

4.4.3 Removing Empty Members from a Duplicate Object

Empty members are created during the CRTDUPOBJ and must be removed once the new source physical file (QRPGLESRC) is added to the library (MODERNSRC).

Step 1 Open the **Commands Log** view (*Window > Show View > IBM i > Commands Log*)

Step 2 Ensure the command will be executed in the **Normal** mode.

Step 3 Enter RMVM in the **Command** field and click **Prompt...**

Result The **Remove Member (RMVM)** dialog displays.

Step 4 Enter the required details for the source object to purge, then click **OK**.

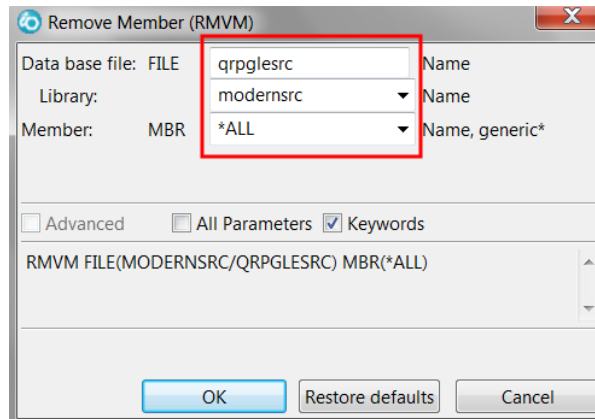


Figure 7: Remove Empty Members

4.5 Create a Source Member Filter

Filtering the source members available in your library(ies) can be very useful. You can regroup member types, for example, to easily find the files needed for modernization.

Follow the subsequent steps to create a source member filter.

Step 1 From the **Remote Systems** view, expand the connection for which you want to create a new filter.

Step 2 Expand the **Objects** menu, then the **Work with members...** menu.

Result The **New Member Filter** dialog displays.

Step 3 Enter the following values to define the IBM i member to filter:

| | |
|---------------|---|
| Library | This is the library the filter will apply to. |
| File | This is the file in the defined library to filter. |
| Member Filter | Leave the asterisk as is to search through all members. |

The **Member text** and **Member type** fields should also be unfiltered to open the filter to a maximum of members available.

In the example below, the library and file correspond to those created in [Create a Library for Modernized Source\(s\)](#) on page 65 and [Create a Source File](#) on page 68.

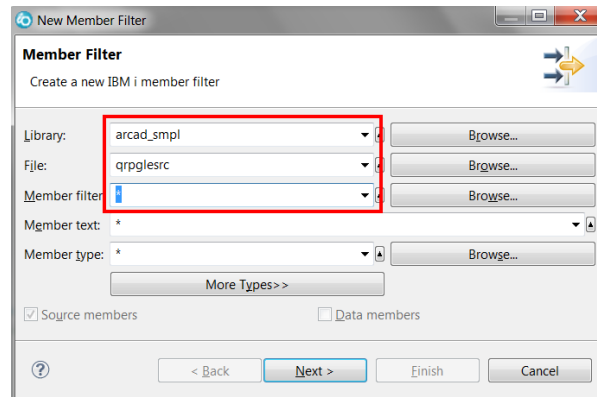


Figure 8: New Member Filter dialog page 1

Click **Next >** to continue.

Step 4 Enter the name of the new filter in the **Filter Name** field and ensure the **Only create filter in this connection** option is checked.

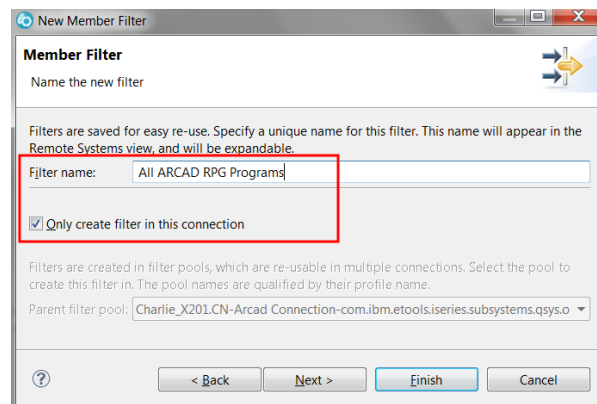


Figure 9: New Member Filter dialog page 2

Step 5 Click **Finish**.

Result The new filter is created and appears in the root of the **Objects** menu (*Objects > Filter*). Expanding the new filter displays all the files included in the scope of the filter.

If you followed the example(s) above, the list displays all the RPG source members.

4.6 Create a Library List

A well-structured library list is essential to a successful source code conversion. Invalid library lists are often the cause of errors during conversion.

Note: A successful RDi source verification will eliminate many of your failed conversion attempts.

The conversion requires the same runtime environment as any program compile would.

For a source member to be compilable, your RDi session's library list must contain all of the necessary libraries. This includes libraries that contain database files, binding directories, service programs, /COPY members etc. In the following example(s), only one library is required: **ARCAD_SMPL**.

Important! You are not required to create a new library list to use ARCAD-Transformer RPG, however you are required to use library lists to successfully transform data.

Reference For more information about why library lists are necessary, see [About the ACVTRPGFRE Command on page 17](#).

There are two recommended ways to add libraries to a list for ARCAD-Transformer RPG. You can use the `ADDLIB` command in the command line, or use the connection's menu in the **Remote Systems** view.

4.6.1 Adding a Library Entry from the Command Line

Step 1 Open the **Commands Log** view (*Window > Show View > IBM i > Commands Log*)

Step 2 Ensure the command will be executed in the **Normal** mode.

Step 3 Enter `ADDLIB` in the **Command** field and click **Prompt...**

Result The **Add Library List Entry (ADDLIB)** dialog displays.

Step 4 Enter the required details for the list, then click **OK**.

Enter the name of the library to add to the list in the **Library** field.

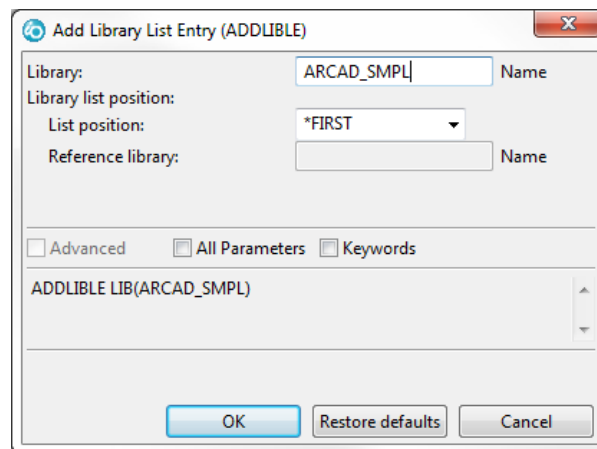



Figure 10: Add Library List Entry dialog

Result Confirmation is displayed in the **Command Log** view and the source appears in the **Library list** menu (*Objects > Library list*).

4.6.2 Adding a Library Entry from the Remote Systems View

Step 1 From the  **Remote Systems** view, expand the connection for which you want to create a new library list.

Step 2 Expand the  **Objects** menu.

Step 3 Right-click on **Library list**, then select **Add Library List Entry....**

Result The **Add Library List Entry** dialog displays.

Step 4 Enter the required details for the list, then click **OK**.

Enter the name of the library to add to the list in the **Library** field.

Refer to [Figure 10: Add Library List Entry dialog](#).

Result The new source object is created and appears in the **Library list** menu (*Objects > Library list*).

4.6.3 Configuring a Permanent Library List

Like a green screen session, the methods of adding a library described above only exist during the life of the active connection (session).

RD*i* enables you to assign a library to a connection on a permanent basis so that it is always available, even after you close the session. Add a library as a property of the connection so every time the connection is activated the libraries will automatically become part of your session.

Step 1 From the **Remote Systems** view, right-click on the connection that contains the library.

Step 2 Select **Properties**.

Step 3 From the **Properties for [Connection Name]** dialog, select the **Subsystems** menu on the right.

Step 4 Enter the name of the library list to permanently add to the sessions in the **Library** field, then click **Add**.

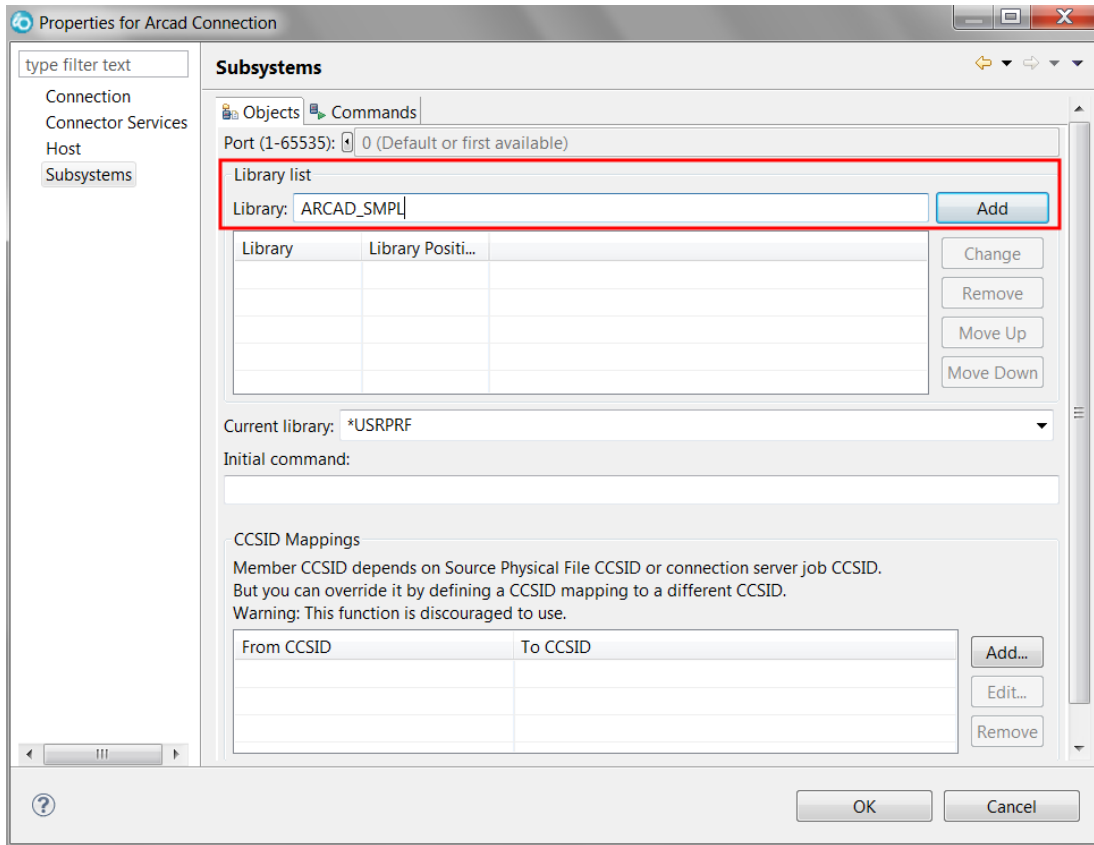


Figure 11: Add Library List to Session

Result The library is added to the table in the **Library list** section.

Step 5 Click **OK**.

RDi issues an `ADDLIB` command every time the connection is activated for each library listed in this table. This means the library list will automatically be prepared and ready for use immediately.

5 About the ARCAD-Transformer RPG Perspective

Chapter Summary

| | |
|---|----|
| 5.1 The Conversion List View..... | 75 |
| 5.2 The Conversion List Editor..... | 75 |
| 5.3 Filtering the Items in a Conversion List..... | 75 |

The ARCAD-Transformer RPG perspective is a dedicated perspective used to manage and edit Conversion Lists and execute massive conversions. This perspective consists of two elements:

- The Conversion List View
- The Conversion List Editor (displayed in the editor area)

The Conversion List View is used to manage the lists, and the editor is used to edit the lists' contents and execute actions on the items in the list.

5.1 The Conversion List View

This view consists of a list that displays all the existing Conversion Lists where each column represents one value of a List Header.

This view also has a toolbar to execute different actions such as adding or deleting a Conversion List or refreshing the contents of the list.

This view also provides an entry point to edit the contents of a selected Conversion List.

5.2 The Conversion List Editor

The **Conversion List** Editor allows you to access the Conversion Items and all the functions that are applicable.

You can use this editor:

- to edit the values of the List Header (Connection Name, Target Source file).
- to manage the contents of the list.
- to assign an object type to the Conversion Item.
- to execute a conversion.
- to access the conversion status of each Conversion Item.

5.3 Filtering the Items in a Conversion List

There are two ways to filter the items that are displayed in a Conversion List. You can filter them by their conversion status or by their informational values.

Follow the subsequent steps to filter the contents of a list by the values found in the columns (**Library, Source File, Source Type, Object Type, Conversion Date** etc.).

Step 1 Right click in the list area and select the  **Filter...** option.

Step 2 Click **Add** to create a new filter in the **Filter** dialog.

Step 3 Select the status you want to use as a filter criteria from the drop down list in the **Columns** field.


Step 4 Confirm the **Operator type** by selecting the correct symbol from the drop-down list.

Step 5 Enter the **Search value**.

Follow the subsequent steps to filter the contents of a list by their conversion status.

Step 1 Right click in the list area and select the **Filter** submenu.

Step 2 Check or uncheck the statuses as needed.

 **Note:** **N/A** means that you want to display the Conversion Items which have not been converted.

6 Selecting a Source Member

Chapter Summary

| | |
|---|----|
| 6.1 Select a Member from the i Projects view..... | 77 |
| 6.2 Select a Member from the Remote Systems view..... | 78 |

For efficiency, the conversion process needs to compile a source member. This means using a job where all the necessary settings (and the library list in particular) have been correctly prepared. To facilitate this, the conversion process uses the underlying IBM i connection related to the selected source member. Therefore, to prepare a conversion, you only need to execute the necessary actions on the parent IBM i connection.

The only preparation for a conversion job is the selection of the fully-qualified source member to convert (the library, the source file, the source type and the member name [see [Managing the Free Form Conversion Options on page 40](#) and [Managing the Default RPG to RPGLE Conversion Options on page 63](#)]).

ARCAD-Transformer RPG enables you to convert source members selected from both the **i Projects Navigator** and the **Remote Systems** views.

When the source member is selected, you can either transform it individually or add it to a Conversion List.

Reference For more information about these actions, refer to [Launching Single-File Conversions on page 80](#) and [Launching Mass Conversions on page 88](#).

6.1 Select a Member from the i Projects view

At least one IBM i Project with linked source members must be available to select a source member from this view.

- Step 1** Open the **i Projects Navigator View** (*Window > Show View > IBM i > i Projects Navigator*)
- Step 2** Browse to and expand the source file containing the .rpg, .rpgle or .sqlrpgle source member(s).

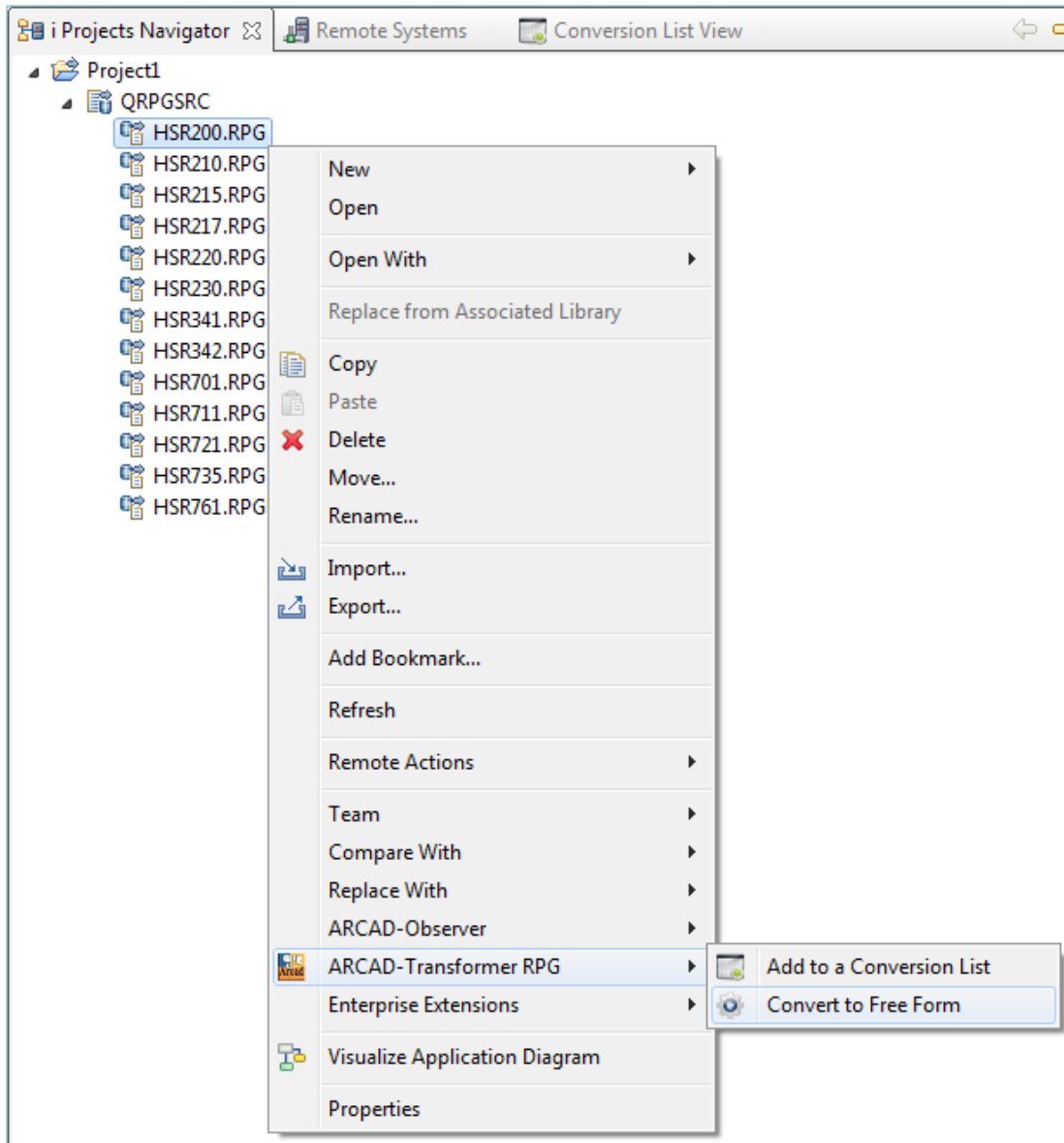


Figure 12: Selecting Source Member from i Projects Navigator view

6.2 Select a Member from the Remote Systems view

At least one IBM i server connection with linked source members must be available to select a source member from this view.

Step 1 Open the **Remote Systems** view (*Window > Show View > Remote Systems > Remote Systems*).

- Step 2** Create a library and a members filter, if necessary. See [Preparing the ARCAD-Transformer RPG Environment on page 64](#).
- Step 3** Browse and expand the source file containing the .rpg, .rpgle or .sqlrpgle source member(s).

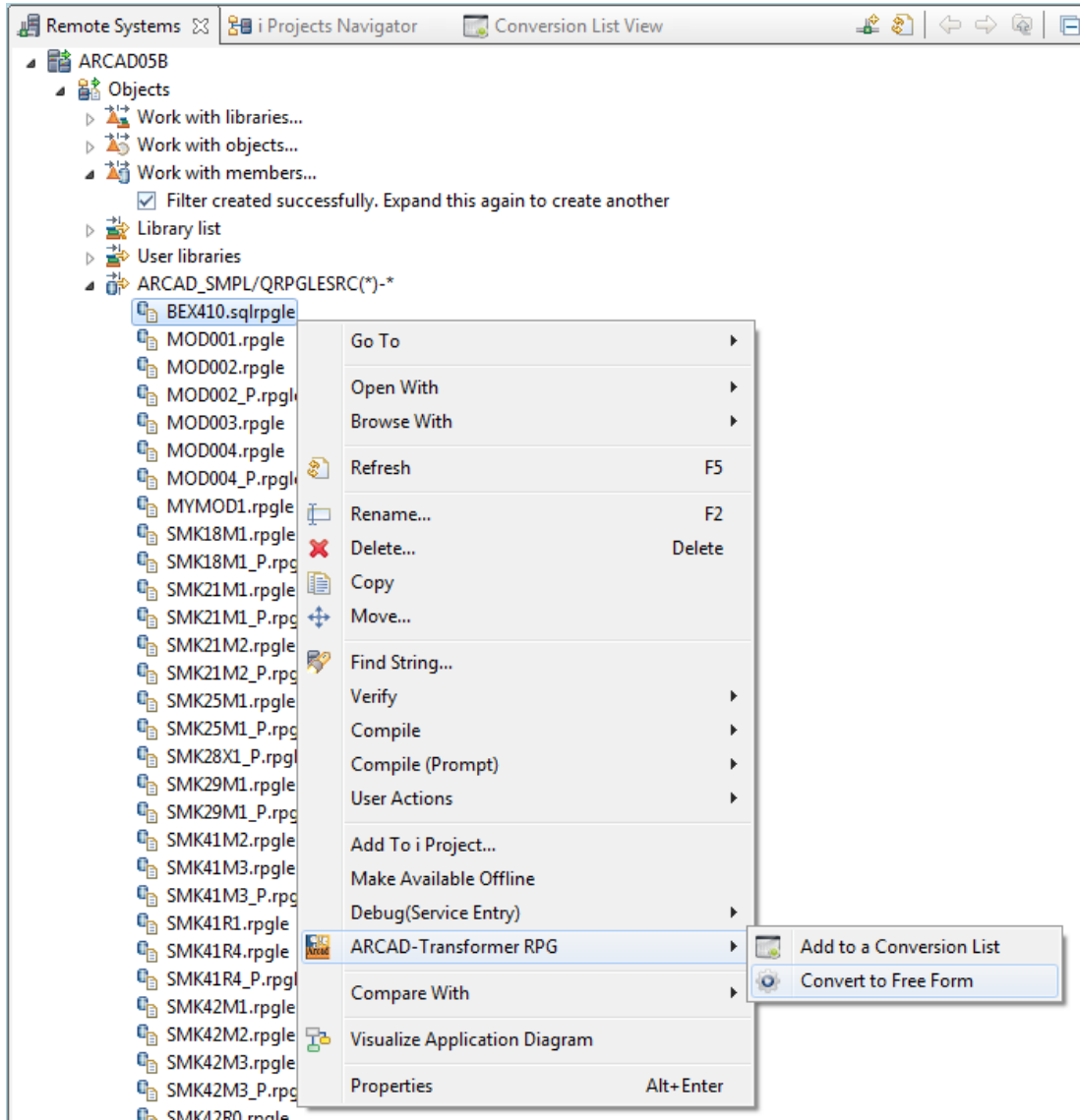


Figure 13: Selecting Source Member from Remote Systems view

7 Launching Single-File Conversions

Chapter Summary

| | |
|---|----|
| 7.1 Executing a Single-File Conversion..... | 80 |
| 7.2 Prompting the ACVTRPGFRE Command..... | 86 |

Conversions can be done on individual files or on groups of files saved in a Conversion List. This chapter contains information about preparing and executing the conversion of a single file.

Reference For more information about converting multiple files in a list, refer to [Working with Conversion Lists on page 88](#).

7.1 Executing a Single-File Conversion

Note: Values entered in the **Converted Source Member Properties** wizard are stored as preferences.

- Step 1** Select the source member to convert. See [Selecting a Source Member on page 77](#).
- Step 2** Right-click on the desired source member to display the contextual menu.
- Step 3** From the ARCAD-Transformer RPG menu item, select **Convert to Free Form**.
- Step 4** Define the necessary information in the **Converted Source Member Properties** wizard.

The following table contains a link to a complete description of each parameter defined only in this wizard (found below the table) and the values allowed for each parameters.

Reference For more information about the parameters in the wizard with preset default settings, (and that are not described in this table), refer to [Managing the Free Form Conversion Options on page 40](#).

| Parameter | Sub-Parameter Name | Values |
|---|--------------------|----------------------------------|
| Source File (SRCFILE) | 1: Library | [required] Name, *LIBL, *CURLIB |
| | 2: Source File | [required] Name, QRPGLSRC |
| Source Member (SRCMBR) | | [required] Name |
| Source Type (SRCTYPE) | | [optional] RPGLE, SQLRPGLE |
| Object Type (OBJTYPE) | | [optional] *PGM, *MODULE, *NONE |
| Convert calculation specs. (CVTCLCSPEC) | | [optional] *NO, *EXTFACT2, *FREE |

Table 53: Converted Source Member Properties

| Parameter | Sub-Parameter Name | Values |
|---|--------------------|--|
| Convert declaration specs. (CVTDCLSPEC) | | [optional] *YES, *NO |
| Destination Source File (TOSRCFILE) | 1: Library | [optional] Name |
| | 2: Source File | [optional] Single values: *NONE , *FROMFILE Other values: Qualified object name |
| Member Name (TOSRCMBR) | | [optional] Name, *FROMMBR |
| Source Line Date (SRCDATE) | | [optional] *CURRENT, *ZERO |

Table 53: Converted Source Member Properties

1. Define the **Original Source Member** information

Source File (SRCFILE)

The library and source file fields specify the name and library of the source file to be analyzed. These fields are automatically filled in with the information corresponding to the selected source member.

The possible values for the source file name are **QRPGLESRC**, the same value as the source file of the member to convert or nothing.

The possible values for the name of the library are:

| Parameter | Description |
|--------------|--|
| *LIBL | The object is searched for in the list of libraries. |
| *CURLIB | The specified object is searched for in the current library for the job. If no *curlib was specified, QGPL is used by default. |
| Library name | Indicate the name of the library containing the object. |

Table 54: Source File (SRCFILE) Library Parameters

Source Member (SRCMBR)

Specifies the name of the source member to analyze. This field is automatically filled in with the information corresponding to the selected source member.

The source member must be RPGLE or SQLRPGLE and it should be a main source. It may be a /COPY clause only if you don't convert calculation specifications.

Note: You cannot access the command prompt page if the selected source member is RPG source mem-



ber type. See [Prompting the ACVTRPGFRE Command on page 86](#).

Source Type (SRCTYPE)

Specifies the syntax for RPGLE or SQLRPGLE C specifications. This field is automatically filled in with the information corresponding to the selected source member.

| Parameter | Description |
|-----------|--|
| RPGLE | The source contains RPGLE instructions without embedded SQL. |
| SQLRPGLE | The source contains RPGLE instructions and embedded SQL. |

Table 55: Source Type (SRCTYPE) Parameters

Object Type (OBJTYPE)

For the conversion of calculation specifications (C), this parameter indicates whether the object directly created from this source is a ***MODULE** or ***PGM**, this information is required when converting to Free syntax for conversion of an **"*ENTRY PLIST ..."**.

However, for the conversion of the declaration specifications only, the value ***NONE** allows the normal preliminary compilation to be omitted. (useful particularly for COPY clauses).

| Parameter | Description |
|-----------|---|
| *PGM | For Free, the prototype for the input parameters of this program will have the EXTPGM keyword. |
| *MODULE | For Free, the prototype for the input parameters of this module will not have the EXTPGM keyword. |
| *NONE | <p>No object type is associated with this source; this value can only be used if you are converting only the declaration specifications (H, F, D, P). It is not allowed if you want to convert the calculation specifications (C).</p> <div style="border: 1px dashed orange; padding: 5px; margin-top: 10px;"> <p>Important! If the Object Type value is *NONE, the Convert calculation specs. (CVTCLCSPEC) must be set to *NO. A message dis-</p> </div> |

Table 56: Object Type (OBJTYPE) Parameters


| Parameter | Description |
|-----------|--|
| |  plays when the type is changed and the correct settings are automatically defined. |

Table 56: Object Type (OBJTYPE) Parameters

2. Define the specifications.

Convert calculation specs. (CVTCLCSPEC)

Specifies whether you want to convert the syntax for RPGLE and SQLRPGLE calculation specifications (C).

| Parameter | Description |
|-----------------------|--|
| *NO | No RPGLE syntax conversion is performed for calculation specifications (C). |
| *EXTFACT ₂ | When possible, old codes operations using the factor 1, factor 2 and result are converted to instructions using extended factor 2. |
| *FREE | All calculation instructions RPGLE (with some exceptions) are converted into Free syntax. |


Table 57: Convert calculation specs. (CVTCLCSPEC) Parameters

Convert declaration specs. (CVTDCLSPEC)

Specify if you wish to convert the syntax of the declaration specifications (H, F, D and P) for the RPGLE and SQLRPGLE.

| Parameter | Description |
|-----------|---|
| *YES | All declaration specifications in RPGLE (with some exceptions) are converted to Free syntax. In addition, clauses "C/Free" and "/End-Free" are all deleted. |
| *NO | The declaration specifications (H, F, D and P) remain in classic syntax. |

Table 58: Convert declaration specs. (CVTDCLSPEC) Parameters

 **Note:** This new syntax option is only available in V7R1Mo, with the installation of PTF Group "Technology Refresh 7". If you do not have this option at your RPGLE or SQLRPGLE compiler level, the conversion is not allowed.

In the following cases, the conversion of a declaration specification is not done because there is no equivalent declaration in Free syntax:

- in F specifications (pos. 18), use of a "File Designation" having the value P=Primary, S=Secondary, T=Array or Table, R=Address.
- in F specifications (pos. 19), use of "End of File"
- in F specifications (pos. 21), use of "Sequence"
- in F specifications (pos. 28), use of "Limits Processing"
- in F specifications (pos. 34), value of "Record Address Type" not blank and not 'A'.
- in F specifications (pos. 35), value of 'T' (Record Address File) for the "File Organization"
- in D specifications, use of keywords **FROMFILE** or **TOFILE** for the field names.

For an F specification using an output file with add (O in pos. 17, A in position 20), the conversion includes the removal of the ADD keyword in the O specification.

In addition, the conversion of a declaration specification is not done in the following case:

- use of a variable or procedure whose name exceeds 99 characters.

For a COPY clause the following rule is applied:

- If it only contains DS subfields, no 'End-Ds' is added at the end.
- In all other cases, an 'End- Ds' is added to the end. You must manually intervene for source lines that use the COPY clause, if the DS continues on lines that follow the /COPY.

3. Define the **Converted Source Member** information (used to define what kind of compilation will be executed).

Destination Source File (TOSRCFILE)

Specify the name of the destination source file and library, receiving the converted source member. You must have sufficient rights for the destination source file.

If the destination **Library** is QTEMP, the source file will be created with the same attributes as the origin source file. See parameter [Authorize QTEMP in batch \(BATCHQTEMP\)](#) to authorize QTEMP in batch mode.

If the destination library is not QTEMP, the source file must already exist.

The possible values for the **Source File name** are:

| Parameter | Description |
|-----------|--|
| *NONE | No source member is generated as output; in this case, |

Table 59: Destination Source File (TOSRCFILE) Parameters

| Parameter | Description |
|-------------------------|--|
| | remember to specify parameter Clean Modified Lines (CLRFRMCHG)(*NO) , in order to see the new source lines in file <i>AARFCHSF1</i> . |
| *FROMFILE | The destination library and source file are the same as the source file of the member to be converted; in this case, you must specify a different member name. |
| Name of the source file | Specify a name and a library for the destination source file. |

Table 59: Destination Source File (TOSRCFILE) Parameters

Member Name (TOSRCMBR)

Specify the destination source member, if you have specified a destination source file.

| Parameter | Description |
|--------------------|--|
| *FROMMBR | The destination source member has the same name as the source member analyzed. In this case, it cannot be in the same source file (in the same library). |
| Name of the member | Specify the member name receiving the converted source. |

Table 60: Member Name (TOSRCMBR) Parameters

4. Define the conversion options.

Further parameters are found on the second page of the wizard. These are the conversion options which can also be managed in the plugin's preferences menu.

Reference For more information about the conversion options, refer to [Managing the Free Form Conversion Options on page 40](#).

The only option not defined in the preferences is:

Source Line Date (SRCDATE)

Indicates how the date field is modified (for each source line) in the converted source.

| Parameter | Description |
|-----------|--|
| *CURRENT | For lines added or modified, the date for each record is set to the current date. For other non-modified lines, the previous date is retained. |

Table 61: Source Line Date (SRCDATE) Parameters

| Parameter | Description |
|-----------|---|
| *ZERO | For all records in the new source, the date is set to '000000'. |

Table 61: Source Line Date (SRCDATE) Parameters

Step 5 Click **Finish** to start the conversion or click **Next >** to prompt the command manually.

When the conversion is complete, if it was successful, you can open the new source member. If unsuccessful, an error message will display giving details of the error(s).

Reference For more information about consulting converted content, refer to [Understanding Conversion Results on page 94](#).

Reference For more information about manually running the command, refer to [Prompting the ACVTRPGFRE Command below](#).

If the conversion fails and you need help understanding the error, refer to [Troubleshooting on page 97](#).

7.2 Prompting the ACVTRPGFRE Command

ARCAD-Transformer RPG offers a way to prompt and manipulate the ACVTRPGFRE command just as it would be in a 5250 emulator. IBM i users will recognize the parameters in the command prompt menu. However, prompting the command manually accomplishes the same task as running the conversion via the wizard.

Follow the subsequent steps to prompt the command.

Note: Values entered in the command prompt are not stored as preferences.

Step 1 From the third page of the **Converted Source Member Properties** wizard, click the **Prompt the command** button to display the RDi command prompter.

Step 2 Enter the conversion options.

The values displayed in the command prompter are those defined on the two first pages of the wizard.

Reference Refer to [Executing a Single-File Conversion on page 80](#) and [Managing the Free Form Conversion Options on page 40](#) for information about the options available.

There is one parameter available in the command prompt that is not available in the wizard:

Authorize QTEMP in batch (BATCHQTEMP)

This parameter allows the authorization of an output source file located in QTEMP for the converted member, even if the job is executed in batch mode.

| Parameter | Description |
|-----------|--|
| *NO | By default, QTEMP is not allowed for the output source file library, when the conversion is done in batch. This especially avoids making a conversion from a GUI for which the result is not accessible. |
| *YES | Indicate *YES if the ACVTRPGFRE command is included in a macro that retrieves the source from QTEMP to integrate it into a development library. |

Table 62: Authorize QTEMP in batch (BATCHQTEMP) Parameters

Step 3 Click **OK** to display the command string.

Step 4 Click **Finish** to execute this command.

8 Launching Mass Conversions

Chapter Summary

| | |
|---|----|
| 8.1 Working with Conversion Lists..... | 88 |
| 8.2 Defining Object Types..... | 91 |
| 8.3 Converting Selected Conversion Items..... | 92 |
| 8.4 Converting an Entire List..... | 93 |

Conversions can be done on individual files or on groups of files saved in a Conversion List. This chapter contains information about preparing and executing mass conversions on items contains in Conversion Lists.

Reference For more information about converting individual files, refer to [Launching Single-File Conversions on page 80](#).

8.1 Working with Conversion Lists

In order to launch a conversion on a number of items that should share the same output library and source, you must create a Conversion List. This section contains information about preparing Conversion Lists.

A Conversion Item is a reference to the conversion of a source member. It contains the following information:

- The fully-qualified source member to convert (the library, the source file, the source type and the member name [see [Managing the Free Form Conversion Options on page 40](#) and [Managing the Default RPG to RPGLE Conversion Options on page 63](#)]).
- The object type used during the conversion.
- The name of the converted source member.
- The date, the status and the related message of the previous conversion.

You can group some Conversion Items into a conversion list if you want them to share the same IBM i connection as well as a unique source file where all of the converted source members will be saved.

All the source member compilations will run in the same job for a conversion list and all the results will be located in the same place. These shared values are stored within the List Header which is defined when a list is created and can be managed in the Conversion List Editor.

8.1.1 Creating Conversion Lists

There are two ways to create a new Conversion List. You can create one during the processes of adding an individual file to a list, and you can create one from the **Conversion List View**.

8.1.1.1 Add Conversion List from Wizard

Follow the subsequent steps to create a Conversion List from the **Add to a Conversion List Wizard**.

Step 1 Select a source member to add to a Conversion List (see [Selecting a Source Member on page 77](#)).

Step 2 Right-click on the desired source member to display the contextual menu.

Step 3 From the  ARCAD-Transformer RPG menu item, select **Add to a Conversion List**.

Step 4 Click at the bottom of the **Conversion List Selection** wizard.

Result The **Conversion List Creation Wizard** is displayed.

Step 5 Define the required information for the list (the List Header).

Reference For more information about these fields, refer to [Editing Conversion Lists on the next page](#).

Step 6 Click **Finish**.

Result The new Conversion List is created and available to be managed in the **Conversion List View**.

8.1.1.2 Add Conversion List from Conversion List View

Follow the subsequent steps to create a Conversion List from the **Conversion List View**.

Step 1 Click the  Add icon:

- in the tool bar.
- in the context menu (right-click in the body of the view, select **Add a Conversion List**).

Result The **Conversion List Creation Wizard** is displayed.

Step 2 Define the required information for the list.

Reference For more information about these fields, refer to [Editing Conversion Lists on the next page](#).

Step 3 Click **Finish**.

Result The new Conversion List is created and available to be managed in the **Conversion List View**.

8.1.2 Populating Conversion Lists

Follow the subsequent steps to add source member files to an existing Conversion List.

Step 1 Select a source member to add to a Conversion List (see [Selecting a Source Member on page 77](#)).

Step 2 Right-click on the desired source member to display the contextual menu.

Step 3 From the ARCAD-Transformer RPG menu item, select **Add to a Conversion List**.

Step 4 Select an existing Conversion List from the **Conversion List Selection** wizard.

Step 5 Click **Finish**.

Result The source file has been added to the Conversion List which is managed in the **Conversion List View**.

8.1.3 Editing Conversion Lists

Conversion Lists are managed in the **Conversion List View** (*Window > Show View > ARCAD-Transformer RPG > Conversion List View*).

Double-clicking on a list in the menu opens the **Conversion List** editor where the information entered during its creation is displayed, as well as a list of all of the source files included in it.

The defined properties in the List Header can be modified in the editor.

| Parameter | Definition |
|------------------|--|
| Connection Name | Click the Browse icon to select the server in which the list should be stored. |
| List Name | [<i>required</i>] Enter a unique name for the list. |
| List Description | Enter a short description of the list to help you identify it. |


The **Target Information** section contains parameters that define the shared output source of all the files in the list.

| Parameter | Definition |
|-------------|---|
| Library | [<i>required</i>] Enter the name of the library that contains the target source file. |
| Source File | [<i>required</i>] Enter the name of the shared source file where all the converted members will be saved. |

The list of source files contains all the key information about each source, including the converted member's object type and the date the conversion was made.

8.1.4 Deleting Conversion Lists

There are two ways to delete a Conversion List.

- Select the list(s) to delete from the Conversion List View and click the  Delete icon.
- Select Delete in the contextual menu by right-clicking on the list(s) to delete.

Click **OK** on the confirmation window to confirm the permanent removal of the selected list(s).

8.1.5 Editing Source Members

ARCAD-Transformer RPG allows you to edit original and converted source members. RDi is used to make these modifications.

8.1.5.1 Editing Original Source Members

Follow the subsequent steps to edit an original source member.

- Step 1** Open a Conversion List in the editor (see [Editing Conversion Lists on the previous page](#)).
- Step 2** Right-click on the Conversion Item that contains the original source member you want to edit.
- Step 3** Select **Edit** from the contextual menu, then **Original Source Member**.
- Result** The source editor opens enabling you to edit the member.

8.1.5.2 Editing Converted Source Members

Follow the subsequent steps to edit an original source member.

- Step 1** Open a Conversion List in the editor (see [Editing Conversion Lists on the previous page](#)).
- Step 2** Right-click on the Conversion Item that contains the converted source member you want to edit.
- Step 3** Select **Edit** from the contextual menu, then **Converted Source Member**.
- Result** The source editor opens enabling you to edit the member.

8.2 Defining Object Types

Defining an Object Type is mandatory to execute the conversion of a source member because it determines the way this source member will be compiled.

The Object Type is defined in the **Conversion Wizard** when executing a single-file conversion (see [Executing a Single-File Conversion on page 80](#)). However, because the mass-conversion wizard only allows you to enter common conversion options, this value must be defined before-hand for mass conversions.

Follow the subsequent steps to define the **Object Type** for items in preparation for mass conversion.

- Step 1** Open a Conversion List in the editor (see [Editing Conversion Lists on the previous page](#)).
- Step 2** Right-click on the Conversion Item(s) for which you need to change the Object Type.
- Step 3** Select **Update Object Type** from the contextual menu.

Step 4 Select the desired object type from the drop down list in the **Update Object Type Wizard**.

Step 5 Click **Finish** to complete the operation.

Step 6 Save the Conversion List.

Result The selected object type appears in the **Object Type** column in the Conversion List.


8.3 Converting Selected Conversion Items


Conversion Lists enable you to convert items that are saved in a list and ensure that the converted members share the same library and source. You can either convert all of the items in a list at the same time, or you can select some of them to convert.

Follow the subsequent steps to select certain items in a Conversion List to launch a mass-conversion.


Step 1 Open a Conversion List in the editor (see [Editing Conversion Lists on page 90](#)).

Step 2 Select the Conversion Item(s) to convert.

 **Important!** Check that an Object Type has been defined for each of the selected Conversion Items. Refer to [Defining Object Types on the previous page](#).

Step 3 Right-click on the Conversion Item(s) and select **Convert**, then  **Convert the selected items** from the contextual menu.

Step 4 Enter the necessary conversion options in the **Conversion Wizard**.

 **Note:** Because the **Convert Decl. Specification** option is not usable if the **Object Type** is ***NONE**, this option will be automatically deactivated for the conversion of all the items for which those options are in conflict.

A message will be displayed in the log to inform you of this modification.

Reference For more information about the parameters in the wizard, refer to [Executing a Single-File Conversion on page 80](#) and [Managing the Free Form Conversion Options on page 40](#).

Step 5 Click **Finish**.

Result The conversion is launched.

When the conversion is complete, if it was successful, you can open the new source member. If unsuccessful, an error message will display giving details of the error(s).

Reference For more information about consulting converted content, refer to [Understanding Conversion Results on page 94](#).


8.4 Converting an Entire List

Conversion Lists enable you to convert items that are saved in a list and ensure that the converted members share the same library and source. You can either convert all of the items in a list at the same time, or you can select some of them to convert.


Follow the subsequent steps to convert all of the items found in a Conversion List.

Step 1 Open a Conversion List in the editor (see [Editing Conversion Lists on page 90](#)).

 **Important!** Check that an **Object Type** has been defined for each of the selected Conversion Items. Refer to [Defining Object Types on page 91](#).

Step 2 Right-click anywhere in the Editor and select **Convert**, then  **Convert all** from the contextual menu.

Step 3 Enter the necessary conversion options in the **Conversion Wizard**.

 **Note:** Because the **Convert Decl. Specification** option is not usable if the **Object Type** is ***NONE**, this option will be automatically deactivated for the conversion of all the items for which those options are in conflict.

A message will be displayed in the log to inform you of this modification.

Reference For more information about the parameters in the wizard, refer to [Executing a Single-File Conversion on page 80](#) and [Managing the Free Form Conversion Options on page 40](#).

Step 4 Click **Finish** to launch the conversion.

When the conversion is complete, if it was successful, you can open the new source member. If unsuccessful, an error message will display giving details of the error(s).

Reference For more information about consulting converted content, refer to [Understanding Conversion Results on page 94](#).

9 Understanding Conversion Results

Chapter Summary

- 9.1 Comparing Original and Modernized Source Members 94
- 9.2 The Conversion List Editor 94

Each successful conversion uses one Conversion Unit and each Transformer RPG license contains a predefined number of Conversion Units. Licenses do not track which source members have already been converted, therefore running a conversion program on a file once, then modifying something and running the same program again counts as two conversions.

9.1 Comparing Original and Modernized Source Members

If you select **OK** when prompted to open the converted source member after a successful conversion, the new source opens in the editor. Open the original member and put them side by side to compare the transformation.

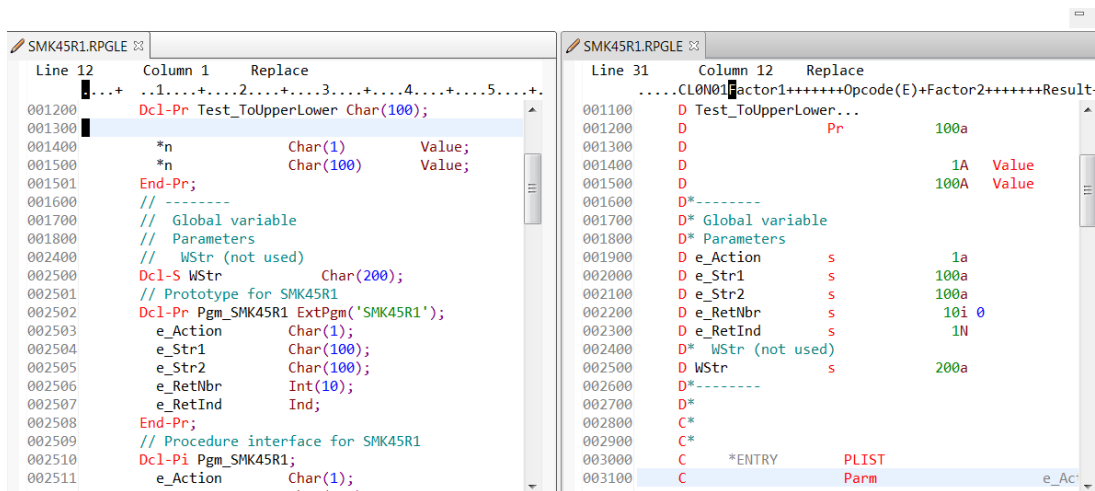


Figure 14: Comparing original and modernized sources

9.2 The Conversion List Editor

To keep a record of your actions, the results of the most recent conversion made to any item(s) are stored in the Conversion Item and displayed in the Conversion List.

The result information consists of four values displayed in the table:

| Value | Description |
|--------|--|
| Status | This is the current status of the item in the conversion process. The possible values are: |

Table 63: The Conversion List Editor




| Value | Description |
|-------------------|--|
| | No Icon = No conversion has been made.  = The conversion succeeded.  with Messages = The conversion succeeded but there are some warnings.  = The conversion failed. |
| Conversion Date | The date the last conversion attempt. |
| Conversion Member | The name of the converted member. Generally this is the same as the original name but the location of the source member is different. |
| Message | This is a summary of the conversion process. The entire message is displayed at the bottom of the editor when a Conversion Item is selected. |

Table 63: The Conversion List Editor

10 Copybooks in ARCAD-Transformer RPG

At the moment when ARCAD-Transformer RPG automatically converts a copybook, it only touches the comment lines and leaves all of the active lines of code intact. This is due to the fact that it uses the RPG compiler to generate the necessary field references. Since a copybook cannot be compiled by itself, Transformer RPG bypasses the conversion of copybooks because of the inherent inability for successful standalone compilation.

What has been suggested to convert the copybook would be to take the copybook and rename it. Then put the copybook along with all of the members that are needed to make up the entire executable, i.e., the program and all the satellite copybooks that are called within it, into a member. Once that conversion of the whole program complex has been successfully undertaken, the portion of code which represented only the copybook could be extracted manually from the overall code and put into its own member to then become a "converted" copy book.

However, the possibility that the copybook contain subroutines, lines of code that initiate with the BEGSR command and terminate with the ENDSB command, may exist. Since ARCAD-Transformer RPG offers an option for converting subroutines to procedures, should the copybook contain any of these subroutines and the conversion is setup to process these routines into external procedures, what was the original copybook containing the subroutine(s) would be replaced with a member containing CALLP commands linked to the procedures which would now exist as procedures carrying the name of the subroutines. Should there be any other programs that referenced the original copybook, those programs would no longer function because the copybook could no longer supply the original code for the required subroutines. This is the primary reason why Transformer RPG does not convert the code in copybooks.

Troubleshooting

CPF9801 – The object xxx in library yyy type *FILE not found

Problem

Before the conversion process occurs, ARCAD-Transformer RPG validates the existence of the destination library and source physical file. If it is not found an error is reported.

Resolution

Verify the destination spelling and retry your conversion.

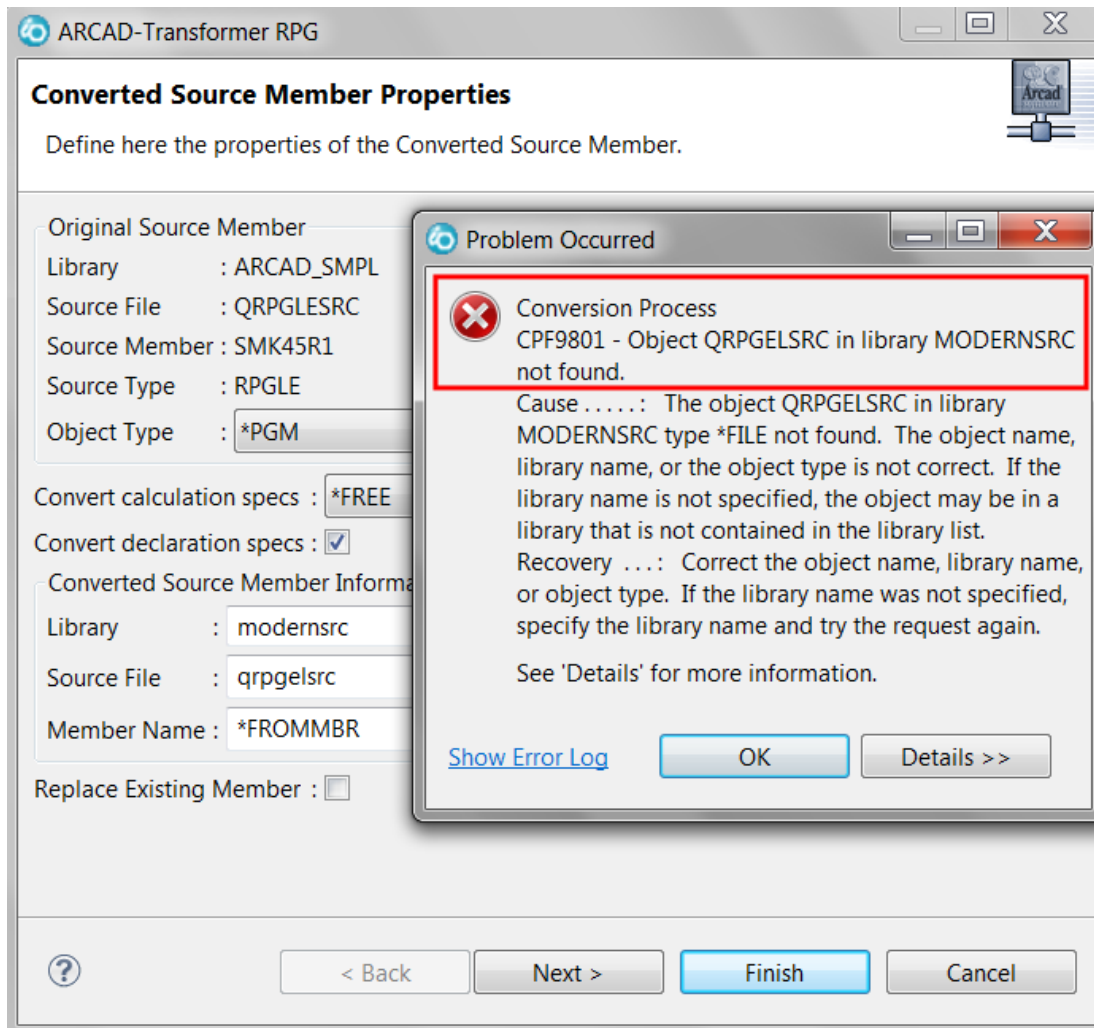


Figure 15: Problem Occurred CPF9801

MSG3542 – Member xxx already exists in source file yyy/zzz

Problem

You are attempting to convert a source member and the destination library / source physical file and member already exist.

1. Resolution 1

Choose a destination with a member name for your converted source that does not already exist.

2. Resolution 2

If you intend to use your destination and member name, select the **Replace Existing Member** checkbox.

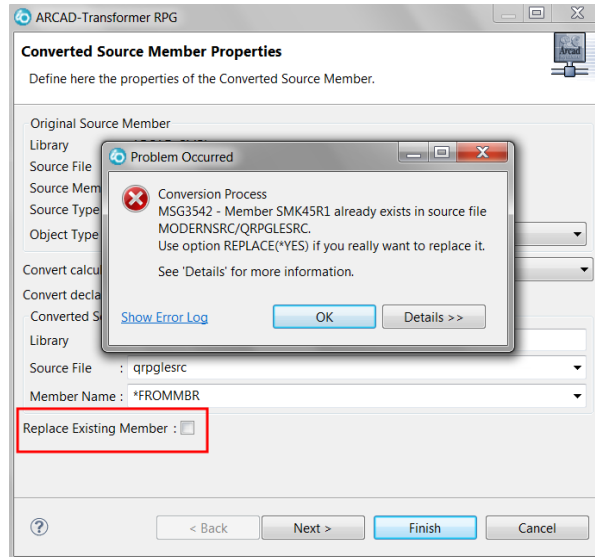


Figure 16: Problem Occurred MSG3542

MSG3579 – The CCSID of the output source file xxx is different from the CCSID of the entry source file yyy

Problem

A mismatch of CCSIDs can potentially cause misconverted or non-compileable source code.

1. Resolution 1

Specify a different target destination to contain your converted source code.

2. Resolution 2

Change the CCSID of the target destination to match your source member that is being converted.

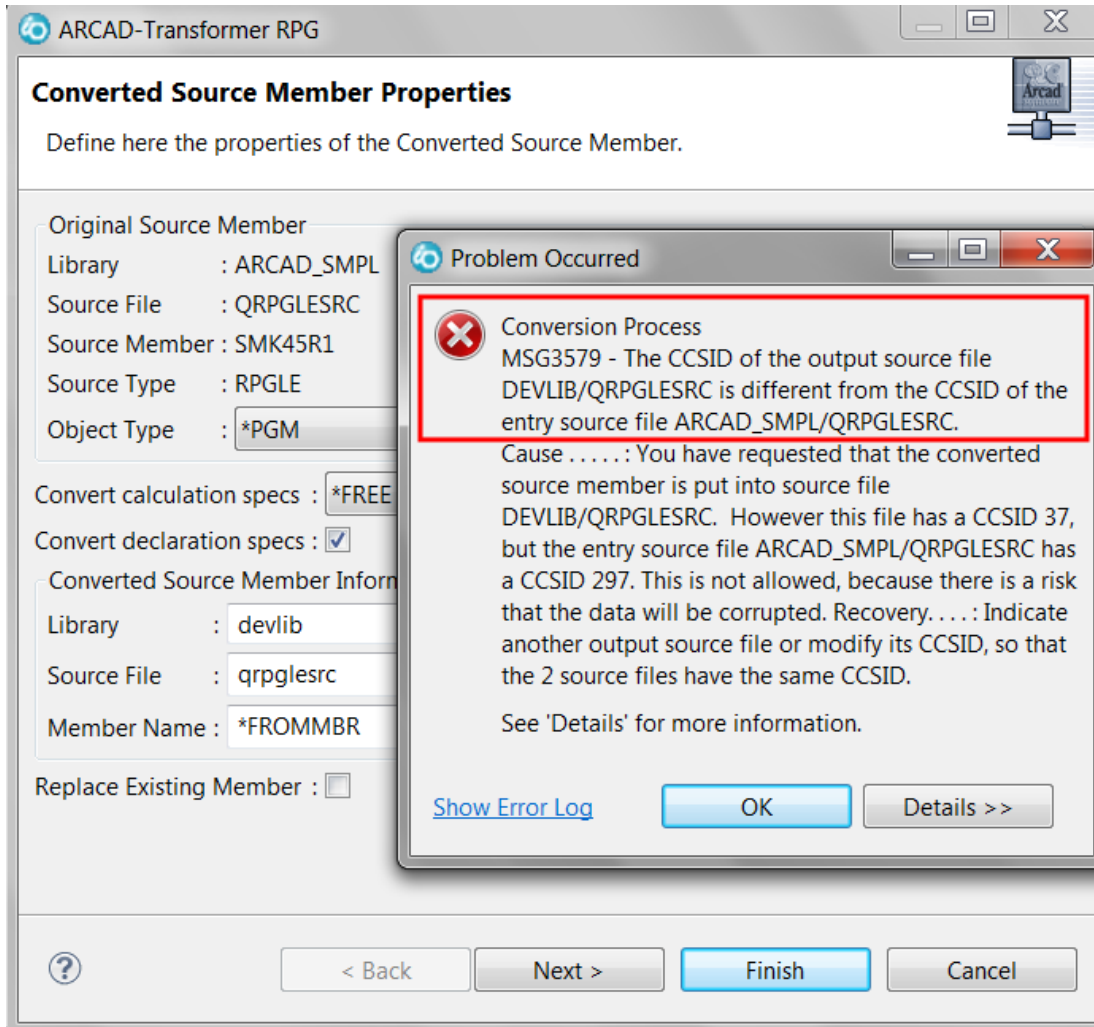
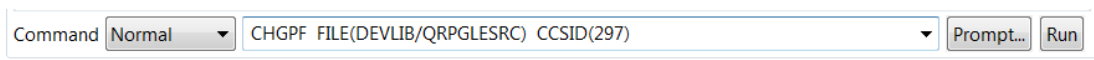


Figure 17: Problem Occurred MSG3579

In this particular example DEVLIB/QRPGLESRC has a CCSID of 37 and ARCAD_SMPL/QRPGLESRC is 237.

If you want to change the CCSID of DEVLIB/QRPGLESRC you will issue a CHGPF command to have it become CCSID 237:



MGR3866 – Error during preparation XRef calculation for xxx: impossible to convert to Free

Problem

A cross reference must be generated in order for Transformer RPG to convert your source code.

The most common cause is that the runtime environment is not prepared properly. See below for specific error messages.

There are several ways to identify why a source member did not convert.

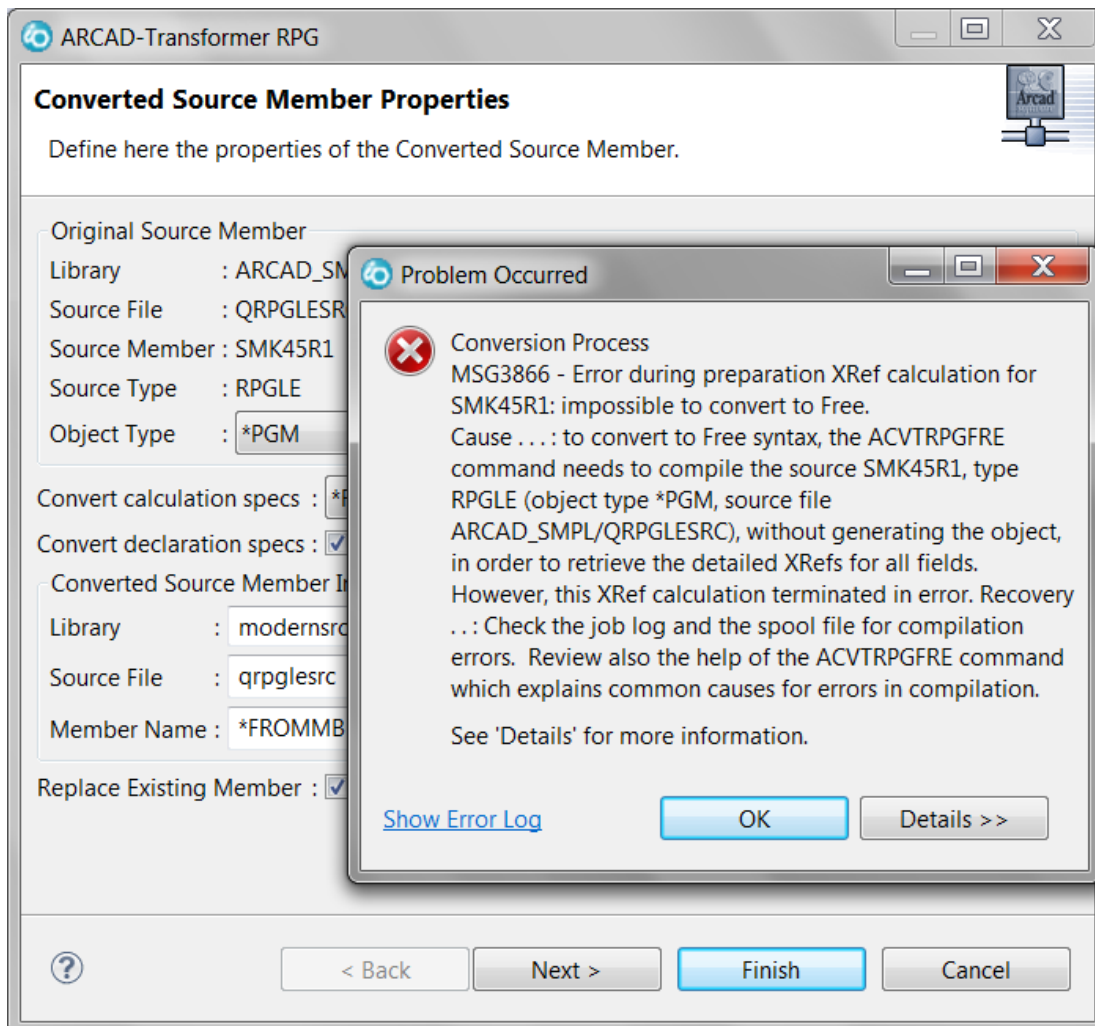


Figure 18: Problem Occurred MSG3866

1. Resolution 1
Incorrect library list – see [Create a Library List on page 71](#) and then retry
2. Resolution 2
Review the RDi **Error Log** view for error messages. If the **Error Log** view is not present on your perspective a quick way to restore it is using RDi's Quick Access. Quick Access is located near the tip of the perspective. In this space type **Error Log**. When the **Error Log** view appears in the search results click on it.

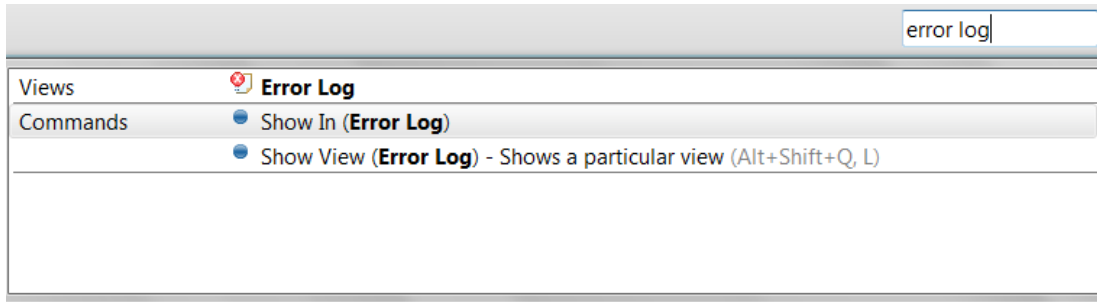


Figure 19: Display Error Log view

Look for the most recent entries in the error log. These will show in detail the causes for the conversion problems.

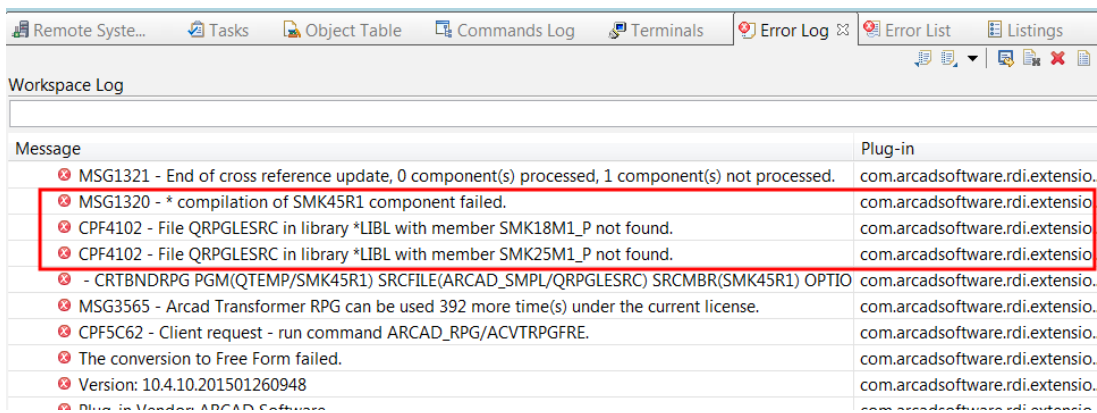
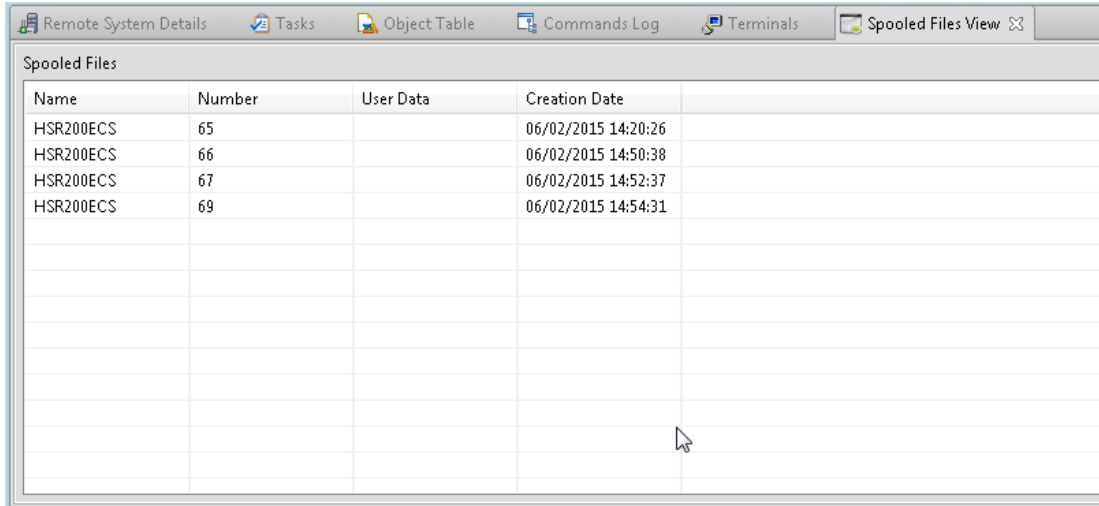


Figure 20: Error Log entries

3. Resolution 3

Read the spool file that is created during the conversion process. The option to view spooled files is presented automatically if a single-file conversion fails:



| Name | Number | User Data | Creation Date |
|-----------|--------|-----------|---------------------|
| HSR200ECS | 65 | | 06/02/2015 14:20:26 |
| HSR200ECS | 66 | | 06/02/2015 14:50:38 |
| HSR200ECS | 67 | | 06/02/2015 14:52:37 |
| HSR200ECS | 69 | | 06/02/2015 14:54:31 |

Figure 21: Spooled Files View

Spooled files can also be accessed in RDi from the **Spooled Files** filter.

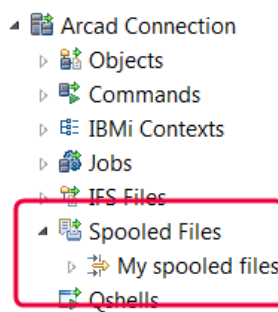


Figure 22: My spooled files

- a. Open **My spooled files** and see *all spooled files for your user ID. If there are not a lot of entries this is a reasonable approach. For ARCAD-Transformer RPG, the printer file will be the same name as the program being converted. In this example we have been using source member SMK45R1. Right click on **Spooled Files** and select **New** to create a new filter. Next, select **Spooled File Filter...**

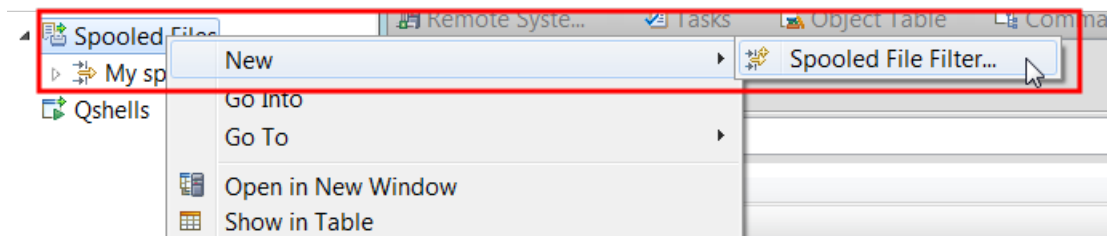


Figure 23: Spooled File Filter

- b. Leave the user ID as *CURRENT. Type the program name in the **Spooled File Name** field. Then click **Next**.

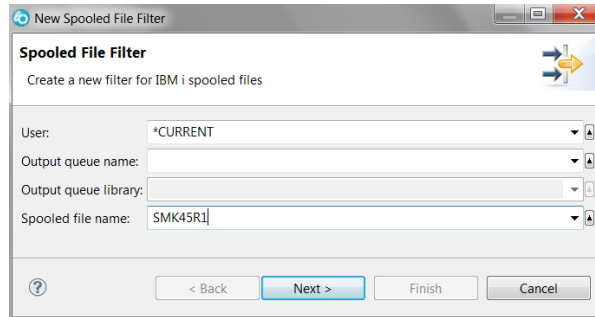


Figure 24: New Spooled File Filter 1

- c. You can accept the name of the filter created by RD*i* and click **Finish**. This will create a filter entry in the **Spooled Files** list.

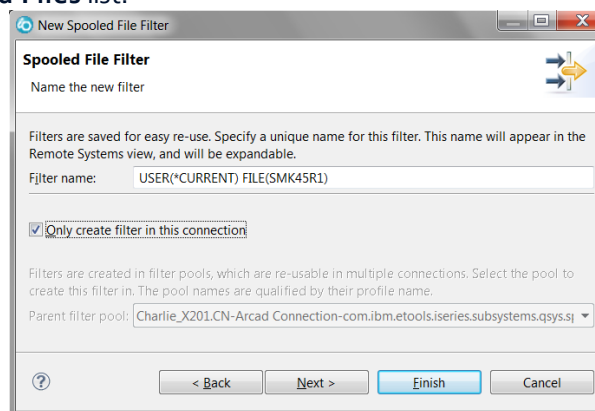


Figure 25: New Spooled File Filter 2

- d. When you expand the list you will see *all spooled files related to this source member. Right click on your selected file and RD*i* will allow you to view the spooled file.

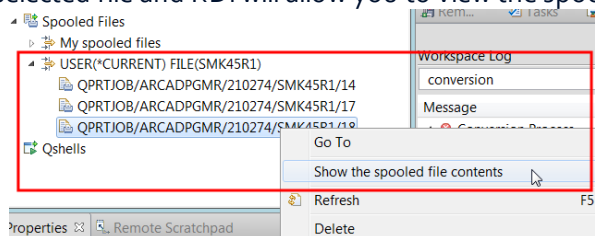


Figure 26: Show the spooled file contents

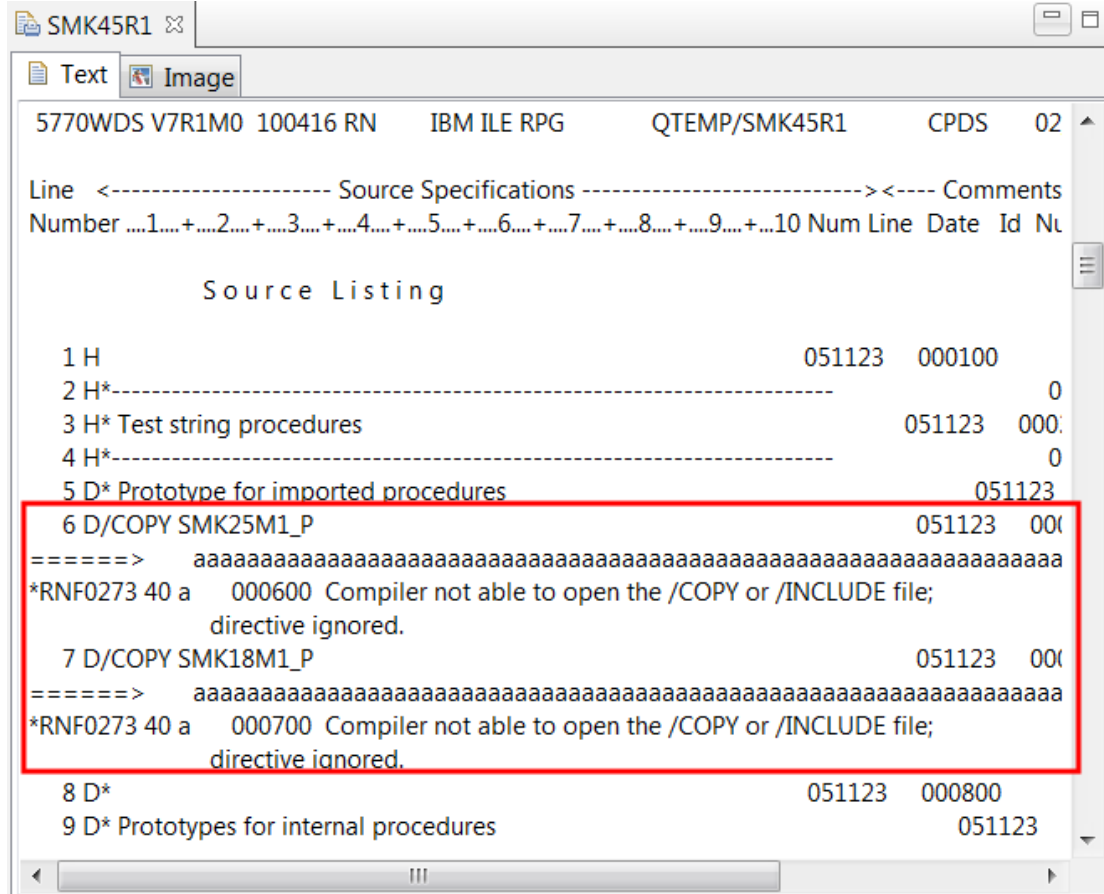


Figure 27: Spooled File Contents

Compatibility with the ARCAD-Server

ARCAD RCP client applications, such as ARCAD-Extract, ARCAD-Verifier, ARCAD-Client or any other Eclipse-based studio, include a consistency version control. This control is based on the file *compliant.xml*.

You may have troubles connecting an ARCAD RCP to the ARCAD-Server after the ARCAD-Server is upgraded from one version to the next. The blocking message in red below may appear when you attempt to connect to the server.

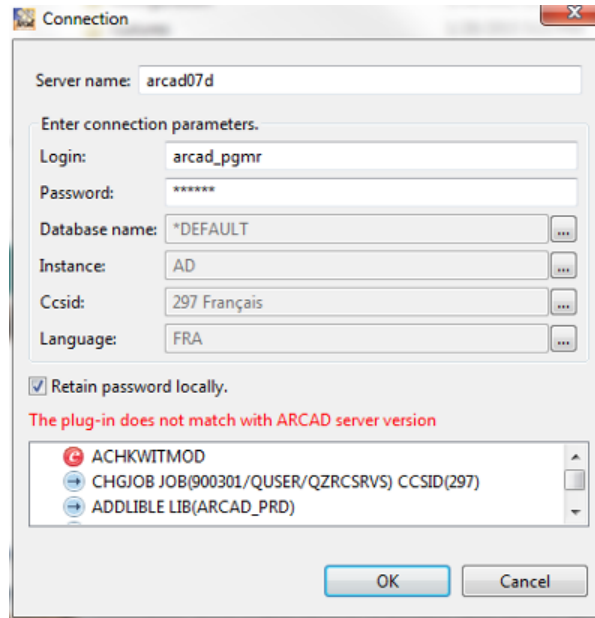


Figure 28: Server Compatibility Error

If you receive this error, install a new version of the ARCAD RCP application, if a new version is available.

If no updated version of the RCP application has been released, and after carefully checking the compatibility aspects in the release notes, update the *compliant.xml* file as described below.

Updating the compliant .xml

Warning This procedure should only be done in emergency situations. It is not recommended to use this as a permanent solution to server-/studio compatibility problems.

The *compliant.xml* file is located in the following directory:

`<installation>/plugins/com.arcadsoftware.core_<version>/compliant.xml`.

Where **<installation>** is the RCP application installation directory and **<version>** is the highest version level in the plug-in directory.

Follow the subsequent steps to update the *compliant.xml* file.

Step 1 Check the existing version of the ARCAD-Server on the IBM i using the following command:

```
dspdataara arcad_prd/arcversion
```

Step 2 Open the *compliant.xml* file for modification.

Step 3 Add a line based on one of the following examples:

- To be able to connect to a server running ARCAD version "aa.bb.cc":
<version major="aa" minor="bb" release="cc" match="&PERFECT;"/>
- To be able to connect to a server running ARCAD aa.bb.whatever the release number:
<version major="aa" minor="bb" release="XX" match="&PARTIAL;"/>

In this last case, a warning message will inform you that compatibility with the ARCAD version on the server is only PARTIAL.

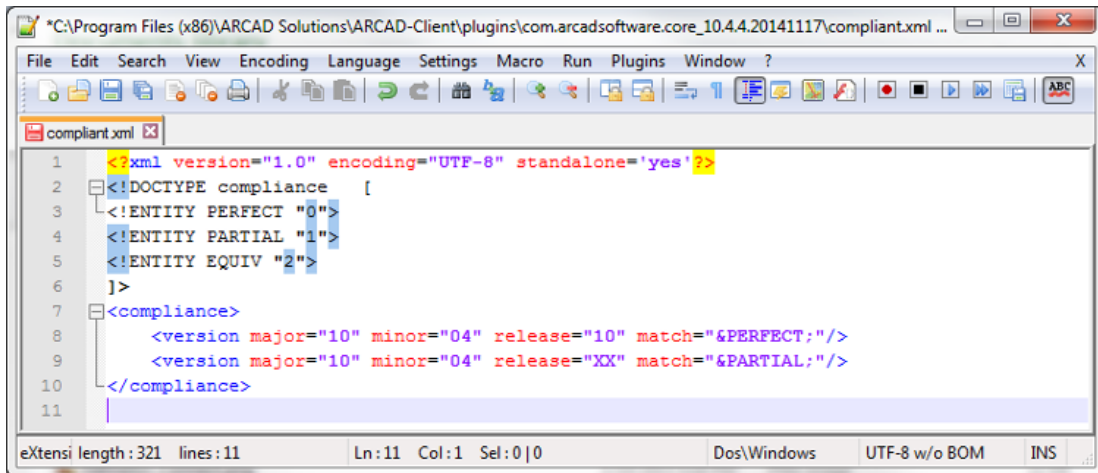


Figure 29: Editing the compliant.xml

F.A.Q.

For more frequently asked questions, refer to [our website](#).

Does the ARCAD-Transformer RPG RDi plug-in work in the 5250 interface?

The product is designed to use as an RDi PlugIn. The product can be used from a 5250 interface, but in that case, you must use the `ARCAD_RPG/ACVTRPGFRE` command, which converts one member at a time.

It is not possible to perform mass-conversions using the 5250 interface unless other ARCAD products are installed.

To apply the license key using a 5250 session use the command `ARCAD_RPG/ALICCVTRPG`.

Does ARCAD-Transformer RPG handle conversion of MOVE or MOVEL?

The conversion of MOVE or MOVEL op codes are handled. See [About the Conversion Operations on page 22](#).

This operation code, widely used in traditional syntax, performs operations for which the behavior depends on the type of the variables and their length; all of the following cases are covered:

- Figurative constant in factor 2 (*Blank, *Zero, *Hival, *Loval, *ALL'o', *ALL'xxxx').
- With or without (p) as operation extender.
- Variable or fixed length field.
- Factor 2 field with a length less than the result field.
- Factor 2 field with a length greater than or equal to the result field.
- Assignment with numeric conversion alpha using %XLate, %Dec, %EditC, and possibly digital shifting by multiplying or dividing by 10, 100, 1000 etc...
- When factor 2 and/or the result field contain a Date/Time/TimeStamp field, conversion using %Date, %Time, %TimeStamp, or %Char, %Dec using the date/time format of the field.

If specified in positions 71-76, indicators are set after the converted instruction.

Does ARCAD-Transformer RPG handle OCL to CL?

No. The converter only converts ILE code to FREE!

- A -

Activation Key 38

- C -

commands

ACVTRPGFRE 14, 16-17, 43, 86, 107

ADDLIB 72

CRTDUPOBJ 68

CRTLIB 66

OVRDBF 18, 54

Conversion Engine 5, 15-16, 36

Conversion Item 88, 91, 94

Conversion List 77, 80, 88, 91, 94

Conversion List Editor 75, 88, 94

Conversion List View 75, 88

Conversion Process 15

Conversion Unit 94

- F -

Free syntax 16-17, 19, 24, 42, 82

- L -

library 16, 63-65, 68, 70-71, 77, 81, 87-88, 92-93, 97

ARCAD_SMPL 64, 68, 72, 99

library list 71, 77, 100

List Header 75, 88

- M -

menu

Objects 67, 69-70, 72

Work with members 70

- O -

operation code 18-19, 22-23, 43, 107

Conversion Operations 22

- R -

Remote Systems 15, 64-66, 68, 70, 72, 77

RPGLE 16-17, 22, 63

- S -

source member 15, 18, 36, 40, 63-64, 69, 72, 77, 80, 88, 91-95, 98