

# Three AS/400 Technologies vs. One Problem

Tech tip courtesy of [Barsa Consulting, LLC](#) and [Dave Schnee](#)

The following text and the attachments illustrate three different methods of solving the same problem. How to convert a number (possibly including a fractional part) from decimal to a range of other number bases. While this problem is not a "burning issue" for most of us, we all ought to know how to use one or two of these non-RPG technologies.

The solutions are written in REXX, C and JAVA. The REXX and C programs are from 1994, the JAVA class is from February, 1999. All 3 work on the AS/400. They have a few differences, though. The REXX and JAVA programs can use arbitrarily long numbers (the REXX program asks the user "how many digits", the JAVA program tries to make an internal choice), the C program is limited to 8-byte floating-point numbers, but it will execute much faster than either REXX or JAVA.

		<b><u>View and/or Download</u></b>
REXX	Every AS/400 ever made	<a href="#">cvtbasen.rex</a>
C	Any AS/400 at V2R3 or later. Needs a "C" compiler (OPM or ILE) to create the program.	<a href="#">cvtbasen.c</a>
JAVA	Any AS/400 at V4R2 or later.	<a href="#">cvtbjava.zip</a>

All three of these solutions also work on other platforms:

REXX	Most IBM mainframes and any (IBM or equal) PC with OS/2
C	(most of the computers in the world)
JAVA	(most of the computers in the world)

The method for installing and making these solutions usable differ, though, even on one platform. For the AS/400:

REXX	copy the source file to a source physical file member (record length at least 112 bytes because of how I coded it). Use it, as is, via the STRREXPRC command naming the source file and the member you copied it into.
C	copy the source code to a source physical file member (record length at least 92 bytes). Compile it with the CRTCPGM or CRTBNDC command (just like any other AS/400 "create program" command). Call the program you have created.
JAVA	This one is different. Read the rest of this document:

A JAVA "class" is the bytecode stream file that represents the function you compile. The compiler (JAVAC) is supplied as part of a JDK (JAVA Development Kit) for your platform (There are many versions of these, many levels, many platforms. Two of them come from Sun, the owner of the language, for free. IBM supplies one with OS/400). I have supplied both the source code (baseconversion.java) and a compiled class file (baseconversion.class). The source file is for you to look at. The class file is platform-independent and will run on the AS/400, your PC (if you have a JRE -- a JAVA Runtime Environment, also supplied free from Sun), many Unix systems today, etc.

To use it on your AS/400, copy the class file to an IFS directory accessible to the AS/400 (Client Access/400 will help here). Run it using the RUNJAVA command from your "regular" command line or start QSHELL (use command QSH), navigate to the directory with the class file and enter: "java baseconversion" (note: do not supply the "class" extension). For the examples below, I copied the baseconversion.class file to directory "Javastuff" in directory "Dave" in the "home" file system of the IFS.

To run this from your command line (V4R2 or later, please), looks like:

```
RUNJAVA CLASS(baseconversion) CLASSPATH('/home/Dave/Javastuff')
```

The rationale for this is that the RUNJAVA (or the JAVA) command needs just the class file name (without the extension) as the CLASS parameter, but needs to be able, through the CLASSPATH parameter to locate the class file.

To run this using QSHELL (V4R3 or later, please), looks like:

```
QSH
    (and then, once in QSHELL):
$
> cd /home/Dave/Javastuff
$
> java baseconversion
```

The rationale for this is that the JAVA command needs just the class file name (without the extension) as the thing to run, but needs to be able, through the current directory to locate the class file.

You could also compile this (or other JAVA code) on your AS/400, but that is beyond the scope of this discussion. To do that, you would use QSHELL (the UNIX-like command line that IBM provides with no help and virtually no commands, as of V4R3) to execute the JAVAC compiler. To do this, execute the QSH command and feel as helpless as you did the first time you saw a DOS command line and had no idea what commands there were. The difference is that, in DOS, there **were** commands if you could find them (including one called HELP). In QSH, there are only a very few commands in V4R3 -- there are at least a few dozen more to come in V4R4.