

Packaging z/OS Open Source (and other) Software for Electronic Distribution

Learn how to package z/OS software from Lionel Dyck. This technique will also work for any package that you want to distribute electronically. Although there are many ways to package your code, this article focuses on one specific technique that Lionel finds helpful.

By Lionel B. Dyck

A close-up photograph of a hand holding a CD-ROM. The CD is held between the thumb and index finger, with the rest of the hand supporting it from below. The CD's surface is highly reflective, showing a vibrant rainbow spectrum of colors. The central hole of the CD is visible, and the text 'By Lionel B. Dyck' is printed in black on the upper portion of the disc's surface.

Did you know that there has been open source software available in the mainframe environment since nearly day one? This started with the 704 and 705 (and probably before) programmers shared their code, including the source, first on card decks (remember those 80 column cards?), then on tape individually and in collections. Eventually code was shared via the World Wide Web and CD-ROM. I remember from 1977 the SHARE MVT Mods Tapes. Later came the Connecticut Bank and Trust tape (created and maintained by Arnold Casinghino on tape and now maintained by Sam Golob and Sam Knutson on tape, CD-ROM, and on the Web). Today you will find collections of open source tools (and toys) from individuals and organizations. I even have contributed my own source to the CBT virtual tape (<http://www.cbttape.org>) and have posted on my own Web site (<http://www.lbdsoftware.com>). This sharing of open source is not unique to z/OS, as there is also a long history of open source sharing in the z/VM community.

The reason for this article is because I've been asked how to package code for sharing. This technique will also work for any package that you want to distribute electronically. There are a lot of options available to you to package your code. This article will present a technique that I have used for many of the packages that I've shared.

BUILDING THE PACKAGE

The assumption is that your package has multiple elements. That being the case, the first thing that I will do is to create a partitioned data set (PDS) using RECFM=FB and LRECL=80.

In this PDS, I create a member I name \$DOC. This member contains a brief description of the package and a description of the other members of the PDS that will be found in it when we are finished building the package.

Let's assume that the tool you wish to share has several REXX Execs and some ISPF Panels.

For the Execs and Panels, we will use both the TSO Transmit and SMCOPY commands to combine all of the individual members into a member of the PDS. See FIGURE 1 for a sample REXX Exec to make the desired package.

The TSO Transmit command, which has an alias of XMIT, is used because it will convert one or more members of a partitioned data set, as well as a sequential data set, into a format that will retain the original contents including directory information

and characters which could be corrupted during a file transfer. The SMCOPY command is used to copy the resulting data set created by the TSO Transmit command into a member of our package partitioned dataset since the TSO Transmit command is restricted to creating a sequential output data set.

If the package contains a large number of elements, you may want to create unique partitioned data sets for each element type, in which case, the TSO Transmit command demonstrated here does not require the MEMBERS option, as the default is to include all members of a partitioned data set.

First the TSO Transmit command to create a temporary sequential data set:

```
XMIT X.Y DS(rexx.exec) OUTDS(temp.xmit)
MEMBERS(one two three) NOLOG
```

Where:

- ▼ X.Y are used as a generic nodename.userid for TSO Transmit, which requires a nodename.userid. Since the output is to a data set, this field can be anything you want.
- ▼ *rexx.exec* is the name of the REXX Exec library where the REXX members are.
- ▼ *temp.xmit* is the data set name of a temporary data set that will be created by the XMIT command
- ▼ *one two three* are the member names of the REXX members

Next the SMCOPY command to copy the temporary sequential data set into a member of the PDS:

```
SMCOPY FDS(temp.xmit) TDS(pds(EXEC))
NOTRANS
```

Where:

- ▼ *temp.xmit* is the data set name of the temporary data set created by the XMIT command
- ▼ *pds* is the data set name of the partitioned data set
- ▼ NOTRANS is required to prevent SMCOPY from doing any form of translation on the copied file.

At this point, repeat these two commands for the Panels library and members with a member name of PANELS in the PDS.

The packaging is now almost complete. You should now have three members within your PDS: \$DOC, EXEC and PANELS, with the EXEC and PANELS members in TSO Transmit format.

FIGURE 1: SAMPLE PACKAGE MAKE REXX EXEC

```
/* rexx */
xmit x.y ds(rexx.exec) outds(temp.xmit) members(one two three) nolog
smcopy fds(temp.xmit) tds(pkg.pds(exec) notrans
delete temp.xmit
xmit x.y ds(ispf.panels) outds(temp.xmit) members(one two three) nolog
smcopy fds(temp.xmit) tds(pkg.pds(panels) notrans
delete temp.xmit
xmit x.y ds(pkg.pds) outds(pkg.xmit) nolog
```

FIGURE 2: SAMPLE RECEIVE REXX EXEC

```
/* this rexx exec is expected to be invoked by the EXEC command
```

```
thus:
```

```
Exec 'dsname(RECEIVE)'
```

```
Prompting will occur for hlq and optional volser.
```

```
and it will then issue the TSO RECEIVE command for the
the following members to create these new data sets:
```

Member	Dataset
EXEC	hlq.EXEC and
PANELS	hlq.PANELS

```
You will then need to copy these datasets/members into
datasets for production use.
```

```
After these data sets are created several of the members
of this install data set will be browsed.
```

```
*/
parse source x1 x2 x3 x4 dsn .
x = PROMPT("ON") /* enable prompting */

say "Enter desired hlq for target data sets",
  "(default is" sysvar("syspref").package.NEW):"
pull hlq
hlq = strip(hlq)

if hlq = "" then hlq = sysvar("syspref").package.NEW

say "Enter optional volser for target data sets",
  "default is to allow standard allocation to",
  "find a volume."
pull volser
volser = strip(volser)

if volser <> "" then vol = "vol("volser")"
else vol = ""

Say "Using HLQ:" hlq
if vol <> "" then
  Say "Using Volser:" volser

queue "dsn('hlq'.exec)" vol
"Receive inds('dsn'(exec))"
queue "dsn('hlq'.panels)" vol
"Receive inds('dsn'(panels))"

if sysvar('sysispf') <> "ACTIVE" then exit
Address ISPEXEC
  "Browse dataset('dsn'($doc))"
  "Browse dataset('dsn'(changes))"
```

At this point, I like to create a member with additional documentation if the tool is moderately complex and another member I call CHANGES to document the change history

for the package (assuming that you will be making changes to this package over time and sharing the changes as you shared the original package).

Next, I recommend the creation of a RECEIVE member (see FIGURE 2), which those planning to use the package can use to rebuild the EXEC and PANELS libraries. This RECEIVE member is a REXX exec that will prompt the user for a high-level qualifier for the EXEC and PANELS libraries and an optional volser. It will then issue the TSO RECEIVE command for both the EXEC and PANELS members to create EXEC and PANELS partitioned data sets. After doing the RECEIVE processing, the \$DOC and CHANGES members are browsed using ISPF Browse if the RECEIVE member was executed under ISPF. The RECEIVE member and usage should be documented in the \$DOC member of the PDS so that the receiver of the package will know that it is there and how to use it.

The last step in packaging is to convert the PDS into a sequential data set that can be e-mailed or posted on a Web site. This is done using the TSO Transmit command thus:

```
XMIT X.Y DS(pds) OUTDS(package.xmit)
NOLOG
```

Notice that in FIGURE 1 that the Make REXX Exec executes all of the necessary

commands to build the find package in TSO Transmit format.

The *package.xmit* now should be downloaded to your workstation in binary format and can then be zipped to reduce the file size and then e-mailed or uploaded to a Web site. I also recommend that you consider submitting your package to the CBT site (<http://www.cbttape.org>) for sharing with a worldwide audience.

USING THE PACKAGE DATA SET

When someone receives your package, they have to do a few things to use it.


First, they must unzip the file if it was received in zip format. Then the xmit file must be uploaded to their z/OS system in binary format into a sequential data set allocated with RECFM=FB and LRECL=80. FIGURE 3 demonstrates using the FTP command from a workstation to upload the file.

After the file is uploaded, the user must then use the TSO RECEIVE command and convert the xmit file back into the package partitioned data set:

```
RECEIVE INDS(upload.data.set)
```

FIGURE 3: SAMPLE FTP UPLOAD

```
C:>cd download
C:\download>ftp zos.host
User: enter userid
Password: enter your password
ftp>bin
ftp>quote site recfm=fb lrecl=80
ftp>cd temp
ftp>put package.xmit
ftp>quit
```

Then the user should read the \$DOC member and execute the RECEIVE member. 

NaSPA member Lionel B. Dyck is a z/OS systems programmer for a large health maintenance organization in California. He has been in systems programming since 1972 and is an active participant at SHARE. lionel.b.dyck@kp.org is his e-mail address.

Supporting Enterprise Networks and Operating Environments

SUPPORT