

BY LIONEL B. DYCK

Installing ISPF Applications the Easy Way

The documentation that comes with many ISPF applications can present a maintenance challenge. Using the techniques presented here, you can simplify this process.

MANY ISPF applications come with installation documentation that instructs you to add their panels, skeletons, etc., to existing ISPF libraries or to add their libraries into the existing ISPF DD allocations. This can present a maintenance challenge with lots of application libraries to be allocated, especially if you have multiple TSO Logon PROCs that must be maintained and updated. This article presents a very simple approach to installing applications.

Each ISPF application comes with some or all of the libraries shown in Figure 1. The approach that I'm going to suggest is to use the LIBDEF facility to define these libraries before the application is invoked. LIBDEF will work for most of these libraries but you'll have to use ALTLIB for any CLIST or EXEC libraries, and for LOAD libraries you will have to check the documentation or call the vendor to determine if the application will work with a LIBDEF for ISPLLIB. If you are unable to LIBDEF the LOAD library, then the alternatives are (1) to place the load module library into the Linklist, (2) use a STEPLIB DD, or (3) use a dynamic STEPLIB tool if you have one (e.g., Tone Software's Dynastep).

GETTING STARTED

The first thing you must do is to create a REXX EXEC or CLIST that will be the starting point for invoking the application. For purposes of this article we'll use a nonexistent application called ARTICLE. This application is shipped with the libraries shown in Figure 2. The process will be fairly easy, but there are some "gotchas" in this application. We will create the invocation command in REXX in the remainder of this article.

1. To start with, we must begin a REXX EXEC with a comment line with the word REXX in it. Since a comment can span multiple lines we can include more than just a brief comment.
2. Next, we use our first command Arg options as shown in Figure 3. This captures any parameters that may need to be passed to the application when it is executed. This way, the parameters are saved in the REXX variable options and are available for later use.
3. The next step is to code the commands to set up the ISPF environment for the application. See Figure 4. The **Address ISPEXEC** sets the REXX environment to ISPEXEC so that any commands are by default routed to the ISPEXEC platform for execution. The **Vget Zapplid** invokes the ISPF VGET services to retrieve the current ISPF application pool name. This **Zapplid** value is then tested to see if it is ART, and if not (which is what we expect when the exec is first entered), then we set up the ISPF environment for the application.

The first Libdef defines the ISPTLIB file for ISPF. This is required because, if you remember, this application has its own command table that resides in ARTICLE.TABLE. For any command table to be active for an ISPF application it must be defined before the **Select** service is invoked.

4. Next, we use the ISPF **Select** service to invoke the command recursively. What I mean is that by using the **sysvar(sysicmd)** function you can invoke

this REXX EXEC regardless of what name you call it when you install it into your production REXX EXEC library. We pass the initial parameters, **options**, by including that variable in the command. The **NEWAPPL(ART)** sets the ISPF environment to the application name of **ART**, and the **Passlib** is used to allow the **ISPTLIB Libdef** to be passed to the command being invoked. After the application is invoked with

the **Select** service, it will return to this code immediately after the **Select** and we can then free the **ISPTLIB Libdef** by invoking the **Libdef** command with only the **Ddname**.

Note: At this point, if you find that you are unfamiliar with the ISPF services I am mentioning, you can either trust me (something I wouldn't recommend) or you can open BookManager and read the *ISPF*

Services Guide. Why do I say not to trust me? Because you need to understand what you are doing and just taking my word for it won't teach you anything. This article can only address a small subset of the circumstances involved in installing an application into ISPF and thus can only provide you with a starting point. By familiarizing yourself with the IBM ISPF publications you will arm yourself with the ability to deal with the other challenges that you will face when installing applications under ISPF. Just point your web browser at <http://booksrv2.raleigh.ibm.com/ispf/ispf.htm> and you will be able to download and read the IBM ISPF publications in Adobe Acrobat (PDF) format. Or, if you want the more "traditional" BookManager format you can find the ISPF publications along with the rest of the OS/390 softcopy publications at www.software.ibm.com/ad/ispf.

The first thing you must do is to create a REXX EXEC or CLIST that will be the starting point for invoking the application.

The following commands set up the ISPF environment to invoke the application shown in Figure 5:

- ◆ The **Address TSO** is used to inform the REXX environment that the next command is a TSO command (instead of an ISPF command). The **ALTLIB** command performs a dynamic addition to the SYSEXEC allocation for the REXX EXEC library that is part of this application.
- ◆ The **Libdef** commands define the other ARTICLE datasets to be used by this application. The Libdef is like a temporary addition to the specified (e.g., ISPLMLIB) existing allocations.
- ◆ The **Select Pgm** command actually invokes the application that is started by a program named ARTICLE. The **Parm** is used to pass the parameters that we saved at the start of this code for use here.

Figure 1: Set of Libraries for Each ISPF Application

Library	DDName	Description
CLIST	SYSPROC	Contains one or more TSO CLISTS
EXEC	SYSEXEC	Contains one or more TSO REXX EXECs
LOAD	ISPLLIB or STEPLIB	Contains the load modules for the application
MSGS	ISPMLIB	Contains messages
PANELS	ISPLLIB	Contains the ISPF Panels (the visual part of the code)
SKELS	ISPSLIB	Contains the ISPF Skeletons used for File Tailoring
TABLES	ISPTLIB	Contains the ISPF tables and/or special command table

Figure 2: ARTICLE is Shipped With the Following Libraries

Library	DDName	Description
ARTICLE.EXEC	SYSEXEC	The REXX EXECs used by the application
ARTICLE.LOAD	STEPLIB or ISPLLIB	The load modules that are the heart of the application
ARTICLE.MSGS	ISPMLIB	The ISPF Messages issued by the application
ARTICLE.PANELS	ISPLLIB	The ISPF Panels that are the visual interface to the application
ARTICLES.SKELS	ISPSLIB	The ISPF Skeleton used during file tailoring
ARTICLES.TABLE	ISPTLIB	The ISPF Command Table (ARTCMDS) which special commands used during the application execution

Figure 3: Arg Options

```
/* REXX Procedure to invoke Application ARTICLE without requiring
   the application libraries be allocated in the LOGON PROC */
```

```
Arg options
```

Figure 4: The Commands to Set up the ISPF Environment for the Application

```
Address ISPEXEC
"Vget Zapplid"

If zapplid <> "ART" then do
  "Libdef ISPTLIB Dataset ID('hlq.ARTICLE.TABLE')"
  "Select cmd('sysvar(sysicmd) options') NewAppl(ART) Passlib"
  "Libdef ISPTLIB"
  Exit
End
```

Figure 5: Commands to Set up the ISPF Environment

```
Address TSO "Altlib Activate Application(EXEC)",
           "Dataset('hlq.ARTICLE.EXEC')"
"Libdef ISPLMLIB Dataset ID('hlq.ARTICLE.MSGS')"
"Libdef ISPLLIB Dataset ID('hlq.ARTICLE.PANELS')"
"Libdef ISPSLIB Dataset ID('hlq.ARTICLE.SKELS')"
"Libdef ISPLLIB Dataset ID('hlq.ARTICLE.LOAD')"
"Select Pgm(ARTICLE) Parm('options')"
```

Figure 6: REXX Commands to Release LIBDEF

```

“Libdef ISPLLIB”
“Libdef ISPSLIB”
“Libdef ISPPLIB”
“Libdef ISPMLIB”
Address TSO “Altlib Deactivate Application(EXEC)”
Exit

```

Figure 7: Completed REXX EXEC to Invoke the ARTICLE Application

```

/* REXX Procedure to invoke Application ARTICLE without requiring
   the application libraries be allocated in the LOGON PROC */

Arg options

Address ISPEXEC
“Vget Zapplid”
If zapplid <> “ART” then do
  “Libdef ISPTLIB Dataset ID(‘hlq.ARTICLE.TABLE’)”
  “Select cmd(“sysvar(sysicmd) options”) NewAppl(ART) Passlib”
  “Libdef ISPTLIB”
  Exit
End

Address TSO “Altlib Activate Application(EXEC)”,
           “Dataset(‘hlq.ARTICLE.EXEC’)”
“Libdef ISPLLIB Dataset ID(‘hlq.ARTICLE.MSGS’)”
“Libdef ISPPLIB Dataset ID(‘hlq.ARTICLE.PANELS’)”
“Libdef ISPSLIB Dataset ID(‘hlq.ARTICLE.SKELS’)”
“Libdef ISPLLIB Dataset ID(‘hlq.ARTICLE.LOAD’)”
“Select Pgm(ARTICLE) Parm(“options”)”
“Libdef ISPLLIB”
“Libdef ISPSLIB”
“Libdef ISPPLIB”
“Libdef ISPMLIB”
Address TSO “Altlib Deactivate Application(EXEC)”
Exit

```

Note: Using this technique for heavily accessed applications will result in increased system overhead. This is because the LIBDEF facility uses dynamic allocation which is not a “cheap” service. You must decide if the extra overhead is worth the simplification in the production installation. If not, then you have to update and keep current every TSO LOGON procedure each time you change the dsnames for a product.

- ◆ After the **Select Pgm** command that processed the application returns to this REXX code, we then free the allocation overrides and return via the **Exit** statement. See Figure 6.

Figure 7 contains the complete REXX code for what we have just written.

OK, we have completed the code for invoking the application ARTICLE without having modified any LOGON PROC or updated any ISPF library. But we are not done yet. This REXX EXEC must now be installed into a REXX EXEC or CLIST library that your users have allocated to their TSO sessions. Then you can tell your users to invoke this application by entering **TSO %ARTICLE** on the ISPF command line.

A better approach would be to create an ISPF menu panel for the users. To do that

you would update the IBM-provided ISR@PRIM (the IBM ISPF Primary Option Menu) with a menu option for your local application menu. Then, on that panel have an option to invoke the ARTICLE application. This is beyond the scope of this article, but you can find information on how to modify the menu in the IBM ISPF publications. IBM provides a helpful technical publication on customizing the ISR@PRIM menu at <http://booksrv2.raleigh.ibm.com/ispf/tr292043.htm>. You can find more information about ISPF at www.software.ibm.com/ispf. At this site, you will even find IBM freeware for ISPF including a tool called TASID, which is where ISRDDN comes from but which provides a lot more information than ISRDDN does.

Some variations on this code may be necessary if the application is shipped with

load modules. Using a dynamic steplib such as Dynastep from Tone Software, you could replace the Libdef ISPLLIB commands as shown:

To add the dynamic steplib:

```

“DS Add(‘hlq.ARTICLE.LOAD’) Front
Nomsg Push”

```

To remove the dynamic steplib:


```

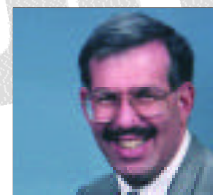
“DS Nomsg”

```

If you don’t have a dynamic steplib, then you can use the new **TSOLIB** command from IBM *before* entering ISPF. This isn’t always easy to do so I wouldn’t recommend it.

Alternatively, you can place the load library into your STEPLIB concatenation in your TSO LOGON PROC or put the library into the Linklist. If your application uses a CLIST instead of a REXX EXEC then you can easily change the ALTLIB from **Application(EXEC)** to **Application(CLIST)** and all will work just fine.

Hopefully, this article has given you a good starting point for installing both homegrown and vendor purchased applications into your ISPF environment. If you are a vendor that ships a product that runs under ISPF, please consider providing an entry CLIST or EXEC along these lines for your customers. 



NaSPA Member Lionel B. Dyck is a lead MVS systems programmer at a large HMO in California. He has been in systems programming since 1972 and has written numerous ISPF dialogs over the years including an FTP dialog that was highlighted in the December 1997 issue of *Technical Support*. He is an active member of SHARE and can be contacted at Lionel.B.Dyck@kp.org.

©1998 Technical Enterprises, Inc. For reprints of this document contact sales@naspa.net.