Confused and Confounded? The ALCS macro libraries explained

by Ian Worthington

There is a lot of confusion regarding which ALCS macro datasets should be used for application assemblies and in what order. Hopefully this article will bring some clarity to this situation.

This article was originally written as an internal memo to our customer to explain our decisions regarding the various dataset concatenations we use for program assemblies and generations but, given experiences installing and customising ALCS at different sites, it may well be of interest to a wider audience.

BACKGROUND

Prior to ALCS 2.1.1, all of the ALCS macros, be they for the programming interface, recoup, system generation, or monitor assembly only, were shipped to the customer in a single dataset. A second dataset contained the macros associated with the IPARS "sample application".

In an attempt to bring some clarity to the issue of which macros were legal to use in what situations, and in order to meet EU directives on clearly defined programming interfaces, ALCS V2.1.1 was shipped with 4 macro libraries:

Library	Description
DXCMAC1	Application programming macros forming part of the GUPI (general-use programming interface: that part of the programming interface guaranteed by IBM to be stable across product releases).
DXCMAC2	Macros associated with the IPARS sample application.
DXCMAC3	Macros required for ALCS generation (and monitor reassembly).
DXCMAC4	Macros for Recoup, monitor user exits, and application programming macros forming part of the PSPI (product sensitive programming interface: that part of the programming interface which may change from release to release, or even as a result of the application of service, possibly necessitating reassembly, or even recoding of your application programs).

Although this was a well-intentioned effort, it was never very clearly explained and by itself appears to cause more confusion than almost anything else!

WORKING WITH THE MACRO SETS

In addition to the libraries listed above, you will have an application macro library, and one or more MVS system macro libraries (typically SYS1.MACLIB, SYS1.AMACLIB, SYS1.MODGEN, and SYS1.AMODGEN, depending upon your MVS level and how your MVS system programmers have chosen to install MVS).

In order to prevent system problems it is important to tightly control the availability of macro function to appropriate parts of the organisation. For example, a naïve application programmer with previous MVS programming experience could decide to use MVS I/O facilities from within an ecb-controlled program leading to severe ALCS performance problems. Even experienced application programmers sometimes make use of MVS facilities that are potentially dangerous and definitely not recommended, such as WTO, the incorrect use of which can hang your MVS system.

It is also important that application programmers are prevented through accident, or well-intentioned naivety, from usurping macros that are part of the IBM-supplied macro set. This is most easily avoided by choosing a concatenation sequence that places the application programmer-controlled macro dataset(s) *after* the IBM-supplied datasets. (This assumes appropriate RACF access controls are in place for the IBM-supplied datasets, of course!)

This does not, of course, prevent application programmers from creating macros with the same names as IBM-supplied macros that are not included in the assembly concatenation sequence for the associated programs. Nor does it prevent the creation of macros with names that might exist in future releases of ALCS. Only enforcing the reserved name rules specified in the product manuals, preferably by encoding them within your library management system, can prevent both of these problems

MACROS .v.TASKS

We can divide ALCS assembly tasks into the following groups:

Normal application programs:	Those programs using only ALCS GUPI.
Special application programs:	Those programs making use of the ALCS PSPI, including Recoup descriptors.
ECB user exits:	Special user-coded ECB-controlled programs invoked by ALCS.
Monitor user exits:	Special user-coded programs invoked by the ALCS monitor.
System generations:	Assembly of tables required to configure ALCS.

There is some debate about the utility of separating programs that make use of PSPI macros from the vast bulk of application programs that use only the GUPI as:

• It tends to orphan off individual programs into a library separate from the rest of a functional package (which may not be a problem if you use a library management system that presents a single user view, but could be if you use simple PDSs to store your program base). You may be able to avoid this problem by moving all components of a package into the PSPI library.

• There are some macros in the PSPI library set that are extremely useful and woefully under-utilised. Likewise your application may have a significant dependency upon one or more PSPI macros. In this case you need to evaluate your program base and decide if the costs of restricting the use of this part of the macro set outweigh the advantages. Alternatively it may be possible to create a separate dataset containing those parts of the PSPI you wish to permit your application programmers to use, and make the updating of this dataset a part of your regular PTF application proce dures.

Personally I feel the segregation of the application program base has been beneficial for those applications for which I have had involvement. Your mileage may of course vary.

For each of these tasks, the recommended libraries and concatenation sequence is as shown in the following table:

Libraries	**Normal Application Programs**	**Special Application Programs**	**ECB User Exits**	**Monitor User Exits**	**System Generations**
DXCMAC1	Required	Required	Required	Required	Required
DXCMAC2	Omit	Omit	Omit	Omit	Omit
DXCMAC3	Omit	Probably required	Probably required	Probably required	Required
DXCMAC4	Omit	Required	Probably required	Required	Not required
SYS1.MACLIB SYS1.MODGEN etc.	Omit	Omit recommended	Omit recommended	Probably required	Not required
Application macro libraries	Required	Required	Required	Probably required	Probably required

In this table:

Required: This library is required for the assembly of this item.

Probably required: This library may be required, depending on what your programs do.

Not required: This library is never required.

Omit recommended: You almost certainly wish to exclude the use of macros in this library in this type of program.

Omit: You definitely wish to exclude the use of macros in this library in this type of program.

NEVER DXCMAC2?

In the above table, you may have notice that I suggest you *never* include DXCMAC2, the library containing the IPARS sample application, in your assemblies. So what's it there in the product for?

The IPARS sample application is just that: a *sample* application, used in practice for distribution of the Message Switching application and other bits-and-pieces. But it also contains lots of components that you are not going to use, and will almost certainly cause naming conflicts with your own application.

If you require components from the IPARS sample application then, it is far better to adopt these as your own and copy them into your own application library. The concerns about service do not apply as strongly here as you probably don't wish to perform a full applications test after applying ALCS service anyway. It is advisable though to keep an eye on what service is being applied to the IPARS FMID as there are occasionally changes (both to the sample application and to other sample components, such as user exits) that you will wish to incorporate into your program base.

ODDS and SODS

There are a handful of components that cause organisational confusion all by themselves. This section discusses these. In general the rule for maintaining a stable and easily serviceable ALCS system is that there must not be, ever, any datasets in front of the IBM-supplied macro datasets within an assembly concatenation. Any changes you require to IBM-supplied ALCS product code (as opposed to the IPARS sample application distributed as a separate FMID (DXCMAC2, DXCSRC2, DXCIWE1, DXCHDR2, etc.) should be applied to the SMP/E-controlled libraries as USERMODs.

Any ALCS macro components positioned ahead of the IBM libraries nullifies the integrity checks SMP/E performs to ensure that service is not being applied to components that have been locally modified.

Also, in order to ensure that ALCS is easy to service and upgrade, the number of USERMODs applied should be kept to an absolute minimum. Some customers have appropriately mandated executive sign off for each and every USERMOD in order to curb the tendency of their programming staff to make ad-hoc "improvements" to ALCS.

There are however a number of supplied components that *must* be modified by end-users in order to configure ALCS:

Library	**Component**	**Comments**
DXCMAC1	DXCSER	Global access serialisation macro. Used by application programs to serialise access to information in the application global area
	RTCEQ	Available for TPF compatibility to define additional functional support console names for use with the WTOPC service.
DXCMAC2	DXCURID	These components are all part of the IPARS sample application and so can be copied to an applications library and modified as required.
	DXCUSAL	
	GLxBx	
DXCMAC3	DXCZCUSR	Communication generation user data validation macro. Used by the communications generation to validate and convert data to be placed in the communications user data area.

The fact that DXCSER and DXCZCUSR are in DXCMAC1 and DXCMAC3 is unfortunate. They need to be distributed within the ALCS FMID (in order that ALCS can be assembled without dependency upon another FMID), but would have been better located within their own PDS.

There are two options available for modifying these components:

1. Take copies of these into a dataset at the head of the assembly macro concatenation. Whilst this is a safe thing to do as these are strictly user components, it does however open up an exposure that other macros that people wish to modify will find their way in there as a matter of expediency.
2. Apply the required modifications as SMP/E USERMODs. Whilst this is undoubtedly an inconvenience, it is almost certainly the lesser of the two evils and, after the initial installation phase, they will not change very often.

Ian S. Worthington is an independent ALCS and MVS systems consultant currently retained by British Airways Speedwing on behalf of Avianca in Bogotá, Colombia. He is always interested in working on new projects and can be contacted via email at IanWorthington@usa.net, or via fax through his virtual offices in New York at (+1 212 208 4463) and in London at (+44 20 7900 2916).