

# Managing Macro Development

Betsy Eggleston  
Minna Popkin

Harvard College Library Technical Services

NAAUG  
7 June 2005

## Table of Contents

<b>I.</b>	<b>Why Use Macros.....</b>	<b>3</b>
<b>II.</b>	<b>Select Software to Use.....</b>	<b>4</b>
<b>III.</b>	<b>Licensing Issues .....</b>	<b>5</b>
<b>IV.</b>	<b>Distribution of Macro Software .....</b>	<b>5</b>
<b>V.</b>	<b>Upgrades to Macro Software.....</b>	<b>6</b>
<b>VI.</b>	<b>Who Decides What Processes to Automate.....</b>	<b>7</b>
<b>VII.</b>	<b>Who Designs and Writes Macros.....</b>	<b>7</b>
<b>VIII.</b>	<b>Training in Writing Macros .....</b>	<b>9</b>
<b>IX.</b>	<b>Designing Macros .....</b>	<b>9</b>
<b>X.</b>	<b>Documentation.....</b>	<b>10</b>
<b>XI.</b>	<b>Testing Macros .....</b>	<b>10</b>
<b>XII.</b>	<b>Distributing Tested Macros.....</b>	<b>12</b>
<b>XIII.</b>	<b>Training and Support in Macro Use.....</b>	<b>14</b>
<b>XIV.</b>	<b>Revising Macros .....</b>	<b>15</b>
<b>XV.</b>	<b>Communication and Feedback.....</b>	<b>15</b>
<b>XVI.</b>	<b>Policy Development .....</b>	<b>16</b>
<b>XVII.</b>	<b>Personnel Changes .....</b>	<b>17</b>
<b>XVIII.</b>	<b>Evaluation .....</b>	<b>17</b>

## **I. Why Use Macros**

### ***Possible Gains from Automation with Macros:***

#### **Time Efficiencies**

Our experience has been that running macros with Aleph doesn't save a significant amount of time in the discrete process of a single task. This may be due to the fact that we have been very conservative and cautious in our macros, including many steps to ensure that the focus is in the right place when text is entered and that the correct button is focused when we issue a command to click the button. We do, however, notice time-savings in training of student workers and temps, in the performance of tasks in less than optimal conditions, that is, when the user is interrupted, and in that there are fewer errors to correct.

#### **Ergonomics**

We see lessening the number of mouse clicks as a major benefit of using macros. Many of the tasks that we have automated have been simply to avoid the need to shift between the keyboard and the mouse. Several staff members have been able to receive and do copy cataloging for as many as 950 items per month since we instituted these macros. There have been no disability claims during this time.

#### **Consistency of Data**

Macros written at any level can help maintain quality data. Even ones that type "Includes bibliographical references" are useful in that they spell every word the same every time. If you are able to make complex macros, the gains can be even higher. One of our concerns when we migrated to Aleph was that, for the first time, monograph catalogers at Harvard were using the MARC 21 Format for Holdings. We are a very large and decentralized institution, and HCL Technical Services, the largest of approximately 30 technical services units adding records to HOLLIS, has 80 staff members. It was a very difficult task to train all staff who were cataloging monographs to use the new format and to comply with the guidelines that had been established. Using a macro that ensured that holdings fields were entered for multivolume works, that indicators were set consistently, and that fixed fields were changed in response to bibliographic format has ensured that our unit's holdings records have been consistent from the beginning and can be depended upon to produce the displays that users and reference librarians can read and understand. This macro even reads the subfield c of the 300 field and sets the Folio designation in the holdings record correctly, an area that had been a real problem before the macro.

### ***Automating Local Procedures***

Aleph offers a great many customizable features. In general, we have tried not to make macros for anything that can be customized in Aleph. Difficulty can arise with the implementation at a large decentralized institution where exigencies of various units necessitate different workflows and there are few basics that apply to all. We weren't able to take advantage of the Aleph feature that allows the LC call number from the bibliographic record to populate the 852 field of the holdings record because such a wide variety of classification schemes are in use around the university. We use a macro to transfer the call number from the bibliographic record to the holdings record. In acquisitions, the number of funds available for book purchases in various units dictates local workflow to some extent, as does the number and size of approval plans. The fact that HCL Technical Services works with over 100 endowed funds, each with its own 33 digit code made a macro to type fund numbers in order records a very useful tool for us. We have also been using a macro for acquiring approval plan material, due to the fact that the complexity of HOLLIS has made loading vendor supplied acquisitions records a problem for us.

### ***Fixing Aleph Bugs: Ex Libris Fix vs. Macro Workaround***

It is important to judge when you are going to be able to get an Aleph idiosyncrasy fixed as a bug through Ex Libris and when the staff time lost in dealing with whatever problem you have is going to be greater than the time it would take to create and distribute a macro workaround. The problem with Cut, Copy and Paste is a good example of a bug that might be easier to deal with via a macro.

## **II. Select Software to Use**

### ***Features to Look for***

#### **Ability to Use Logic**

This will allow more flexibility in the type of processes that you can automate. While a good deal of repetitious work goes into cataloging and acquisitions, processes very often include decision-making based on information already contained in the record. If/Then statements that utilize that information allow the macro to make some of those decisions quickly. If/Then statements also can prevent errors by only allowing the macro to run if the conditions are right.

#### **Ability to Deal with Controls**

To a large extent, Aleph depends on Windows Controls for user input. These are the boxes where you type in text, buttons that you click, etc. Any macro program that you use with Aleph should be able to recognize and act on Windows Controls, unless you are only using it to type text.

#### **Provision of Pause and Timing Controls**

Macros tend to run as fast as the CPU will let them. In many cases, Aleph server response time, the speed at which the screen can receive text, and the speed at which a user can input data will be much slower. Any macro program that you choose should include commands that wait for the machine to type text, for the server to respond to commands, and for users to make choices or to wand in or type barcodes, call numbers, or other variable text.

#### **Ability to Deal with System Response Time**

Ideally, a macro program will be able to wait an indeterminable amount of time while the Aleph client reacts with the server (and to stop waiting immediately when Aleph is ready to accept more input). It may be that only a built-in macro program would be able to get precise information from Aleph about when it is ready for more input, but a stand-alone product that can at least read the outward signs that a process is done, for example, that focus has changed to a different button or that a window has appeared or disappeared, will be much more useful in creating efficient macros than one that doesn't.

#### **Unicode**

Aleph version 15 and higher use Unicode text encoding. If you want macros to manipulate text and re-enter it in Aleph and your records contain many diacritics, your macro program should be Unicode compliant. We have gotten around the fact that Macro Express is not Unicode compliant by using Visual Basic scripts run by Macro Express macros to process bibliographic record text.

#### **Forms for User Prompts: Multiple Choices in One Prompt Window**

Some macro programs allow you to define "forms", windows to get user input. While it is not essential that the software you choose allows the user to input more than one piece of information at a time, it can be very useful and increase the macro's speed simply because all user input can be gotten in one step.

#### **How It Works with Other Software**

No matter what software program you choose, be sure that it will interact well with Aleph, first and foremost. Other desirable features include the ability to load and run other programs, or to make use of Perl scripts, Java scripts or Visual Basic applications.

## **Web Pages That List Macro Software**

- <http://tucows.mundofree.com/win2k/macros2k.html> Tucows provides evaluation of software and links to download sites
- <http://www.sharewarejunkies.com/cgi-bin/search/search.pl?Terms=macro> Sharewarejunkies.com provides reviews and links to download sites

## **Conclusion**

There are no macro software products that I know of that include all of these features. If you want to do a comparison of products that you find on the web, create a spreadsheet listing the desirable features in one matrix and the product in the other.

The Harvard libraries use Macro Express with our Aleph client. It is somewhat limited in that it is not Unicode compliant and it doesn't provide a way to create customized forms. We have supplemented to some extent with VB scripts run by macros. Most of our references in this handout are to Macro Express and to situations that arise due to its constraints. As this product is used widely with Aleph, and as several other macro software products present similar issues, we feel comfortable using it as our main example.

## **III. Licensing Issues**

### **Single vs. Multi-User Licenses**

If you are just starting to work on macros, you may want to obtain trial copies of different macro software programs to try them out and compare them. Companies usually offer a free demonstration version of their software for a limited period of time.

When you have selected the software program that you want to buy, find out what kinds of license options are available. Usually companies offer some kind of multi-user licenses in addition to the single one-user licenses. Multi-user licenses usually come with just one installation package and a right to install the software on a certain number of workstations. Installation must be centrally coordinated or managed.

### **Pricing**

Some software companies offer different pricing options. A one-time purchase of a single-user license does not usually include any future updates. However, many companies offer discounts on software updates to existing license holders. You can also inquire about annual licenses or maintenance agreement which may include technical support from the software company and possibly any software upgrades that are released while your agreement stays valid.

Some companies give discounts on limited-scope licenses. For instance, "academic pricing" for educational institutions is usually lower than the list price. You may even be able to negotiate special pricing if the license limits the software use to the ILS only.

## **IV. Distribution of Macro Software**

### **Who Installs the Program and Troubleshoots Software Problems**

Institutional support is a requirement for a successful implementation of any macro program. Responsibility for installing the program should lie with whoever is most knowledgeable about the various pieces of software on department computers, ideally the IT department. This should ensure that when computers are upgraded or hard drives are replaced the macros that you have come to depend on will still be available. It will also help avoid a situation where the one person for whom macros were a "pet project" leaves and there is no one to continue support

for the program. In order to ensure continuity, macro program installation should be assigned to a position rather than to a person.

Troubleshooting, both of the macro program and the individual macros, will often need to be done by expert users or by whoever wrote the particular macro, as this requires a high level of familiarity with exactly what steps are occurring in the workflow. While troubleshooting can be done by any expert user, it is important that there be expert users who have been assigned the responsibility for this task and that it be considered part of their job.

### ***Installation Configuration Decisions***

Several of the macro software packages that are available consist of a macro editor and a macro player. In some cases the macro player can be installed on its own. Since there are some security issues with macros, it may be desirable to limit the number of staff members that have access to the macro editor. Macro Express can be distributed in such a way that the entire macro software package is installed on all users' computers or the editor and player can be installed for macro developers, and the player only to end users.

### ***Any Additional Software, Such as Perl, VB***

Macro Express can be made more robust by having it run scripts compiled in Visual Basic or Perl. One consideration here is that scripts created with Visual Basic can be compiled as .exe files and run on any computer; only one VB installation is needed. Perl must be installed on every computer where its scripts are run.

## **V. Upgrades to Macro Software**

### ***Assuring that Everyone is Using the Same Version***

While it would be great if we could assume that any macros written in an earlier version of the software would be playable with a newer version, it isn't true. If staff members are going to be sharing macros, it is best if they are all running the same version of the macro software. We work in an environment where IT can push software down to our computers. It either happens overnight, if we leave our computers on, or the next time we turn them on. If you don't have a similar situation, it is important to plan a systematic approach to installing the new version, (perhaps a macro set to run on startup?) as software version is a definite factor in troubleshooting.

### ***New Features, Fixes to Bugs***

The biggest impetus for upgrading your macro software is to take advantage of new features and fixes to bugs. It is also good to be running the latest version of the software, in that that is the version most likely to be supported by the software's vendor. However, versions come out often enough so that the time it will take to upgrade the software and test your old macros will not warrant installing every update issued by the vendor. Any upgrade will need to be evaluated thoroughly to assure that the time saved by the new features is greater than the amount of time expended to update existing macros.

### ***Revisions of Existing Macros***

Whenever you consider an update to your macro software, you should download a trial copy and test it with each of the macros that is in use. Occasionally, bug fixes will render a macro command that you might have used idiosyncratically, actually taking advantage of the bug, unusable. After revising all macros and creating a new macro file for use with the upgrade, test the upgrade and the new file on at least one other computer in the department, choosing one that is set up as differently as possible from the original test machine, to ensure that if there are changes that respond differently to different computer environments, they have been addressed.

## **VI. Who Decides What Processes to Automate**

### ***Analyzing and Prioritizing Needs***

Analyzing and prioritizing needs should start at the management level. If there are a large number of potential users doing a variety of jobs, this may need to be done by a committee, probably with input from someone familiar with the macro software's potential. It will be a good time to do a thorough analysis of the workflow in order to cut down on inefficient steps. Writing good macros can be expensive so it is important to make sure that the process is cost effective. It can also be a good time to standardize workflow across units, if possible, since it is more efficient to write one macro for a process than to write several that achieve the same goal.

After agreement is reached on the optimal workflow, individual tasks can be enumerated and considered in light of whether or not there are elements of these tasks that a macro can perform. Using a macro program like Macro Express, a great many decisions can be made using IF statements, either taking the condition from the record on the screen, files on the hard drive, or from user input via dialog boxes. Hence, almost any task where the computer has access to the information needed to make the right decision on what to do next, is a candidate for automation, as well as those that need a minimum of user input. Deciding which tasks these are will need to be done by a person familiar with what commands are available in the macro software and with hands-on experience with Aleph functions. Ideally this decision-maker will also be experienced with the workflow of the department and with the local and national standards for the task that the macro is automating or will work closely with someone who is.

### ***Getting User Input***

Whoever makes the initial decision on what macros need to be written, user input is important. It is only someone who has cataloged 1000 books in the Aleph Cataloging module who can tell you what are the most frustrating, time consuming, and boring parts of the job. These should be the first candidates for macros. In addition, paying attention to end-user needs is essential to having macros accepted and used. If the macro as created does not fit in with the user's perception of how the task is done, she won't use it. If getting the macro to run properly is more trouble than doing the job by hand, she won't use it.

Unless the macro writer is also doing the job that the macro is being written for, she should take the macro idea to an end user early in the process.

## **VII. Who Designs and Writes Macros**

### ***Avoiding Duplication of Effort***

It should be made clear from the beginning that time spent in writing macros is valuable time and that efficiency is as important in macro writing as it is in the general workflow. In a setup where one person has been assigned responsibility for writing all macros, this duplication effort is not of an issue, but what happens more often is that the macro software is purchased with the idea that all staff members will be able to write their own macros. Except in very rare cases, this will not happen. In large units two or three people are likely to emerge as having an interest and talent for macro writing. Once the writers have self-identified, the next step should be to establish good communication between or among them so that each knows what the others are working on. In addition, each can be assigned a different area of responsibility, perhaps in one of the following ways:

- One person designs the macros, another codes them, a third writes documentation and is responsible for their distribution and testing
- One person only writes macros to be used in cataloging, another only writes them for acquisitions, another for serial check-in
- When a macro writer begins a project, he/she posts the title and description to a shared space or website so that others can share information they might have on such a project and avoid the same project

## ***Taking Advantage of Native Talent vs. Hiring a Programmer***

Our experience has been that by far the best option is to hire a programmer who is interested in becoming a librarian. We were able to do that because of a funding windfall. Before that, we were in the position that I described above where a few experienced staff members had emerged as macro writers. The advantage of depending on local talent is that it is more likely that macros will be appropriate for the workflow and that if they need tweaking, the writer will understand why and will be more likely to understand and sympathize with users than if the macro was written by someone that doesn't know the work. The down side of using native talent is that there is one less person to get the department's work done, and it may be that the staff member who is good at macro writing is working at a salary above that which you might pay a beginning programmer. More will be said later about looking at ways to estimate saving or return on investment for macros, but it is a good idea to start with such an estimate in order to decide who can be assigned to this task without a net loss of productivity. In some cases programmer time is available from IT or Systems offices. Unless the programmer is willing to spend time getting user input, or has experience in cataloging, acquisitions, serials, or whatever department the macros are being created for, this option may also be frustrating and result in a loss rather than a gain in efficiency. However, it must be remembered that the time that staff spend learning to create macros is not free. The efficiency of macros created by trained programmers can far outstrip that of those created by self-trained library staff because the training time is much shorter and often techniques that the programmer knows can be employed to make macro run faster and more reliably.

## ***Centralized vs. Distributed Effort***

The advantage of a centralized macro writing effort is that it is more efficient. You have one person or a small group of people in close association learning to use the macro writing software and solving the problems of how to use macro commands with Aleph. Time doesn't need to be spent over and over again learning the same tasks. In addition, the approach to programming and distributing macros is more likely to be consistent.

The disadvantages of this approach are that each task that needs to be automated will have to wait for the centralized macro writer to schedule time to address it, centralized macro writers can't be as experienced in all of the tasks that they are writing macros for, and they will be less likely to be available for troubleshooting and problem fixing.

One way of addressing this problem is to develop a network of local experts. These experts would be responsible for understanding how the macros and the macro software works and would be able to modify macros created centrally, but would not need to be a highly trained as the person directly responsible for writing macros. They can also advise the macro writer on workflow that meets local needs.

## ***Local vs. Common Needs***

When there is an Aleph issue that affects the entire university library staff, your first thought should be to ask if this is something that can be customized in Aleph tables. If not, writing a macro that is generic enough to meet everyone's needs may be the best thing to do.

You can even write macros that will serve a wide variety of users by using variables in place of local codes and creating shell macros for local units that supply values for those variables and then run the generic macro. This is described in more detail in the "Technical Aspects" handout.

Alternatively, if you have a network of local experts, they can be given responsibility for local customization.

## ***It's an Addictive Time Sink for Some People***

It would be misleading to imply that designing and writing macros is always quick and easy. Writing macros has a great deal in common with playing computer games and for some people, it is possible to become "addicted" and display the following symptoms:

- All work time is consumed by Macro Express
- Macros are never ready to be used, because they are always being improved
- Falling behind in regular work
- Working late into the night perfecting macros
- Coming in on weekends to work on macros

Supervisors should be wary of giving staff *carte blanche* to work on macros, balancing the cost of the time spent in creating them with the cost of the time they are saving. If you are essentially using one staff position for creating macros, the money you save by the efficiency of the macros must exceed that staff person's base salary plus benefits. It is wise to consider whether the person who will be assuming this job is capable of balancing it with other responsibilities and of containing this part of the job so that the cost doesn't outweigh the benefits.

## VIII. Training in Writing Macros

Having a formal training program for writing macros is one way to shorten the amount of time it takes writers to get up to speed. Having a training program assumes, however, that someone is available to be the instructor. I don't know of anyone who can be hired to do this.

I believe that most libraries embark on using macros the same way we did, with some interested staff members teaching themselves how to do it by first using the macro recorder, viewing the resulting macro in the macro editor, and improving on the recorded macro by exploring the commands and help screens available in the program and then experimenting to see what works in Aleph. It is best to do this experimenting in a training/testing version of the database, of course.

In developing a curriculum for training program, I would follow a similar model. In addition, since macro software that does more than just type text strings often involves at least rudimentary programming skills, I would have as a prerequisite for such training, a basic programming course. If you are using Macro Express, you can also take advantage of the new manual, *Macro Express Explained*, by Joseph Weinpert. You can order it from the Macro Express website <http://www.macros.com/macexplained.htm> for \$49.95. There is also a Google User Group named Macro Express for Aleph Users where you can post queries and get help from your peers.

## IX. Designing Macros

### ***User Input***

The first step in actually writing a macro is to get user input on the following:

- Is automating this task a real need?
- Where should this macro start?
- What errors is it likely to encounter?
- Exactly what are the steps in performing this task, including closing windows, hitting <Enter>, typing text, choosing menu options?
- What decisions need to be made?
- Which choices are the most likely (should be set as defaults)?

### ***Supervisor Approach***

Supervisors will also need to be interviewed to vet the proposed workflow and to establish standards for the work to be accomplished and for the accuracy of the macro.

### ***Input from Cataloging Standards and Local Policies and Practices***

Macro writers should have access to all documentation of standards that are required, AACR2, LC Rule Interpretations, MARC formats, and local decisions. They should refer to the documentation often to ensure that the macro's input in records meets those standards or they should work with an expert in the field. Using macros can be an efficiency disaster if they are, over and over again, doing things wrong.

### ***Vetting the Design***

In most institutions there are technology experts and there are cataloging experts or acquisitions experts or circulation experts. It is wise to have one of these area experts review the initial macro design for adherence to standards before writing the macro.

## **X. Documentation**

### ***Help for Macro Users***

Macro users should be provided documentation that they can use both as a learning tool before they start using macros and as a reference tool later. General macro documentation should include instructions on how to open and close the macro program, how to stop a macro that is running, and how to interpret common error messages, as well as general trouble-shooting help.

Documentation should also be attached to each macro, explaining – in commonly understood terminology – what the macro is supposed to do, when and where it can be run, and how to run it. Clearly stated assumptions will help the end-user understand what conditions must be met and what steps must be taken before running the macro.

If macros are distributed to other work units that might have different workflow and local policies, it should be made clear which work unit or workflow the macro was written for.

### ***Document Each Macro***

Each macro should carry documentation with it, either on comment lines (Remark command in Macro Express) or in separate notes (Notes tab in Macro Express). This documentation should include all the information intended for the end user, but also more technical information, such as other macros called or external files and scripts used by this macro and variables used by this macro. Inserting comment lines in the script will help locating various parts of the macro when revisions are needed. It will also help keep the documentation current whenever the macro is revised and facilitate any future troubleshooting. Additionally, when macros are distributed to other work units, the inline comments will make local customizations easier.

### ***General Programming Practices***

When more than one person works on the macro code, either simultaneously or over time, common programming practices should be agreed on and documented for later referral. See more information about this in the “Technical Aspects” handout.

## **XI. Testing Macros**

### ***Scope of Testing***

A macro that will be used by several people and on different workstations should also be tested on different workstations and in a variety of situations. Broad testing is a major part of the quality control of the macros.

As mentioned earlier, macros tend to run faster than all other parts of the system have time to react. Individual workstations vary in the speed at which they can carry out the macro commands, depending on the number of applications running on the local workstation and the amount of system memory and other resources that they are using. The network connection speed and the Aleph server response time will also affect the timing of macros.

### ***Recruiting Testers***

Although macros are – and should be – tested as part of the regular workflow, testing still takes more time than ordinary daily work. Obtaining and installing test macros, learning about the new macros, running the macros in various situations, testing with different options, documenting test results, and submitting test feedback all take additional time. Supervisor approval should be obtained before asking staff to participate.

Various factors should be considered when selecting and recruiting testers. First of all, testers should have a thorough knowledge of the tasks that the macro is automating and be familiar with the workflow, the standards, and

local policies, so they can recognize when the macro does not work right. On the other hand, it may be useful to get test feedback from staff who are new to the work as well as from experienced staff. Furthermore, some of us like to experiment and adventure to unknown territories more than others. Willing testers will probably do a better job than reluctant ones. Also, a commitment to documenting results and providing feedback is essential, so that you will get the information that you need from testing.

## ***Distributing Test Macros to Testers***

Test macros can be distributed to testers either individually or as part of a package. Consider whether other macros are used as part of the same workflow, and distribute test macros together with already tested macros that are in use as part of the same workflow. We started out by sending individual macros to testers as e-mail attachments, but as our macro repertoire grew larger, we developed a whole separate testing macro file system. Each work unit has a macro file that contains those macros that the unit is currently testing as well as all the macros that have already been tested earlier.

## ***Giving Instructions to Testers***

Testers should be instructed to run the macros in a variety of situations, with different records, at different times of day, and even try to make them fail. Rigorous testing includes running the macro in all the different ways that it is expected to be used.

## ***Testing***

Macro testing should cover various points, including

- Does the macro do what it is supposed to do?
- Does it do it right: does it adhere to standards and local policies and practices?
- Does the macro break?
- Does it take expected system or data errors into account (does it work in all situations)?
- Does it meet a real need?
- Are instructions clear? Does the user understand what the macro is supposed to do?

Obviously, all situations may not come up during the testing period, but rigorous testing and an attempt to cover as many of them as possible will help save time later by allowing more of the possible problems to arise and bugs to be fixed before the macro is distributed to everyone.

## ***Feedback***

Soliciting and receiving feedback from testers can benefit from some structure. A form letter that sets up the scope of testing, including the deadline, and a feedback form that users fill out to submit their test results can be helpful in gathering the same information from all testers in a timely manner. The feedback form can also serve as a reminder to testers about the different things they should pay attention to during testing. Our test feedback form asks for the number of times and the times of day the tester ran the macro, descriptions of any problems that may have arisen during testing, and if the macro documentation was helpful in describing the parameters of use. Setting a reasonable deadline for getting feedback from testers will help move the process along.

When receiving testing feedback, it is good to keep in mind that not all issues should automatically result in macro revisions and not all problems with macros are macro problems. Some of them may be desktop issues. For example, emptying the Windows Recycle Bin may speed up macros or prevent timing problems. Also, relying on standard Window shortcuts, such as <Ctrl>c, in macros may be problematic if these shortcuts don't work consistently.

## ***Revisions, Re-distribution, and Re-testing***

As mentioned above, judgment should be used when evaluating testing feedback. Some problems will, obviously, be addressed by revising the macro. For others, educating or warning the user may be a better way to go. A statement can be placed in the end-user documentation explaining certain situations that might come up when running the macro, or a text box warning could be coded in the macro particularly in situations where the macro will be able to recognize a situation requiring user attention but cannot handle it otherwise.

When, in testing, a macro does not work as intended and trouble-shooting is necessary, debugging scenarios built into the macro code may be helpful. There may be several iterations of the macro during the testing period. Numbering test versions may help keep track of them. Macro Express does not provide a separate way of coding version numbers, but they can be coded as part of the macro name. However, one should use caution when a macro is coded to run another macro that the reference to the other macro uses the correct name including the right version number.

When the macro has been revised, it needs to be re-distributed to testers and re-tested. Testers should then run the macro in similar situations where it broke before to see if errors were fixed.

### ***When Is Testing Complete?***

Setting testing benchmarks ahead of time will help you determine when a macro has been tested enough and can be distributed to all end users. This may be difficult, however, as macros vary in complexity and scope of application. For simple macros, one tester (other than the developer) running the macro in all possible situations for half a dozen times in one or two days may be enough, while complex macros will require more time and many more times to run to ensure that all possible problems will have had a chance to appear.

### ***Removing Test Versions of Macros***

It is important that test macros are recognized as such, and that once testing is done, earlier versions of the macros are removed from staff computers. Managing macros and macro files during the testing period can be challenging. Reflecting the testing status and test version number in the macro naming scheme will help track test macros and their versions. Replacing test macros with the tested versions should also be done with care; filing test macros separately from tested macros may help with this.

## **XII. Distributing Tested Macros**

### ***Centralized vs. Distributed System***

Macro distribution includes making macros available to end users, assigning activation methods, and communicating about new macros, including new, revised versions. A basic decision about macro distribution involves the question of whether or not the process should be controlled centrally or whether local work units or even individual staff members should have the control and responsibility to choose the macros they need, to obtain them, and to assign activation methods to them.

The size of our department made it very important for us to assure that everyone was running the same version of each macro, so we decided to control the process centrally.

Macro Express macros can reside either on the local computer or on a network drive, and the same macro file can be used by several people at the same time, as the program makes a local cache copy of the file at the time of connecting to the file and runs macros from that cache. Our department includes several divisions with somewhat different needs, so we decided to distribute one macro file into each division's shared network directory. Each division's directory is mapped as a network drive to the same drive letter. This allows us to make references to this space in the macro code without having to specify a different location for each division in the macro itself. The choice where the macros will reside has implications to trouble-shooting and support as well as to the ability to customize macros to local or individual needs.

If macro distribution is centralized, support is easier, when you know that everyone is running the same version of the macro. The distribution process is simpler, when you can compile a macro file in one single location, or copy a pre-compiled macro file to a few locations rather than having to copy it to many places. On the other hand, customization of macros, including their activation methods will be difficult or involve more work.

The distribution of macros to local workstations or to individual staff members will result in duplication of effort in installing or importing the same macro many times, and this extra effort only multiplies itself when macros are re-

distributed after revisions. Support and trouble-shooting may also be more difficult because you cannot be sure if everyone is running the same version of the macro. Customization of macros will be easier, whether it is for preferred activation method (shortcut) or for local workflow, but with every revision, those customizations will have to be re-done.

## ***Activation Method(s)***

Macro Express offers several different activation methods. We chose the hotkey activation but we continue to explore other options as our macro library grows larger. Here are some of activation methods and their advantages and disadvantages:

1. Command key / hotkey / keyboard shortcut:
  - More for users to remember
  - Easy to use once memorized
  - Good for macros that are run frequently (several times a day)
2. Menu, open for one selection only:
  - Only one shortcut to remember
  - More steps to take when running a macro
  - Good for macros that are run infrequently
3. Menu, remains on top:
  - Nothing to remember if pops up automatically
  - Intrusive if remains always on top
  - Good for macros that are run infrequently
4. Individual, playable macros:
  - Cannot be edited
  - Cannot run other macros from within them

## ***Distribution Methods***

Macro Express macros can be distributed as individual files or compiled into one or more macro files with their own internal macro management system. Individual macros can be imported between macro files using the Macro Express importing feature. However, some information may be lost in the process. For example, the program only keeps one time stamp for each macro, and this is automatically updated when the macro is imported to a different macro file. Also, if there is a hotkey conflict with an already existing macro, the new macro is imported with no activation.

For centralized distribution, a whole macro file can be compiled and then copied using any of the standard Windows file copying mechanisms. Because of the number of different mapped network drives (folders) in use in our department, we created a macro that copies a pre-compiled macro file to all the various division folders, takes a backup of the previous version of the file, deletes older backups, and creates a log of these actions. This has simplified the distribution method for us significantly.

We have also decided to use two different macro files: one for day-to-day production and the other for testing macros. The testing macro file is built by first creating a copy of the production file and then importing those individual macros that a particular division will be testing. Our separate division folders, all mapped to the same network drive path, allow us to distribute different test macros to different divisions. For any project of limited time and scope, a separate macro file can be compiled with macros that will only be used for that project. Because switching between different macro files can be challenging to some users, a macro can be created to shift between the macro files.

When new or revised macros are distributed, communication must go out to end users about the availability of the macros with information about the macros being distributed. Changes should be highlighted when distributing revised macros. User documentation should be included in the communication when distributing new macros.

## **XIII. Training and Support in Macro Use**

### ***Initial Training***

All users should be aware of basic features of the macro software program as needed for anyone running macros, including

- How to open and close the program, unless this is automatic
- What to do if the program stops running
- How to interpret most common error messages that are likely to come up in the normal workflow
- How to stop a macro before it has run to the end

Support and trouble-shooting responsibilities should be assigned to position(s) to ensure continuity in case of staff changes. These responsibilities must be communicated to all users so they will know who to contact when they need help in using macros.

### ***Instructions for Each Macro***

For all macros and all users, the following information is necessary:

- What the macro does, what it is for
- How to activate (run) the macro
- Where the focus and the cursor should be when the macro is activated
- What the macro assumes (what it does not do or what it does not check)
- How to handle expected errors that the macro will deal with

While this information should also be available in the documentation, it may be useful to give the instructions in other ways as well, as people have different learning styles.

### ***Support and Trouble-Shooting***

In a large work unit like ours, it is helpful to have local staff members assigned to support others in using macros. Troubleshooting, both of the macro program and the individual macros, will often need to be done by expert users or by whoever wrote the particular macro, as this requires a high level of familiarity with exactly what steps are occurring in the workflow. While troubleshooting can be done by any expert user, it is important that there be expert users who have been assigned the responsibility for this task and that it be considered part of their job.

Staff who is expected to trouble-shoot macro problems will need to have a thorough understanding of what the macro does and what it does not do. They will also have to be familiar with the way all the macros and files involved with any particular macro are identified, so they can interpret error messages that include names of macros or files. These people will also need to know about the dependencies between macros and files, such as, "This macro runs the subroutine macro XYZ, which uses a text file at W:\ABC.txt for processing data."

### ***Level of Training and Support***

The amount of training and the level of support needed will depend on the type of macro in question and on the user's sophistication and comfort level with technology. Obviously, simple, straight-forward one-action macros will not require much training, whereas with complex, process-based macros, it is particularly important that users (and especially troubleshooters) understand the parameters of the macro.

Depending on the user's sophistication and comfort level with technology, more training and even individual hand holding may be needed. If a group is at the same level of technical skills, you can do group demonstrations and training sessions.

## **XIV. Revising Macros**

### ***Evaluate Need***

Macros need to be revised from time to time for different reasons: to fix an error; to update the process to meet new standards, policies, or workflow; to improve the macro after learning more about your system; or to allow another work unit to customize the macro for their local needs. Changes in the MARC format and other standards, new versions of Aleph, and changes in local policies should prompt you to review your macro to decide if and how they need to be adjusted.

### ***New Versions of Software or Systems***

New versions of Aleph may require changes to some or all of the macros you have created for an earlier version of Aleph. For example, the interface structure is drastically different between versions 15 and 16, and any macros that operate on the Aleph client interface level must be revised when migrating between these versions.

You may wish to upgrade the macro software that you use in order to take advantage of enhancements and bug fixes, but you should test your macros to ensure that they work with the new version of the software, and you may have to revise them. When you upgrade your macro software, the software should be distributed together with the revised macros. As with any revisions to macros, you should test the revised versions on several different workstations.

Some macros may run differently under different versions of the operating system. Again, you should try to test your macros under a new Windows version before distributing them to all staff.

### ***Development Loop***

Revising macros involves a whole development loop in itself, including getting user input, making design/priority decisions, code writing, testing and distributing the revised macros, and communicating about the changes to users and to the staff who help users or trouble-shoot.

You should first review the information you have about the new environment that is prompting the macro revision to get an idea of possible changes that are expected. When it is considered safe, you should test your existing macros in the new environment to see where they break in order to identify where changes are needed.

## **XV. Communication and Feedback**

Good communication throughout is critical at every stage of the process, and it is worth emphasizing separately. Users should be informed about any changes that affect their work. They should also be given various means to inform those charge of macro development about their needs and how macros are meeting those needs.

We have assigned one person in each division to serve as the primary contact in macro-related issues. All communication about any changes will go at least to those primary support people and the division heads. All staff will get announcement about new or revised macros and any major changes to the process or policy. Both the local macro support contact and the division heads act as conduits of staff input and feedback about any macro-related issues.

In addition, we have an online form on the web that can be used to submit questions, comments, suggestions, or any feedback to the macro implementation team. Our macro website also lists all the macro support staff with their e-mail addresses and phone numbers, to make it easier to find the information who to contact.

Furthermore, any contact that the macro implementation team has with staff – whether in assessing macro needs, soliciting input for new macro design, or coordinating testing – can be used as an opportunity to listen and hear what staff find useful and what they wish macros could do for them. It is our job them to keep listening and asking more questions.

Once feedback is received, you will have to decide what to do with it. Judging possible macro ideas can be difficult and it is important to get more than one person's input on any given idea. We collect submitted ideas onto our wish list and periodically prioritize this list for further macro development. Regardless of the outcome, you should always follow up on feedback you have received, to make sure that staff feel their input mattered and to encourage them to continue submitting feedback to you.

## **XVI. Policy Development**

### ***Programming Practices***

To communicate between those involved in macro programming and to document practices for any future programmers, a policy should be developed on what programming practices are desirable, what are acceptable, and what should be avoided. Adhering to common programming practices is also important to ensure that different macros work together and to help users understand what to expect from macros over time.

### ***Distribution***

Deciding on who should get which macros is the first step in developing a distribution policy. Giving staff access to only those macros that they need in their daily workflow would minimize the risk of unintentionally invoking a wrong shortcut and running a macro that performs tasks that the user is not familiar with. On the other hand, distributing all macros to everyone would simplify the distribution process. In Macro Express access to individual macros can also be limited by assigning a password for running the macro.

The distribution decision also depends on your policy on local or individual customization of macros. While flexibility to customize one's own macros, including assigning different activation methods, may be desirable to individual staff members or work units, greater benefits may be reached by simplifying the distribution and ongoing macro support by ensuring that everyone will have the all the same macros with the same activations.

Distribution method possibilities may inform your policy decisions. Macro Express allows you to import macros between macro files so long as the macro file resides in a directory where the user has read-write privileges. If the target macro file is on the local workstation, each user can have a different set of macros, and local adjustments to macros will not interfere with those macros that others are using. On the other hand, compiling one file with all the latest versions of the macros and copying it to the destination directory is not only simpler to do but will also guarantee that everyone is using the latest versions of the macros.

### ***The Scope of Macro Use***

Limited scope of macro software use may have been negotiated in the license agreement to get better pricing. Your macro development policy should take this into account.

Copying data from non-Unicode-compliant systems into a later version of Aleph may result in corrupt characters in your database. While this is not a macro issue, the use of macros may make these types of errors easier to produce and slip through without the user noticing.

### ***Standards, Policies, and Practices***

Your macro policy should help ensure that macros adhere to cataloging standards and local workflow policies and practices. It is important to include into the macro workflow a step where macros are vetted against any relevant standards, policies and practices. Having expert staff test macros will also help guarantee a solid macro design.

### ***Determining Benchmarks***

When setting up macro policy, it is wise to decide ahead of time what will be used as benchmarks for evaluating both the macros and the development and implementation process. The question of why you want to use macros is central here. If straight time savings at the task level are important to you, then set up a way to measure the time it

takes to accomplish the same tasks with and without using macros, and compare those also to the time spent developing macros. It is good to revisit your goals and benchmarks as you develop more macros and learn more about macros and get more feedback from users and from other measures, such as database errors over time.

## **XVII. Personnel Changes**

### ***Justifying Staff Time***

As mentioned before, macro writing responsibilities should be assigned to a position and not to a person. However, people may leave positions, and sometimes vacant positions are not filled, at least not immediately. Thinking about what to do when your macro expert gets a better job or when there are any personnel changes may help you get prepared for the new situation. Making sure that all documentation is up to date before the person leaves will help the next person to continue.

If you are trying to get a new position established or if a vacant position is considered for elimination or reassignment, having statistics documenting the benefits of investing the time and effort in macro development, and being prepared with statistics to back up the need for personnel/staff time, may be essential to continuing macro development.

### ***Projects vs. Ongoing Macro Development***

If you approach macro development as a one-time project, make sure the macro developer has documented the work in a way that allows others to maintain the macros. If your approach is for ongoing macro development, ongoing documentation is equally important, because it is always more difficult to go back and try to document things afterwards.

## **XVIII. Evaluation**

Assessing your needs and deciding ahead of time what your goals are will help you evaluate the value of your efforts afterwards. After you have worked with macros for some time and know their possibilities and limitations better, you may need to reassess your needs and readjust your goals.

Ask yourself the question, “Why are we implementing macros?” Are time savings the most important goal? Does reducing key strokes and mouse clicks provide value to us? Is data consistency more important than time savings at the individual task level? How can you measure these outcomes?

Time savings at the task level may be easy to measure, but what if you are not realizing any time savings? Reducing key strokes, mouse moves and clicks, and the waits in between may produce ergonomic benefits even if the process is no faster with macros than without. Number of mouse clicks and key strokes can be counted and compared. Are there other ways to measure improved ergonomics? Increased data consistency may produce long-term time savings, but can you measure those? Instinctively we all understand the importance of data quality, but it is difficult to quantify its benefits, because it affects so many parts of the bibliographic process, from technical services to public services to the system migration to the end-users years from now.