FROM: V. L. Voydock, R. J. Feiertag

DATE: June 7, 1971

SUBJECT: A Proposal for the Elimination of CACL's

This memo presents a number of arguments for the removal of the concept of the CACL from the Multics File System, and its replacement by a new construct which provides all of the benefits of the CACL with none of its problems.

There are a number of problems which make CACL's much less useful than they might otherwise have been. These fall into three categories:

a) "Too global" access specification

b) Difficulty of determining the effect of changing a CACL entry

c) Interaction with rings

First let us consider a). Let us define access type as follows: two segments are of the same access type if they require exactly the same access in order to be correctly utilized. (For example, all non-execute-only-pure procedure segments are of the same access type, namely "re").

By "too global" access we mean that a given CACL entry applies to all segments in the directory. Thus if one has segments of different access types in a given directory and one wishes to use the CACL to control access to these segments, one must put the "or" of the access types into the CACL. For example, if one has both executable and writeable segments

segments in a directory the mode specified in the CACL must be "rewa". This totally disables the Multics protection mechanism, since one can now write into pure procedure segments and attempt to execute data segments. Thus to safely use the CACL, all segments in a given directory must be of the same access type. This is a rather harsh restriction.

This problem is greatly multiplied if we introduce the concept of extended access. Extended access is software interpretable access which will vary from subsystem to subsystem. In order to use the CACL, all segments in a given directory must not only be of the same standard access type but of the same extended access type. This is a very harsh restriction.

The second problem is that it is difficult to determine what effect the changing of a CACL entry will have on access to the segments in the directory. This depends entirely on what access appears on the ACL of each particular segment. For example, a user might put Jones.Multics on the CACL of one of his directories with "re" access, thinking that this would give Jones.Multics "re" access to everything in that directory. As we all know, this is not necessarily the case. If segment X in this directory has an ACL entry "*.Multics.* null" then Jones.Multics will not have access to X even though he appears on the CACL.

The third problem comes about due to the interaction of CACL's and rings. Since the CACL is logically appended to the ACL of every segment in the directory, a change to a CACL entry is effectively a change to every ACL. Thus, in order to determine whether a given user has the right to modify the CACL, the system must apply the rules described in the memo on ACL's above. In particular, the system must verify that the validation level of the user making the request is less than or equal to that user's first ring bracket, r1, with respect to every segment in the directory. Besides being expensive to implement, this makes the CACL practically useless if one has segments from more than one ring residing in the same directory. For example, if one is running in ring 4, one cannot manipulate the CACL of any directory containing a message segment, since message segments reside in ring 1.

This extremely harsh restriction must be imposed just to enforce the rules governing access control. In addition, if a user wishes to securely make use of the ring mechanism by limiting the rings in which a segment can be read or modified or executed he must either fully specify the ACL of that segment (i.e. have a "*.*.*" entry on it) so that the CACL will not be used or he must verify that the CACL does not contain an entry that would invalidate his limitation. For example, suppose a user wishes to create a segment that is only manipulable in ring 1. Let us also suppose that the CACL of the directory in which the segment is to be created contains the single entry "*.*.* rewa 4, 4, 4". Then the user would have to fully specify the ACL of his segment since otherwise anyone not explicitly

appearing on the ACL would be able to manipulate the segment from ring 4.

It should be clear from the above that the CACL, as it is now defined, is almost useless. Two proposals, that have been previously discussed, for changing the definition of the CACL have major flaws. The first proposal is to have one CACL per ring. This solves the problems raised by c), but does nothing for problems a) and b). For this reason it is unacceptable.

The other proposal was to have **multiple CACL's.** That is, segments would somehow be gathered into groups each group having its own CACL. This solves problems a) and c). Segments could be grouped according to access type or ring or whatever else proves useful. This does not solve problem b). In order to intelligently change a CACL entry one must still have detailed knowledge of the ACL's of all segments in that CACL group. In addition, since managing a single CACL now causes some confusion, managing multiple CACL's could not help but increase the confusion. Next, when one wishes to change a CACL entry one must check the ACL of every segment in the group to see if the change is legal -- an expensive operation. Finally, there is the problem of how to do the grouping. If it is done by name, then segments with more than one name would fall into more than one group and one's access to a segment would depend on which name one used to reference it. This does not seem to be a good idea. If we are not to do the grouping by name, the only reasonable choice seems to be to introduce a new branch attribute, let us call it "CACL group number" by which segments will be grouped. When the user creates a segment he must specify a CACL group number for that segment. This adds one more complication to the life of a Multics user. For these

reasons, we contend that this scheme will be difficult to explain and use and should be discarded.

The CACL construct provides two features that we would like to preserve. First, it provides a way to specify the initial access to apply to newly created segments. That is, it saves the user the trouble of specifying an ACL every time he creates a segment. Of course, because the access is "too global" (see problem (a) above) this isn't as useful as it could be. Second, it provides a way to specify global access, that is, control access to a large group of segments at once. Again because its access is "too global" this is not as useful as it could be.

We now propose a scheme which provides both these features in a much better way and has none of the problems discussed above. We propose that the CACL be eliminated and that the concept of an "initial ACL" be introduced. When a segment is created, its "initial ACL" will be appended to the ACL with which it was created. The name by which the segment was created determines which "initial ACL" to use, and the system allows the user to specify that a group of segments will all use the same initial ACL by using the star convention. For example, the user could specify that all segments created with a name that matches "*.pl1" should have the same initial ACL. A naming convention is reasonable here where it wasn't for multiple CACL's since initial ACL's only come into play when a segment is created. That is, the naming convention is used only to determine what the ACL should look like when the segment is created. If one later renames a segment this will, of course, have no effect on its ACL. Whereas, the reader will

recall, in the case of multiple CACL's, the name was used to determine access on each name reference. That is, if one renamed a segment and referred to it by its new name, one might have different access to it than before it was renamed.

Before discussing the virtues of this scheme, let us discuss two problems with it. The first is an intrinsic problem with rings which our scheme, of course, does not solve. A user trying to create a segment which can only be used in certain rings must have complete control over what appears on the ACL of that segment and therefore cannot blindly use initial ACL's. The problem is best illustrated by an example. Suppose a user wants to create a segment X which is fully accessible in rings lower than 4 and not at all accessible in other rings. He would create X with ACL "*.*.* rewa 3, 3, 3". Suppose the initial ACL associated with X contained the single entry "Jones.Multics.* rewa 4, 5, 5"; the rules for modifying ACL's would allow the initial ACL to be appended and suddenly the segment would be accessible outside of ring 3. Therefore, the user would either have to check the initial ACL to insure that it did not contain an entry which violated security or not use the initial ACL. The perceptive reader will have noted that this problem is analogous to one encountered with CACL's above. The problem is intrinsic to the ring mechanism. Only the creator of a "protected" segment knows which rings should be able to access it and in what way. The system can only insure that a change to an ACL is legal it cannot insure that it is suitable.

Another possible problem is that our scheme may significantly increase the size of a directory since information now stored in the CACL will have to be stored in the ACL of every segment in the directory. What effect this will actually have has not yet been determined.

how much ?

The virtues of our proposal are many. First, it provides the initial access feature of CACL's. In fact, it is better since it allows different segments in a directory to have different initial access. Global access, the other feature that CACL's provide, can be had by using the setacl command with the star convention. For instance, if one wishes to give "rwa" access to all pl1 segments in a directory to Jones.Multics, one merely types

"setacl *.pl1 rwa Jones.Multics"

This is better than the global access CACL's provide since it allows the user to specify different global access to different groups of segments.

In addition to doing a better job of solving the problems CACL's were meant to solve, our proposal has the virtue of simplicity. Problem (b) discussed above disappears. In addition, in order to determine who has the right to access a segment one merely lists its ACL. Thus our scheme should eliminate much confusion while providing users with more flexible means of controlling access to groups of segments.