TO: Distribution

From: C. D. Tavares

Date: 03/01/78

Subject: RCP Resource Management Databases

INTRODUCTION

This MTB outlines the proposed design of the RCP Resource Management Facility Database. An overview of the Resource Management Facility (née Device and Volume Management Facility) can be found in MTB-350.

Two databases are described. The first, the Resource Registration Database, contains information about specific devices and volumes. The second, the Resource Type Description Table, contains global parameters that control the operation of the Resource Management Facility, including descriptions of generic types of resources.

THE RESOURCE REGISTRATION DATABASE

This database will serve as the repository of all static information about all devices and volumes registered, including resource name, owner name, unique ID, access information (ACL, ring and AIM), physical location information, user comments, error counts, and resource attributes. (In some cases, the database will not contain the information directly, but will contain some locator that specifies where the desired information can be found.) It will not include dynamic information, meaning such items as, "Is this reel mounted now, and if so, by whom?" Such operational information will remain in the RCP database proper.

Since each device and volume at a site must be registered before it can be used with the MR7.0 version of RCP, the number of entries in the database will be quite large. (For example, the Air Force Data Services Center library contains roughly 100,000 tapes alone.) Because of this, the space/time trade-off will be made largely in favor of space. Commonly duplicated data fields longer than one word will not be contained within the records describing the resource. Instead, a one-word locator which uniquely identifies that longer field will be used to locate it. For example, the pathname of an ACS that serves 1,000 tapes will

appear only once in the database; and the record describing each reel that is controlled by it will contain a one-word locator to that string. In this example, which is typical, 40K can be saved.

However, speed will not be sacrificed in the case of routine and common operations against the database such as, "Give me the information about pack 12345", "Find the name of the reel with UID N", and "Tell me all the reels owned by Smith.Project." This consideration implies two important requirements: First, each record in the database must be locatable through several keys at once. For instance, one should be able to obtain the same record for a resource when asking for it with a resource name, or a resource UID, or as one of the resources owned by Smith. Second, each key should be able to locate several records at once. In this manner, each resource owned by Smith can have a key which identifies it as one of the resources owned by Smith.

The current implementation of vfile_ contains control orders (those for manipulating "record descriptors") that make it suitable for managing this database. Being able to use vfile_ not only releases us from the necessity of coding another hairy list-structure manager, but allows easy generation of reporting programs and salvagers to be run against the database.

Actually the "database" consists of multiple vfiles. Each vfile contains all the information about resources of one type. For instance, there will be one vfile for printers, one for tape reels, and for disk drives, and so on. The term "database" may be loosely applied to mean both an individual vfile or the collective assembly of vfiles, but this is generally unambiguous.

Resource Description

Each resource record will contain various fields of information (or a short locator specifying where the information can be found external to the physical record.) The fields are:

1) Resource name. This will be an ACC string containing the resource ID (e.g. "dska_05", "C109583", "apl_train".)
2) Resource unique ID. This will be generated in hardcore using the code that generates segment UID's.
3) Potential attributes (locator). This is a two-word field that specifies which attributes of the resource type this particular resource can assume.
4) Current attributes. Specifies which attributes are currently active.
5) Protected attributes. Specifies which attributes are currently protected.
6) Potential AIM bounds. The resource will be open to acquisition by a process if that process' authorization is between

the low bound and the high bound, inclusive.

7) AIM bounds. These will be set at acquisition time to the current authorization of the acquiring process. Thereafter, this resource will be useable only by processes with this authorization. A privileged command will allow the two elements of the AIM bounds to describe a range. This will be used for multi-class devices and storage system logical volumes.
8) Owner (locator). The userid (Name.Project) of the user who is the accounting owner of the resource.
9) ACS pathname (locator). Specifies the directory in which the ACS segment for this resource is located. The name of the segment will be <resource_name>.acs .
10) Location (locator). A short string describing the physical location of a resource (e.g. "offsite", "vault".)
11) Comment (locator). An arbitrary string (char (168) maximum) settable by any resource executive for this resource.
12) Error count, and
13) Number of uses. Given a suitable algorithm, these two fields could give some indication about the recent reliability of a resource.
14) Reserver chain. This item exists mainly for the edification of the Reserver mechanism, and its meaning and use will not be described here; except to mention that it can serve as an indicator that only immediate reservations will be honored for the resource described by this record.

Key/Record Correspondence

The design of the keying structure of the file determines three parameters: what information (in terms of correspondence between items) can be reasonably gathered, how fast this information (with particular emphasis on frequent operations) can be gathered, and how well various fields can be shared to decrease storage space.

The most common operation by far will be the location of a record by resource name, followed closely by the location of the record by its resource UID (the Volume Retriever will use this mechanism extensively.) Each record in the database will be associated with one UID key and one name key.

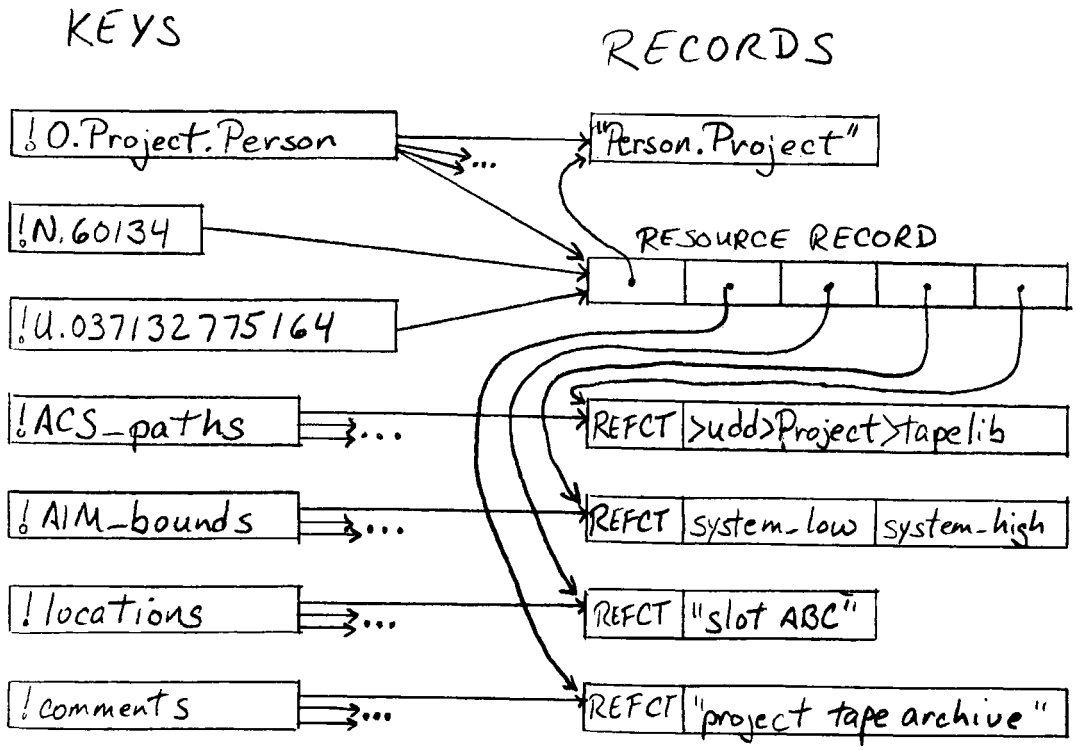
The next most common operation will be the the location of volumes for a given accounting owner. Each record will also be associated with an owner key. Because we want this information to be available to a project administrator on the project level, we will arrange the owner key in the order "Project.Person", and make use of the properties of vfile_ which allow a programmer to seek to the first key beginning with a given string. The first record associated with an owner key will not be a resource record, but will be a string containing the owner name. Each re-

source record referenced by an owner key will contain, in its owner locator field, a vfile_ descriptor to this first record. (Although the owner name is trivially computable from the value of the owner key, there is no guarantee that it was the owner key instead of, say, the UID key that was used to find this record; and there is no way for a record to contain a short locator for a key-- locators, which are vfile_ descriptors, can only reference records.)

Many of the record fields will be shared (e.g. the pathname of the ACS, as described above.) This requires us to be able to search all records of a certain type (e.g. all known ACS pathnames) to determine if we can share an existing entry. There will be a known "canned" key for each type of shared field. This key (really, multiple keys with exactly the same name) will point to all known fields of a single type. Descriptors inside each resource record will refer to the appropriate shared field. A reference count associated with each shared item (e.g. a single physical ACS pathname string) will enable the freeing of the record when it is no longer needed.

An example of the descriptor references and chaining properties of a resource record is shown in Figure I. Fields contained directly in the record are omitted.

Figure I: Sample Chaining for Tape 60134



Field Access Control

The Resource Management Database will be located in ring 1. Users will be able to access it only through selected gate entries. This allows the associated programming to control access to certain fields of records at the field level. The reading or writing of each field by a process, given the process' access to the chosen resource, will be subject to very definite rules.

The matrix below shows the minimum access needed to perform the named actions. The access notations "r", "e", and "w" refer to fully factored effective access, including ACL and ring brackets of the ACS, and the AIM bounds associated with the resource. The notation "p" specifies that access to a privileged gate is required to perform the action.

	read	write
Resource name	r	p (to create)
Resource UID	r	p (to create)
Potential attributes	r	p
Current attributes	r	rw (note 1)
Protected attributes	r	rew
Potential AIM bounds	p	p
AIM bounds	r	can't (note 2)
Owner	r	can't (note 2)
ACS pathname	r	rew
Location	r	p
Comment	r	rew
Error count	r	rew (note 3)
Number of uses	r	rew (note 3)
Reserver chain	can't	can't

Note 1: Additionally, "e" access is needed to alter a current attribute which is also a protected attribute.
Note 2: These are automatically written on behalf of the accounting owner's process at acquisition and release time.
Note 3: The only operation permitted is the setting of both fields to zero simultaneously.

THE RESOURCE TYPE DESCRIPTION TABLE

This table serves to define the types of resources available at a site, and associates with them various parameters such as potential attributes, default reservation time limits, and so on. It is used not only to control the RCP Resource Management Facility at run time, but controls certain aspects of registration and contains information which makes it possible for a site to define certain common subclasses of resources.

This table will be readable from the user ring, and will be installed via the "install" command.

Table Entries

An entry exists in the table for each type of resource (e.g. "tape_drive", "disk", "forms".) The entry contains information necessary to the operation of RCP, and information necessary and desirable for facilitating the registration operation.

The parameters necessary to RCP operation are:

1) Whether this resource type specifies a device or a volume.
2) If it is a device, what volume type(s) may be mounted on it.
3) If it is a volume, what device type it must be mounted on. (This information is also used to "imply" the reservation of a matching device when a volume is reserved.)
4) An exhaustive list of attributes which any resource of this type may be defined to possess as potential attributes.
5) A limit for the number of resources of this type that any one process may reserve at one time.
6) A default and a maximum time limit for reservations of this resource.
7) An advance warning interval, which may be used to prompt the operator to prepare a resource for intended use.
8) For volumes, a flag specifying whether a manual clearing operation must be performed after a volume of this type is released by a user, and before another user may acquire it.

The parameters necessary during the registration process are:

1) A charge type, which is used in communications between RCP and the Initializer process, for accounting purposes.
2) A list of default potential attributes.
3) A minumum and maximum AIM access class (which becomes the potential AIM bracket.)

Resource Subtypes

Each table entry may contain a description of one or more resource subtypes. Resource subtypes are useful in the registration process, to register groups of volumes or devices with multiple parameters that are common to the group, but different between different groups. For instance, a site may want to define a subtype of the "tape" resource, named "secure", which automatically specifies a different AIM access bracket and charge type from the default "tape" resource.

A subtype within an entry is defined by a name, and explicitly defined values for any or all of the parameters necessary for registration. When a resource is registered in terms of this subtype, the defined parameters for the subtype override any default defined parameters in the main entry.

A site may define the table in such a way that subtypes are mandatory or optional during the registration process. If the main entry contains values for all of the registration parameters, then a resource of this type may be registered without referring to any subtype that may exist. However, a site may omit one or more of these paramaters in the main entry and instead define them only within subtypes. This makes it necessary to specify one of the defined subtypes when registering a resource of this type. For instance, a secure site may not want to specify a default AIM bracket for tapes, but instead will choose to force the operator to explicity specify (and therefore realize) whether he is registering a tape with a low or high AIM bracket.