IPC-
Roach

To: Distribution

From: Richard J.C. Kissel

Date: March 20, 1978

Subject: Design review of MTB 365

There will be a design review to discuss MTB 365 on Wednesday March 29,1978 in Conference Room II at CISL.

To: Distribution

From: Richard J.C. Kissel

Date: March 17, 1978

Subject: Resource Reservation for Release 6.5

Introduction

This MTB describes the part of the Resource Reservation Facility to be done for Release 6.5 (see MTB 352 for an overview of the complete facility).

Changes to the enter_abs_request and the absentee manager are described as well as a user ring interface to Resource Control. These changes will allow the user to have an immediate reservation of tape and disk drives and tape and disk volumes necessary for his absentee job. If the desired resources are not available at the time the job is to run the job will not be started at that time but will try again to get the necessary resources at a later time (see MTB 364 for a more complete discussion of the absentee manager).

Changes to the ear Command

The user interface to the ear command will be extended with the addition of a new control argument:

-resource,-rsc "resource_desc1 resource_desc2 ..."

which takes a quoted string of resource descriptions as an argument. Each resource to be reserved for this absentee request is described by one of the resource descriptions and the absentee job will not start running until all of the described resources are available. Each resource_desc has the following format:

resource_type resource_spec {-number,-nb N}

where resource_type must be first and the other two arguments may occur in either order.

The resource_type must be one of:

tape_drive
disk_drive

tape_vol
disk_vol

Note that tape and disk may be used instead of tape_drive and disk_drive to maintain compatibility with the assign_resource command, however, their use is discouraged in the hope that in the future they may be used as synonyms for tape_vol and disk_vol instead.

The resource_spec is either:

resource_name
or -attributes,-attr attribute_string

where resource_name is the name of the desired resource, e.g. tape_02 or 050102; and attribute_string is a string of attributes of the resource, e.g. "track=9,den=800". If the resource_name begins with a "-" then it must be preceded by the "-name,-nm" control argument. If the resource_type is tape_vol or disk_vol then the resource_name must be specified. If the resource_type is tape_drive or disk_drive (or tape or disk) then either the resource_name or an attribute string may be specified.

The currently allowed attributes are:

Attribute		Value	Default
Tape Drives:			
	model=	400, 500, 600	500
	track=	7, 9	9
	den=	200, 556, 800, 1600	800
Disk Drives:			
	model=	181, 190, 400, 451, 500	451

Finally, if the resource_type is tape_drive or disk_drive (or tape or disk) and an attribute string is specified then a number of resources of that type may be specified using the -number argument.

An example of a complete reservation using the ear request follows:

```
ear foo -resource "tape_vol 50102 tape_vol U309 tape_drive
        -attr track=9,den=1600 -nb 2"
```

The ear command will take the entire string following the -resource control argument and pass it to parse_resource_desc_$check described at the end of this MTB. A code telling whether or not this is a valid reservation description string will be returned and the ear command will proceed accordingly. The reservation description string will

also be part of the information given to the absentee manager by the ear command.

Changes to the Absentee Manager

When the absentee manager is about to start a job it will check to see if there is a reservation description string with the job. If there is, it will first call parse_resource_desc_ with the string. It will get back structures suitable for input to resource_control_$reserve which will be called next (both of these subroutines are described at the end of this MTB). An indication of whether or not the reservation was made will be returned. If the reservation was made then the absentee manager will go ahead and start the job. If not the job will not be started and the same process will be repeated at some later time until the reservation is made. See MTB 364 for details.

What follows is MPM type documentation for the subroutines mentioned previously.

Note that the resource_control_$reserve subroutine should only be called with the structures obtained from calling the parse_resource_desc_ subroutine or entry. Also, only privileged reservations are currently supported (i.e. system = "1"b in the calling sequence of resource_control_$reserve).

Name: parse_resource_desc_

This subroutine takes a reservation description string as input and returns pointers to two structures containing the necessary information to make a reservation of the described resources. It calls cv_rcp_attributes_$from_string to convert the attributes character string to a bit string if necessary.

Usage

```
declare parse_reserver_desc_ (char (*), ptr, ptr, ptr,
     fixed bin (35));

call parse_reserver_desc_ (desc_string, area_ptr,
     resource_desc_ptr, reservation_desc_ptr, code);
```

where:

1. desc_string (Input)
   is a reservation description string, normally obtained from a command level interface.

2. area_ptr (Input)
   is an area in which the structures to be returned will be allocated.

3. resource_desc_ptr (Output)
   is a pointer to the resource description structure described in resource_control_desc.incl.pl1.

4. reservation_desc_ptr (Output)
   is a pointer to the reservation description structure described in resource_control_desc.incl.pl1.

5. code (Output)
   is a standard system status code.

Notes

If area_ptr is null then the system free area will be used.

If an error is detected a non-zero code will be returned and both pointers will be returned null.

Entry: parse_resource_desc_$check

This entry takes the same inputs and returns the same outputs as parse_resource_desc_. However, more complete diagnostics are available in case an error is detected.

Usage

```
declare parse_resource_desc_$check (char (*), ptr, ptr, ptr,
     fixed bin (35));

call parse reserver_desc_$check (desc_string, area_ptr,
     resource_desc_ptr, reservation_desc_ptr, code);
```

where:

1. desc_string (Input)
   is a reservation description string, normally obtained from a command level interface.

2. area_ptr (Input)
   is an area in which the structures to be returned will be allocated.

3. resource_desc_ptr (Output)
   is a pointer to the resource description structure described in resource_control_desc.incl.pl1.

4. reservation_desc_ptr (Output)
   is a pointer to the reservation description structure described in resource_control_desc.incl.pl1.

5. code (Output)
   is a standard system status code.

Notes

If the resource description string is not valid then the sub_error condition will be signalled by a call to sub_err_ with a description of the error. Processing will continue after the call.

If area_ptr is null then no structures will be allocated and both output pointers will be returned as null. That is, only a syntax check of the input will be done.

Name: resource_control_

Entry: resource_control_$reserve

This entry reserves a resource or group of resources for use by a process.

Usage

```
declare resource_control_$reserve entry (pointer, pointer,
     bit (1) aligned, fixed bin (35)));

call resource_control_$reserve (description_ptr,
     reservation_desc_ptr, system, code);
```

where:

1. description_ptr (Input)
   is a pointer to the structure containing a description of the resources to be reserved. See "Resource Description" below.

2. reservation_desc_ptr (Input)
   is a pointer to the structure containing reservation information for the resources to be reserved. See "Reservation Description" below.

3. system (Input)
   specifies, if "1"b, that the calling process wishes to perform a privileged reservation. See "Notes" below.

5. code (Output)
   is a standard system status code.

Resource Description

The argument description_ptr points to the following structure: (This structure is declared in the include file resource_control_desc.incl.pl1.)

```
dcl 1 resource_descriptions aligned based (description_ptr),
    2 version_no fixed bin,
    2 n_items fixed bin,
    2 item (num_items refer (resource_description.n_items))
          aligned,
```

```
        3 type char (32),
        3 name char (28),
        3 uid bit (36),
        3 attributes (2) bit (72),
        3 owner char (32),
        3 acs_path char (168),
        3 aim_bounds (2) bit (72),
        3 location char (168),
        3 comment char (168),
        3 error_count fixed bin,
        3 number_of_uses fixed bin,
        3 state bit (36) aligned,
        3 status_code fixed bin (35);
```

where:

1. version_no
   is the current version number of the structure. It should be set to "resource_control_version_1".

2. n_items
   specifies the number of resources described by this structure. A consistent combination of the following elements must be supplied for each resource described.

3. type
   specifies the type of resource desired (e.g., "tape", "disk_drive".) It must be supplied.

4. name
   is a specific device or volume name. If this element is supplied, an attempt is made to acquire the named resource. If this element consists of blanks, a resource is chosen depending on criteria specified by other elements of the structure, and the name of the resource chosen is returned in that element.

5. uid
   is the unique ID of a specific device or volume. If this element is supplied, an attempt is made to acquire the specified resource. If this element is "0"b, a resource is chosen depending on criteria specified by other elements of the structure, and the unique ID of the resource chosen is returned in this element.

6. attributes
   contains the specification of attributes which the resource chosen must possess. If these elements are "0"b, the resource to be acquired need not possess any particular attributes. The attributes of the resource chosen are returned in these elements.

7. owner
   is the owner of the resource. If system = "1"b, this element specifies the name of the user for whom the resource is to be acquired. Otherwise, this element is ignored and the resource is acquired for the calling process.

8. acs_path
   is the pathname of the Access Control Segment (ACS) for this resource. It must be supplied.

9. aim_bounds
   are a pair of AIM access classes, specifying the minimum and maximum process authorization that can be permitted to both read and write to this resource. This element is ignored on input.

10. location
    contains a character-string description of the location of this resource. It is ignored on input.

11. comment
    contains a character-string comment which is associated with this resource.

12. error_count
    contains a count of the number of I/O errors which have been attributed to this resource.

13. number_of_uses
    contains a count of the number of mounts performed using this resource.

14. state
    is for the use of resource_control_ and should not be modified by the user.

15. status_code
    is a standard system status code. If code is nonzero, one or more items in the structure will have a nonzero status_code specifying in more detail

why the attempt to manipulate the described resource was refused.

Reservation Description

The argument reservation_desc_ptr points to the following structure declared in the include file resource_control_desc.incl.pl1:

```
dcl 1 reservation_description aligned based,
    2 version_no fixed bin,
    2 reserved_for char (32),
    2 reserved_by char (32),
    2 reservation_id fixed bin (71),
    2 group_starting_time fixed bin (71),
    2 asap_duration_limit fixed bin (71),
    2 flags aligned,
     (3 auto_expire bit (1),
      3 asap bit (1),
      3 rel bit (1),
      3 sec bit (1)) unaligned,
    2 n_items fixed bin,
    2 reservation_group (num_items refer
                         (reservation_description.n_items)),
      3 starting_time fixed bin (71),
      3 duration fixed bin (71);
```

where:

1. version_no
   is the current version number of this structure. It should be set to "resource_control_version_1".

2. reserved_for
   specifies the group id of the process for whom this reservation is made. The use of a "*" for a component name is permitted. If this element is blanks the group id of the current process is used.

3. reserved_by
   is the group id of the process which is charged for this reservation (see "Notes" below). This element is ignored for an unprivileged reservation and the current process group id is used.

4. reservation_id
   is an identifier for this reservation group. This

value must be used in all future references to this reservation.

5. group_starting_time
   specifies the time at which this reservation group is to start. If this time is less than or equal to the current time the current time is assumed.

6. asap_duration_limit
   specifies, if asap = "1"b, a time interval after which a reservation is no longer desirable.

7. auto_expire
   specifies, if "1"b, that this reservation group should be cancelled if the process for which the reservation is being made terminates before the reservation expires.

8. asap
   specifies, if "1"b, that the reservation should be made to start as soon as possible from the group_starting_time subject to asap_duration_limit. The group_starting_time is output in this case. If "0"b, the reservation is made to start at the group_starting_time.

9. rel
   specifies, if "1"b, that group_starting_time is relative to the current time. If "0"b, group_starting_time is an absolute time from January 1 1901, 0000.0 hours Greenwich Mean Time.

10. sec
    specifies, if "1"b, that group_starting_time, asap_duration_limit, starting_time and duration are in seconds. If "0"b, group_starting_time, asap_duration_limit, starting_time and duration are in microseconds.

11. n_items
    is the number of reservations described by the reservation group. It must equal resource_descriptions.n_items in the associated resource_descriptions structure.

12. starting_time
    for each resource being reserved specifies the time the reservation is to start relative to the group starting time.

13. duration
for each resource being reserved specifies the duration of the reservation. If this element is zero a site specifiable default value is used and the duration is returned.

Notes

If system = "1"b, reservation_description.reserved_by is used to specify the group id of the process to be charged for this reservation. The user must have "re" access to the gate rcp_sys_ to specify system = "1"b.

The structures resource_descriptions and reservation_description are strongly dependent on each other. That is, for each resource described in resource_descriptions there must be a corresponding entry of the same index in reservation_description.

Name: cv_rcp_attributes_

The cv_rcp_attributes_ subroutine contains several entry points that are useful in manipulating RCP resource attribute specifications and descriptions.

Entry: cv_rcp_attributes_$to_string

This entry point takes an RCP resource attributes specification and returns a character representation of the specified attributes.

Usage

```
declare cv_rcp_attributes_$to_string entry (char (*),
     bit (72) dimension (2), char (*), fixed bin (35));

call cv_rcp_attributes_$to_string (type, attributes, string,
     code);
```

where:

1. type (Input)
   specifies the type of resource from which attributes was obtained (e.g., "tape", "disk_drive".)

2. attributes (Input)
   is an RCP attribute specification.

3. string (Output)
   is a printable RCP attribute description.

4. code (Output)
   is a standard system status code.

Entry: cv_rcp_attributes_$from_string

This entry point applies a printable RCP resource attribute description to a given resource specification and returns a new attribute specification as the result.

Usage

```
declare cv_rcp_attributes_$from_string entry (char (*),
     bit (72) dimension (2), char (*),
     bit (72) dimension (2), fixed bin (35));

call cv_rcp_attributes_$from_string (type, attributes,
     string, new_attributes, code);
```

where:

1. type (Input)
   specifies the type of resource to which attributes and string apply.

2. attributes (Input)
   is an RCP attribute specification.

3. string (Input)
   is a printable RCP attribute description that is to modify attributes.

4. new_attributes (Output)
   is the new RCP attribute specification.

5. code (Output)
   is a standard system status code.

Notes

If an error occurs while converting the attribute string then the sub_error condition will be signalled by a call to sub_err_. This call will be restartable and will include a message explaining why the string was in error.

Default attributes are returned for any attributes not specified in the attribute string.