Published: 8/14/67

Identification

The I/O Registry File Maintainer
S. I. Feldman

Purpose

The I/O Registry Files describe all of the I/O devices known to a system. Certain parts of these files are of interest to IOS outer modules. An I/O Registry File Maintainer (IORFM) is supplied to meet the needs of such modules.

The I/O Registry Files

The I/O Registry Files have the following declaration:

```
dcl 1 rf based(p),
  2 level fixed bin(17),          /*level=1 for a CIOC channel, 2 for
          "                         a device connected directly to
          "                         a CIOC channel, etc.*/
  2 user_id char(50),             /*id of user assigned to this device*/
  2 ioname char(32),              /*ioname of DSM for this device*/
  2 pdt_name char(32),            /*name of PDT in DMP*/
  2 force_udmp bit(1),            /*if 1, force the use of a universal
          "                         device manager process*/
  2 udmp_user_id char(50),        /*user_id of universal device manager
          "                         for this device, if any*/
  2 in_use_switch bit(36),        /*set ON at attach time and OFF
          "                         at detach time*/
  2 hangupable bit(1),            /*if ON, device can hang up*/
  2 allocate bit(1),              /*if ON, Reserver should be called
          "                         with each resource_name as argument.*/
  2 temp_rf bit(1),               /*delete this file when detaching
          "                         device*/
  2 temp_link bit(1),             /*connection with next file is
          "                         temporary. Blank out resource_name
          "                         entries upon detachment*/
  2 nup fixed bin(17),            /*number of entries in up array,=N1*/
  2 ndev fixed bin(17),           /*number of entries in devices,=N2*/
  2 ntypes fixed bin(17),         /*number of entries in att_types=,N3*/
  2 present_type_index fixed bin(17), /*index in att_types array of
          "                         type with which device was last
          "                         attached*/
  2 down_slot fixed bin(17),      /*position of upname for this file
          "                         in up array of next registry file*/
```

```
  2 alloc_type char(32),        /*use this type in calls to the
          "                       Reserver alloc$resource
          "                       entry*/
  2 up(N1),                     /*registry files pointing to this one*/
    3 uptype char(32),
    3 upname char(32),
  2 devices(N2),                /*entries for devices associated
          "                       with this registry file*/
    3 resource_name char(32),   /*name used in calls to the Reserver
          "                       and the Device Assignment Module*/
    3 profile_relp bit(18),     /*relp to device profile for this
          "                       device*/
    3 device_type fixed bin(17),
  2 att_types(N3),              /*special information for each type
          "                       by which this device may be known*/
    3 type_name char(32),
    3 ccm_type char(32),        /*type of CCM to be spliced in above
          "                       the DSM*/
    3 trace_down bit(1),        /*if ON, trace down to next registry
          "                       file.  Otherwise, stop here*/
    3 alloc_down bit(1),        /*if ON, must call Reserver to
          "                       allocate a device of type
          "                       down_type, and use returned
          "                       resource_name as down_name.
          "                       In either case, find next RF by
          "                       using down_type and down_name*/
    3 look_only bit(1),         /*keep tracing down to other RFs
          "                       under trace_down control, but
          "                       only to compute CCM typename*/
    3 down_type char(32),       /*used as described above*/
    3 down_name char(32),       /*used as described above*/
    3 extra_mode char(32),      /*character string to be
          "                       concatenated with mode to be
          "                       passed to DCM*/
    3 dcm_type char(32),        /*used as type in attach call to
          "                       DCM if trace_down is OFF or
          "                       look_only is ON*/
    3 dcm_name char(32),        /*used as ioname2 of attach call to
          "                       DCM if trace_down is
          "                       OFF or look_only is ON*/
  2 free_storage area((15000));
```

These files are chained together in a bi-directional threaded list. The level number of a file indicates the general class of device: A level 1 I/O Registry File (RF) represents a GIOC channel and any wired-on devices (such as data sets). A level 2 RF represents a device connected to a level 1 device, and so on. The "down" direction is toward lower level numbers.

The RFs are organized into directories, which in turn are accessed via the I/O Registry File Directory Directory. In the following, the "type" of a file is its directory name, and the "name" of the file is the entry name of the file in that

directory. Several level k+1 devices may be associated with a single level k device. The "up" array contains the types and names of the files for devices in this category. An example of such a multiplicity of devices is a remote IBM System 360 Model 20. This remote computer (level 2) is connected to Multics with a telephone line and a Bell 201 data set (level 1). Attached to the remote computer might be a printer and a card reader (both level 3).

For various reasons, it may be convenient to consider several physical devices as a single device, and therefore to assign single Registry File to them. For example, a full-duplex typewriter channel requires two GIOC channels to handle a single data set and typewriter. The "devices" array of the RFs contains the name of each of the devices, their device type, and a relative pointer to the profile for the device.

When a device is attached, several different type arguments may be used. (The type corresponds to the type defined above, the name of a directory). Most of the directories and files will have several names to permit such calls. Different types may imply different chaining of files, the attachment of a different DCM and a different CCM type, and different modes for the DCM. The att_types array permits such handling. There is an entry in that array for each of the possible type arguments with which the device may be attached. For a detailed discussion of the chaining, see Section BF.2.23 (the Attachment Module).

Calls to the I/O Registry File Maintainer

The following describes the calls accepted by the I/O Registry File Maintainer. The information in the arrays in the RFs is readable by any program having access to the file, but only a few parts of the RF may be modified. (The Attachment Module modifies other parts of the RFs, but does this directly without calling the IORFM).

The following declarations hold for all of the calls:

```
dcl type char(32),          /*name of directory for file*/
    name char(32),          /*name of file within that
                              directory*/
    cstatus bit(18);        /*status for this call*/
```

Registry File Creation

In order to create a Registry File, the following call is made. A new segment will be created in the appropriate directory, and the segment will be initialized by copying a prototype file into it. The call is:

```
call iorfm$create(type,name,cstatus);
```

This call creates a file with name name in directory with name type. If such a file already exists, then set bit 2 of cstatus and return. If a file with name "prototype" exists in that directory and is accessible to this user, and create a new file with name name and initialize it with a copy of the prototype, and return. If the prototype file is non-existent or inaccessible, set bit 1 of cstatus and return.

This call may be used to create temporary Registry Files. An example of such usage is the dialup of an IBM 1[illegible] teletypewriter. Such machines have insufficient identificati to associate them automatically with a particular file, so a temporary file is created with a default device profile. If the temp_rf bit is ON in the file (as it would be in this case), the RF is destroyed by the Attachment Module when the device is detached. This call may also be used to create certain types of permanent file, such as Registry Files for standard tape reels. Other permanent files will be created without use of the IORFM by certain priveleged users.

Device Information

In order to get information on the devices associated with the file, the following call may be made:

```
call iorfm$get_devices(type,name,nwanted,nreturned,
     listptr,cstatus);
dcl nwanted fixed bin(17),    /*number of elements of
                               devices array wanted*/
    nreturned fixed bin(17), /*number actually
                               returned. nreturned
                               Always ≤ nwanted*/
    listptr ptr,
    1 list(nwanted) based(listptr),
      2 resource_name char(32),
      2 profile ptr,
      2 device_type bit(18);
```

If the RF in directory type with name name is inaccessible or non-existent, then set bit 1 of cstatus and return. Otherwise, return the information in the structure declared above. If the validation level of the caller is not in the access bracket of the indicated RF, a copy of the RF is made with an access range equal to the validation level of the caller. The profile pointers point to the copy in this case. In any case, these pointers are computed using the profile relative pointers in the original file. Because of the copying, it is possible for a non-priveleged user to read parts of a Registry File without being allowed to modify any entries in it.

As an example of the use of this call, a DCM may need to get at the device profile for one of the channels it is handling. It would issue this call for the level 1 RF, search for the proper device_type, and then access the profile using the corresponding

pointer.

Upnames

In order to find the names of the files above this one, the following call is made:

```
call iorfm$get_ups(type,name,nwanted,nreturned,up_ptr,cstatus);
dcl nwanted fixed bin(17),          /*as above*/
    nreturned fixed bin(17),        /*as above*/
    up_ptr ptr,
    1 up(nwanted) based(up_ptr),
      2 uptype char(32),
      2 upname char(32);
```

If the file is non-existent or inaccessible, set bit 1 of cstatus and return. If nwanted is less than 1, set bit 3 of cstatus and return. Otherwise, set nreturned equal to min(nwanted,rf.nup). Copy the first nreturned elements of rf.up into up_ptr->up and return. This call may be used for tracing through a sequence of RFs given the one with lowest level number.

Per-Type Information

If the name of a RF is known, and a type with which it might be reached are known, the following call may be used to find the index of the att_types array corresponding to type

```
call iorfm$search_types(type,name,index,cstatus);
dcl index fixed bin(17);
```

If the file is inaccessible or non-existent, set bit 1 of cstatus and return. Otherwise, search the att_types array for a typename equal to type. Store the index of that element of the array in index and return. (If no such typename is found, set bit 4 of cstatus and return: there is an error in the Registry File).

This call may be used by an outer module that knows the type and name of an RF and wishes to look at the next RF lower in the chain. The index returned may be used in the following call, which is used to get the chaining information for the RF for the type:

```
call iorfm$get_type(type,name,index,ptr,cstatus);
dcl index fixed bin(17),
    ptr ptr,
    1 type_entry based(ptr),
      2 call_type char(32),
      2 ccm_type char(32),
      2 dcm_type char(32),
      2 dcm_name char(32),
      2 down_type char(32),
      2 down_name char(32),
```

```
        2 extra_mode char(32),
        2 down_slot fixed bin(17),
        2 trace_down bit(1),
        2 alloc_down bit(1),
        2 look_only bit(1);
```

For comments on the elements of the type_entry structure, see the declaration of the RF, above. For more details, see BF.2.23. If index is zero, the value of present-type-index in the RF is used as index. Otherwise, the index should be derived from a call to iorfm$search_types or be the index of a loop. (The latter would be the case if the caller wanted information on all of the elements of the att_types array). If the file is inaccessible ornon-existent, set bit 1 of cstatus and return. If index is negative or greater than rf.ntypes, set bit 3 of cstatus and return. Otherwise, return the information in the structure.

Calls for DCMs

There are two calls that store information into Registry Files. These calls are intended to be used by DCMs managing devices that may dial into the system. Such a DCM must be able to associate the level 1 and level 2 RFs. In order for the Attachment Module to trace through these RFs, the down_name, up_type, and up_name entries in the RFs must be filled in.

The following call stores a value into a down_name entry:

```
call iorfm$set_down_name(type,name,index,down_name,cstatus);
dcl fordex fixed bfor(17),
    down_name char(32);
```

If the file is non-existent or inaccessible, set bit 1 of cstatus and return. If the user does not have write permission for the file from the ring whose number equals the validation level of the caller, set bit 5 of cstatus and return. If the index is too large or to small, set bit 3 of cstatus and return. Otherwise, store down-name in rf.att_types(index).down_name and return.

In order to establish the up link in the file below, the following call is provided:

```
call iorfm$set_up(type,name,index,uptype,upname,cstatus);
dcl index fixed bin(17),
    uptype char(32),
    upname char(32);
```

If the file is inaccessible or non-existent, set bit 1 of cstatus and return. If index is out of range, set bit 3 of cstatus and return. If the user does not have write permission for the file from the ring whose number is equal to the validation level at the time of the call, set bit 5 of cstatus and return. Otherwise, store uptype in rf.up(index).uptype and store upname in rf.up(index).upname and return.

Summary of Cstatus Bits

1 File inaccessible or non-existent
2 File already exists
3 Number out of range
4 Typename not found
5 File not writable