To: Distribution

From: N.S.Davids

Date: April 2, 1981

Subject: Changes in the MRDS submodel interface

Send comments by one of the following means:

By Multics mail (on System M)
Davids.Multics (nsd.m)

By Telephone:
HVN 341-7790 or 602-249-7790

By Continuum (method of choice):
Link to transaction 298 (subject: dsmd_changes) in the mrds_sec meeting.

INTRODUCTION

This MTB discuses the deficiencies in the current subroutine level user interface for MRDS submodels (dsmd_) based in part on changes required by the new security approach [1]. It then discribes the proposed solution which includes the implementation of a new subroutine level interface. The user documentation for this new interface is included.

DEFICIENCIES IN THE CURRENT INTERFACE (dsmd_):

1) Currently the dsmd_ entries get_dsm_header, get_dsm_relations and get_relation all return pointers to structures which are also used by mrds modules to communicate with each other. As these communication needs change (submodel security and long submodel names) the format of these structures must change which would break application programs using them.

2) The current structures do not allow for expansion, they lack both version numbers and a way for the user to tell the system which version of the structure is expected back. There is no way using the current structures and entry points to return relation and attribute access information to the user.

3) The entries that return a pointer to a structure, allocate that structure in a user supplied area. The area and simfault conditions are not being trapped.

4) Because the iocb pointer to the submodel vfile_ is stored as internal static only 1 submodel may be open at a time.

5) Implementation of the new security features of MRDS requires that the authorization of the requestor and the security state of the database be checked before any information about the model (relation and attribute names) be returned.

6) The length of submodel relation names is based on the length of an entry name, this restriction is not necessary since the submodel relation names never act as entry names.

PROPOSED SOLUTION:

The historical imperative requires that the current dsmd_ interface (with bugs fixed) remain in place. In order to provide for extensibility and multiple submodel openings an entire new set of entry points which at least duplicate the existing entry points in function will have to be written. Rather than having two independent sets of entry points with different calling sequences in the same interface a new interface msmi_ (MRDS submodel interface) will be written.

The changes in the mrds submodel interface will therefore consist of two parts, first necessary changes to dsmd_ to correct bugs, screen out model information in a secure database and handle the new longer submodel relation and attributes names; and second the implementation of a new interface (msmi_) which will not have the problems found in dsmd_.

Changes for dsmd_

The include files which declared the structures described in the user manual will be removed from all internal mrds code (except the user interface). The entry points that use these structures will translate from the internal mrds structure to this user interface structure.

Version 5 submodels (MR9.0) allow relation and attribute aliases to have up to 64 characters while the old structures can only accommodate 32. In the event that a name is longer than 32 characters The error code mrds_error_$name_too_long will be set and the first 31 characters and an "*" will be returned to the user. For version 4 or earlier submodels this change will be completely transparent, since names have a maximum length of 32 characters.

If the database is secured then those structure elements which contain information about the model (database path, relation names, and attribute names) will contain the null ("") character string if the caller is not a DBA. The elements will contain model information if the caller is a DBA or the database is not secure.

The entire dsmd_ module will be marked obsolete in the manual but will continue to be supported. It cannot just be undocumented because the effects of names longer than 32 characters and security must be described.

An unrelated modification which should be done at the same time would be the removal of the entries force_close_dsm get_mode, get_submodel_pn, validate_rel, and validate_rel_close from the dsmd_ module. These entries were undocumented during the last release and except for validate_rel no one complained. The validate_rel entry is being replaced with a set of dsl_ entries that are more complete [2].

User specification for the msmi_ module

Name: msmi_

This is a subroutine interface to the MRDS submodel data structure. The submodel data structure is created by the create_mrds_dsm command and may be displayed by the display_mrds_dsm command. This interface replaces the obsolete dsmd_ interface.

Entry: msmi_$close_submodel

This entry disassociates an opening name and a submodel to prevent further access to that submodel through that opening_name.

Usage:

```
declare msmi_$close_submodel entry (char (*), fixed bin
     (35));

call msmi_$close_submodel (opening_name, code);
```

where:

1. opening_name (input) (char (*))
   Is the name identifying the submodel opening.

2. code (output) (fixed bin (35))
   Is a standard error code.

Notes:

The submodel - opening_name association must already have been made by a successful call to msmi_$open_submodel. If the opening_name is not known the error code mrds_error_$open_name_not_known is returned.

Entry: msmi_$get_attribute_data

This entry returns the attribute information for the given relation.

Usage:

```
declare msmi_$get_attribute_data entry (char (*), char (*),
     ptr, fixed bin, ptr, fixed bin (35));

call msmi_$get_attribute_data (opening_name, rel_name,
     area_ptr, str_version, attribute_data_ptr, code);
```

where:

1. opening_name (input) (char (*))
   Is the name identifying the submodel opening.

2. rel_name (input) (char (*))
   Is the name of the relation for which attribute data is desired.

3. area_ptr (input) (ptr)
   Is a pointer to a freeing area where the mrds_dsm_attribute_data structure will be allocated.

4. str_version (input) (fixed bin)
   Is the version of the mrds_dsm_attribute_data structure that is to allocated.

5. attribute_data_ptr (output) (ptr)
   Is a pointer to the allocated structure.

6. code (output) (fixed bin (35))
   Is a standard error code.

Notes:

The submodel - opening_name association must already have been made by a successful call to msmi_$open_submodel. If the opening_name is not known the error code mrds_error_$open_name_not_known is returned.

If the area pointed to by the area_ptr parameter is too small for the mrds_dsm_attribute_data structure to be allocated in it the error code error_table_$area_too_small is returned. If the area_ptr parameter is null the error code error_table_$badcall is returned. If the area is not a freeing area the error code mrds_error_$not_freeing_area will be returned.

The following is version 1 (currently the only version) of the mrds_dsm_attribute_data structure (see Appendix F for the include file mrds_dsm_attribute_data.incl.pl1). If the str_version parameter refers to a version of the mrds_dsm_attribute_data structure that is not supported or does not exist the error code error_table_$unimplemented_version will be returned.

```
dcl 1 mrds_dsm_attribute_data based
            (mrds_dsm_attribute_data_ptr) aligned,
      2 version fixed bin,
      2 number_of_attributes fixed bin,
      2 attributes (mrds_dsm_attribute_data_num_atts refer
            (mrds_dsm_attribute_data.number_of_attributes)),
        3 submodel_attribute_name char (64),
        3 model_attribute_name char (32),
        3 read_access bit (1) unal,
        3 modify_access bit (1) unal,
        3 null_access bit (1) unal,
        3 mbzl bit (33) unal;
```

where:

1. version
   Is the version of the structure.

2. number_of_attributes
   Is the number of attributes in submodel relation view.

3. submodel_attribute_name
   Is the name of the attribute in the submodel.

4. model_attribute_name
   Is the name of the attribute in the model.

5. read_access
   Set to "1"b if the submodel has read access set for the attribute.

6. modify_access
   Set to "1"b if the submodel has modify access set for the attribute.

7. null_access
   Set to "1"b if the submodel has null access set for the attribute.

8. mbz1
   Set to "0"b.

If the submodel refers to a secure database and the user calling msmi_$get_attribute_data is not a database administrator for the database then the value of model_attribute_name will be null.

If null_access has a value of "1"b then both read_access and modify_access will have values of "0"b.

Entry: msmi_$get_relation_data

This entry returns information about each relation in the submodel.

Usage:

```
declare msmi_$get_relation_data entry (char (*), ptr, fixed
     bin, ptr, fixed bin (35));

call msmi_$get_relation_data entry (opening_name, area_ptr,
     str_version, relation_data_ptr, code);
```

where:

1. opening_name (input) (char (*))
   Is the name identifying the submodel opening.

2. area_ptr (input) (ptr)
   Is a pointer to a freeing area where the mrds_dsm_relation_data structure will be allocated

3. str_version (input) (fixed bin)
   Is the version of the mrds_dsm_relation_data structure that is to be allocated.

4. relation_data_ptr (output) (ptr)
   Is a pointer to the allocated structure.

5. code (output) (fixed bin (35))
   Is a standard error code.

Notes:

The submodel - opening_name association must already have been made by a successful call to msmi_$open_submodel. If the opening_name is not known the error code mrds_error_$open_name_not_known is returned.

If the area pointer to by the area_ptr parameter is too small for the mrds_dsm_relation_data structure to be allocated in it the error code error_table_$area_too_small is returned. If the area_ptr parameter is null the error code error_table_$badcall is returned. If the area is not a freeing area the error code mrds_error_$not_freeing_area will be returned.

The following is version 1 (currently the only version) of the mrds_dsm_relation_data structure (see Appendix F for the include file mrds_dsm_relation_data.incl.pl1). If the str_version parameter refers to a version of the mrds_dsm_relation_data structure that is not supported or does not exist the error code error_table_$unimplemented_version will be returned.

```
dcl 1 mrds_dsm_relation_data based
          (mrds_dsm_relation_data_ptr) aligned,
      2 version fixed bin,
      2 number_of_relations fixed bin,
      2 relations (mrds_dsm_relation_data_num_rels refer
            (mrds_dsm_relation_data.number_of_relations)),
        3 submodel_relation_name char (64),
        3 model_relation_name char (32),
        3 append_access bit (1) unal,
        3 delete_access bit (1) unal,
        3 null_access bit (1) unal,
        3 mbz1 bit (36) unal;
```

where:

1. version
   Is the version of the structure.

2. number_of_relations
   Is the number of relations in the submodel.

3. submodel_relation_name
   Is the relation name defined in the submodel.

4. model_relation_name
   Is the corresponding name of the relation as defined in the model.

5. append_access
   Set to "1"b if the submodel has append access set for the relation.

6. delete_access
   Set to "1"b if the submodel has delete access set for the relation.

7. null_access
   Set to "1"b if the submodel has null access set for the relation.

8. mbz1
   Set to "0"b.

If the submodel refers to a secure database and the user calling msmi_$get_relation_data is not a database administrator for the database then the value of model_relation_name will be null.

If null_access has a value of "1"b then both append_access and delete_access will have values of "0"b.

Entry: msmi_$get_submodel_info

This entry returns general information about the submodel.

Usage:

```
declare msmi_$get_submodel_info entry (char (*), ptr, fixed
     bin, ptr, fixed bin (35));

call msmi_$get_submodel_info (opening_name, area_ptr,
     str_version, submodel_info_ptr, code);
```

where:

1. opening_name (input) (char (*))
   Is the name identifying the submodel opening.

2. area_ptr (input) (ptr)
   Is a pointer to a freeing area where the mrds_dsm_submodel_info structure will be allocated.

3. str_version (input) (fixed bin)
   Is the version of the mrds_dsm_submodel_info structure that is to be allocated.

4. submodel_info_ptr (output) (ptr)
   Is a pointer to the allocated structure.

5. code (output) (fixed bin (35))
   Is a standard error code.

Notes:

The submodel - opening_name association must already have been made by a successful call to msmi_$open_submodel. If the opening_name is not known the error code mrds_error_$open_name_not_known is returned.

If the area pointer to by the area_ptr parameter is too small for the mrds_dsm_submodel_info structure to be allocated in the error code error_table_$area_too_small is returned. If the area_ptr parameter is null the error code error_table_$badcall is returned. If the area is not a freeing area the error code mrds_error_$not_freeing_area will be returned.

The following is version 1 (currently the only version) of the mrds_dsm_submodel_info structure (see Appendix F for the include file mrds_dsm_submodel_info.incl.pl1). If the str_version parameter refers to a version of the mrds_dsm_submodel_info structure that is not supported or does not exist the error code error_table_$unimplemented_version will be returned.

```
dcl 1 mrds_dsm_submodel_info based
            (mrds_dsm_header_info_ptr) aligned,
      2 version fixed bin,
      2 submodel_version fixed bin,
      2 database_path char (168),
      2 submodel_path char (168),
      2 date_time_created fixed bin (71),
      2 creator_id char (32);
```

where:

1. version
   Is the version of the structure.

2. submodel_version
   Is the version of the submodel data structure.

3. datebase_path
   Is the absolute path of the datamodel for which the submodel is defined.

4. submodel_path
   Is the absolute path of the submodel.

5. date_time_created

Is the Multics clock value (suitable for input into date_time_) for when the submodel was created.

6. creator_id
Is the ID of the user who created the submodel. It has the form of "Person.Project.Tag".

Entry: msmi_$open_submodel

This entry associates a submodel with an opening_name so that it can be used by other msmi_ entries. The same submodel may be associated with multiple opening names.

Usage:

```
declare msmi_$open_submodel (char (*), char (*), fixed bin
     (35));

call msmi_$open_submodel (opening_name, path, code);
```

where:

1. opening_name (input) (char (*))
   Is the name identifying the submodel opening. This name must be unique within the opening process (as determined by PL1 comparison rules), not only for submodel openings but any operation within the MDBM subsystem that takes an opening_name name. Multiple openings of the same submodel must have different opening_name names.

2. path (input) (char (*))
   Is the relative or absolute path (with or without the dsm suffix) of the submodel to be opened.

3. code (output) (fixed bin (35))
   Is a standard error code.

Notes:

The opening name can be any length and can be made up of any sequence of ascii characters. If the opening_name has already been used the error code mrds_error_$open_name_already_known is returned. If there is no room to create another opening_name the error code mrds_error_$too_many_open_names is returned. The exact number of opening_names depends on the length of the names already used but is large (> 1000).

REFERENCES

[1] The New MRDS Security Approach, MTB-501

[2] Changes to the MRDS dsl_ Subroutine Interface, MTB-504