May 2002

In this issue

3  A program to facilitate decentralized RACF administration
68  Inside IBM – IBM mainframe security since October 2000
82  RACF restructuring: coding
98  Converting from ACF2 to RACF
106 Remote security – inexpensive firewalls
112 Information point – reviews
115 RACF news

© Xephon plc 2002
RACF Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: fionah@xephon.com

North American office
Xephon
Post Office Box 350100
Westminster CO 80035-0100
USA
Telephone: (303) 410-9344

RACF Update on-line
Code from RACF Update, and complete issues in Acrobat PDF format, can be downloaded from http://www.xephon.com/racf; you will need to supply a word from the printed issue.

Subscriptions and back-issues
A year’s subscription to RACF Update (four quarterly issues) costs £190.00 in the UK; $290.00 in the USA and Canada; £196.00 in Europe; £202.00 in Australasia and Japan; and £200.50 elsewhere. The price includes postage. Individual issues, starting with the August 1999 issue, are available separately to subscribers for £48.50 ($72.75) each including postage.

Editor
Fiona Hewitt

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Contributions
When Xephon is given copyright, articles published in RACF Update are paid for at £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above or download a copy of our Notes for Contributors from http://www.xephon.com/nfc.

© Xephon plc 2002. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
A program to facilitate decentralized RACF administration

This article describes the MCINTY program, which was developed to help with decentralized RACF administration.

The MCINTY program was developed to help with decentralized RACF administration. This can be difficult to implement using the standard RACF commands because their authorization checks are rather inflexible, don’t provide enough granularity, and have no easy dialog interface.

The program addresses these problems and also enables the installation to easily store and retrieve its own additional information in RACF profiles by using the userdata fields. These fields are extremely useful but cannot be accessed using any IBM-supplied RACF commands; they’re specifically defined by IBM for installation use.

USE RDATA
Details of the userdata structures can be found in the section entitled ‘Special fields’ below. Note that USERDATA is completely different and separate from INSTALLATION DATA, which is displayed and maintained by the standard RACF commands.

Userdata can be stored in any profile in any class, not just user profiles. Retrieved fields are put into CLIST variables, which makes it easy to implement a dialog-based administration interface using CLIST or REXX.

Decentralized administrators can be given the authority to retrieve specific fields in profiles to which they have ‘MCINTY’ access by extending the standard RACF controls while still retaining sufficient control to ensure the integrity of the system and the data.

Only user-defined extensions (held in userdata fields) can be UPDATED, whereas READ access can selectively be given to any information the installation chooses (above and beyond that normally allowed using standard RACF commands).
MCINTY FEATURES
The main features of MCINTY are as follows:

- It can retrieve standard IBM-defined RACF fields from any segment.
- It runs under TSO/E for use with CLIST/REXX to provide a dialog-based administration interface. It is more efficient than trapping the output from RACF commands, being both much faster and unaffected by changes in display format.
- Retrieved data is put into CLIST variables and, optionally, written to the terminal.
- It can retrieve/update userdata fields in the base segment to allow the installation to store its own data in the RACF database.
- Authorization checking is performed to ensure that the caller is authorized to make the request, using field level access or authority over the owning group. Details of the authorization required can be found below in the section entitled ‘Authorization’.
- It uses standard, documented IBM interfaces for compatibility with system upgrades. No changes are required to exploit new fields in the RACF templates.

COMMAND SYNTAX
To understand the command syntax, you need to know the abbreviations for the MCINTY functions. These are shown in Figure 1.

The command syntax is as follows:

- **Function** – GET | ADD | REP | DEL
  - **GET**. Retrieve standard fields or userdata fields.
  - **ADD**. Add a userdata field.
  - **REP**. Replace a userdata field (add if not there).
  - **DEL**. Delete userdata fields.

Note that a ‘GET’ operation can reference multiple field names,
whereas the update operations (ADD, REP, DEL) work only on one field.

- **Profile.** The full profile name.
- **Fields.** One or more field names to be retrieved/altered in the profile.
  - For non-userdata requests, field names must be valid existing field names as documented by IBM. Userdata requests can supply any field name.
  - If ‘fields’ is omitted for a GET USERDATA request, then ALL USERDATA fields are retrieved.
  - Field names can be suffixed with a data-conversion character to convert fields held internally in non-character format to displayable characters.

The valid suffixes are as follows:

- .X Convert from hex to character
- .B Convert from binary to character
- .P Convert from packed to character.

For example:

```
MCINTY 'function'          Abbreviation
PROF('profile')            PR
FIELDS('fields')           FI
CLASS('class')             CL
SEGMENT('segment')         SEG
DATA('data')               DA
FLAG('flag')               FL
USERDATA                   USR
NORGROUP                   NORG
NOLIST | LIST               NOL
NOMSG | MSG                 NOM
GENERIC                    GEN
TRACE                      TR
DEBUG                      DEB
```

*Figure 1: Function abbreviations*
See the *OS/390 Security Server (RACF) – Macros and Interfaces* manual for the field names and formats.

- **Class.** Must be a valid active class (default=USER).

- **Segment.** Must be a valid RACF segment name (default=BASE). For userdata operations, only the BASE segment is supported.

- **Data.** Used for ADD/REP/DEL operations on userdata or GET for non-userdata.
  - For ADD/REP it is the data to be associated with the userdata field name. It can be a quoted string or a simple string.
  - May be used on ‘DEL’ operations to delete specific occurrences when there can be multiple entries with the same ‘USRNM’ value.
  - May also be used to GET a specific occurrence in a repeat group which is part of standard RACF data (see ‘Repeat groups’ below).

- **Flag.** Used for ADD/REP operations on USERDATA. This is the value to be assigned to the USRFLG field and must be a number from 0-255. It is converted to binary before storing. The default is 0.

- **USERDATA.** Use with the ‘GET’ operation.
  - Indicates that the field names specified are userdata fields and not part of the standard RACF template.
  - The value of the USRNM field is used as the name of the userdata entry.
  - See ‘Special fields’ below for a description of userdata.

- **NORGROUP.** Use to force a field to be formatted as a non repeat-group.
  - NORGROUP may be necessary in very rare cases where a
simple field contains binary data that looks like a repeat-
group when retrieved, in which case it will be displayed and
returned in the CLIST variable incorrectly.

– Be warned that using NORGROUP for a field that is a repeat-
group will either cause the data to be incorrectly formatted or
cause message MCI10E to be issued.

– See ‘Special fields’ below for a description of repeat-groups.

- **LIST/NOLIST.** Display (LIST) or not (NOLIST) retrieved data at
  the terminal. The default is LIST.

- **MSG/NOMSG.** Display (MSG) or not (NOMSG) status messages
  at the terminal. NOMSG also suppresses the display of retrieved
  data as if NOLIST were coded. The default is MSG.

- **GENERIC.** Search for a generic profile even if the profile name
  in PROF(‘profile’) contains no generic characters.

- **TRACE.** Gives a diagnostic trace of authorization checks made
  by the program.

- **DEBUG.** Deactivates the recovery routine to allow the MVS
  symptom dump to be taken.

OUTPUT

**CLIST variables**

CLIST variables are always created for GET requests and can be
accessed directly by reference in CLIST/REXX.

*Simple fields (non repeat-group entries)*

For each simple field retrieved, a CLIST variable is created with the
same name as the field.

For example, when the command line includes FIELDS(NAME), the
NAME field from a user profile is put into a CLIST variable called
NAME.

If duplicate fields exist (in USERDATA it’s possible to ADD multiple
fields with the same name), the last field found is the one put into the
variable; but all fields found are listed at the terminal and can be
SYSOUTTRAP’d (OUTTRAP in REXX) if required.

Repeat-group fields
Where a field is a member of a repeat-group, the number of occurrences
retrieved (the ‘count’) is put in a variable with the same name as the
field specified, and a numeric suffix is appended to the variable name
to uniquely identify each occurrence of the repeat-group.

For example, if a user was connected to ten groups and the command
line included ‘FIELDS(CGGRPNM) ’ the following variables would
be created:

‘CGGRPNM’ = 10 (the number of CGGRPNM occurrences)
‘CGGRPNM1’ = the name of the 1st connect group
‘CGGRPNM2’ = the name of the 2nd connect group
   etc
‘CGGRPNM10’ = the name of the 10th connect group.

Userdata fields
For each occurrence retrieved a variable is created with a name equal
to the contents of the Userdata Name field (USRNM), and its value is
the contents of the Userdata Data field (USRDATA). In addition, the
Flag value (USRFLG) is put into a variable with the same name plus
a suffix of ‘F’.

For example, if the following command had been issued to add
userdata fields to the user profile for USER1:

```
MCINTY ADD PR(USER1) FI(JOBCAT) DATA('OPERATIONS') FLAG(3)
```

then issuing the following command:

```
MCINTY GET PR(USER1) USERDATA
```

the following variables would be created:

Variable: JOBCAT   Contents: OPERATIONS
Variable: JOBCATF  Contents: Ø03
Specific occurrences
Where a specific occurrence of a repeat-group has been requested (using the DATA() parameter), a single numbered variable is created in the same way as for repeat-group fields, even if only one is returned, and the count variable is set to one.

If no occurrence is found to match the value in DATA(), the count variable will be zero and no other variables created. (See the example in the ‘Repeat-groups’ section in ‘Special fields’ above.)

Terminal output
This section describes information optionally displayed at the terminal, which can be suppressed with the NOLIST option.

Non-userdata display
For a GET for non-userdata fields, all the data is displayed for each field in turn. If retrieving repeat-group information, all occurrences of the first field are displayed and then all occurrences of the next field, and so on. Each line displayed consists of the field name followed by the field data.

For example, if the ADMIN group contained two users called USER1 and USER2, both connected with AUTH=USE, issuing the command:

```
MCINTY GET PROF(ADMIN) FIELDS(USERID,USERACS.B) CLASS(GROUP)
```

would produce the following:

```
USERID USER1
USERID USER2
USERACS 16
USERACS 16
```

Note that the USERACS fields (connect attributes) are converted from the internally held binary values (of X‘10’ in this example) to displayable numbers because of the .B suffix used.

Userdata display
For a GET request for userdata, the fields requested are displayed in the order requested, or, if no specific fields are requested, all fields are displayed in the order found. The information displayed for each field
consists of the field name (USRNM), followed by the data (USRDATA),
followed by the value (in decimal) of the user flag (USRFLG).

For example, if the following commands had been issued to add
userdata fields to the user profile for USER1:

```
MCINTY ADD PR(USER1) FI(JOBTITLE) DATA('SYSTEMS PROGRAMMER')
MCINTY ADD PR(USER1) FI(JOBCAT) DATA('OPERATIONS') FLAG(3)
```

then issuing the following command:

```
MCINTY GET PR(USER1) USERDATA
```

would produce:

```
JOBTITLE SYSTEMS PROGRAMMER 000
JOBCAT OPERATIONS 003
```

EXAMPLES

This section gives examples of how MCINTY can be used:

- List users connected to a group and their connect attributes
  
  ```
  MCINTY GET PROF(group_name) FIELDS(USERID,USERACS.B) CLASS(GROUP)
  ```

- List a user’s name and all the groups the user is connected to:
  
  ```
  MCINTY GET PROF(userid) FIELDS(NAME,CGGRPNM)
  ```

- Replace (or add if not there) a userdata field called JOBTITLE in
  Fred’s user profile. Class defaults to USER.

  ```
  MCINTY REP PROF(FRED) FIELDS(JOBTITLE) DATA('SYSTEMS PROGRAMMER')
  ```

- Retrieve the JOBTITLE userdata field from Fred’s user profile:

  ```
  MCINTY GET PROF(FRED) FIELDS(JOBTITLE) USERDATA
  ```

- Retrieve all userdata from a user profile, then delete all their
  userdata:

  ```
  MCINTY GET PROF(userid) USERDATA
  MCINTY DEL PROF(userid) USERDATA
  ```

- Sample REXX to list userid, associated name, and connect
  authority in group TECHSUP:

  ```
  /* rexx */
  "MCINTY GET PROF(TECHSUP) FIELDS(USERID,USERACS.B) CLASS(GROUP)
  NOMSG"
  ```
AUTHORIZATION

Authorization required

The authorization to perform an operation is checked as follows:

- System SPECIAL users can perform any operation. (This is optional – see the section entitled ‘Customization’ below.)

- The following group-authority will allow a user to perform the actions described:
  
  - GROUP-AUDITOR. Read any field in any profile owned by the group or any subgroup.
  
  - GROUP-SPECIAL. As for GROUP-AUDITOR, plus update any USERDATA field in any profile owned by the group or any subgroup.

- AUTH=CONNECT.
  
  - Read any field in user-profiles owned directly by that group only.
  
  - Read any field in that group profile only.

- READ/UPDATE access to ‘Authprof’ in the ‘FIELD’ class enables the caller to read a field (standard and userdata) or to update a userdata field, in any resource profile in the class specified in the Authprof. (See below for a description of ‘Authprof’.)

- The above checks are made by MCINTY explicitly before attempting to access the data. If all of these checks fail, standard field-level access checking will be used on the call to RACF to access the database.
Authprof
Authprof is a special profile used by this program to control access to specific fields within any resource profile. It is defined in the standard RACF FIELD class, and follows the standard naming conventions for field-level access to all segments, except for USERDATA which is not a segment name but has special meaning in the implementation of MCINTY access checking.

Profile format
The format of the profile is:

    class.segment.field

or

    class.USERDATA.field

where:

- ‘Class’ is the name of the RACF class in which the profile is defined.
- ‘Segment’ is the name of the segment containing the field to be accessed. For userdata fields, the segment name is always USERDATA.
- ‘Field’ is the name of the field to be accessed.

An example profile might be USER.BASE.DFLTGRP

Special use of ‘?’
Generics can be used to allow access to a range of segments/fields, but special use is made of the ‘class.?’ and ‘class.segment.?’ profiles.

Access to these profiles enables a user to process as follows:

- ‘Class.?’: All fields in any segment in the specific class (this includes all userdata fields).
- ‘Class.segment.?’: All fields in the segment and class (this excludes userdata fields).
- ‘Class.USERDATA.?’: All userdata fields in the named class.

This is to avoid the need for an administrator with wide scope having to be put on the access list for all individual fields.
For example, READ access to ‘DATASET.BASE.?’ allows a user to read any field in the base segment of any dataset profile, even if a more specific ‘field-level’ profile exists, for example ‘DATASET.BASE.OWNER’.

Note that you should take care when implementing field-level access control, as it can change the behaviour of standard RACF commands by allowing/restricting access to specific fields on a GLOBAL basis.

Profiles of the form ‘class.USERDATA.field’ will not affect standard RACF commands, but ‘class.segment.field’ may.

RACF command processors and panels support field-level access checking only for fields in segments other than the BASE segments of RACF profiles. MCINTY performs field-level checking in all segments.

See the OS/390 Security Server (RACF) Security Administrator’s Guide for information on activating and using field-level access.

&RACUID
Placing &RACUID on the access list for an authorization profile in the FIELD class is supported, even if the class is not raclisted.

This is checked only when users perform an operation on their own user profile.

It can only be used to give users access to fields (userdata or standard) in their own user profile, either for read or update.

Note that &RACUID does not work on generic Authprof profiles for userdata fields, eg ‘USER.USERDATA.*’. This is because USERDATA is not recognized as a valid segment name in normal RACF processing and is thus not supported by field level access checking as specified by FLDACC=YES on the ICHEINTY macro interface.

SPECIAL FIELDS

Repeat groups
A repeat group consists of one or more sequential fields within a
profile that can be repeated within that profile. A field that belongs to a repeat group is defined only once in the template, but can be repeated as many times as necessary within the actual profile. A count field precedes the repeat group in the profile, indicating how many of these groups follow.

A typical use of repeat groups is the list of groups and connect information in a user profile that the user is connected to.

The program automatically recognizes fields retrieved in repeat group format and displays each occurrence separately.

To select ALL occurrences, you should omit the DATA() parameter; a count variable and numbered variables will be created as described in the section on repeat groups in ‘CLIST variables’ above.

To select a specific occurrence within a repeat group (for example, to retrieve connect information from a user profile for a specific RACF group), specify in the DATA() parameter a value to be compared to the first named field in FIELDS(). When an occurrence is found with this field matching, values for the same relative occurrence will be retrieved for all other fields named in FIELD().

This is best explained by example:

```
MCINTY GET PR(FRED) FI(CGGRPNM,CGAUTHOR,CGAUTHDA.P) DATA(SYS1)
```

Each occurrence of the CGGRPNM repeat group is scanned for a match with ‘SYS1’. When one is found, the remaining repeat group fields (CGAUTHOR and CGAUTHDA) are scanned and the values from the same relative occurrence are retrieved.

This example retrieves the group name (CGGRPNM), connect owner (CGAUTHOR), and connect date (CGAUTHDA) for the connect entry ‘SYS1’ in FRED’s user profile (default CLASS=USER). The date is converted from packed to character because of the .P suffix.

MCINTY stores these fields in variables named CGGRPNM1, CGAUTHOR1, and CGAUTHDA1 respectively. Note the numeric suffixes on the variables created because the fields are repeat groups (see ‘Specific occurrences’ earlier). The CGGRPNM variable will contain the value 1 to indicate how many numbered variables were created. If FRED was not connected to group SYS1 then CGGRPNM would be zero and no other variable would be created.
USERDATA

The USERDATA field defined in the RACF templates is provided by IBM for installation use and exists in all profile types (RACF classes), not only user. It can be used to store additional information such as a user’s job title, pager number, e-mail address, a group’s function, etc.

It is a combination field that defines a repeat group where each occurrence within the repeat group contains the following three fields:

- **USRNM.** 8 characters; used as the name of the entry.
- **USRDATA.** 1-255 bytes; contains the data.
- **USRFLG.** 1 byte; can be used as a flag.

An additional field called USRCNT contains the number of USERDATA occurrences that exist in the profile.

MCINTY can be used to maintain these fields while providing selective control over who can read/update individual entries.

Installation exits or other programs can access the data using the ICHEINTY macro interface.

Figure 2 shows the format of data returned by an ICHEINTY request for ALL USERDATA occurrences.

Hidden fields

Hidden fields are supported to allow data that should not be displayed to be stored in a USERDATA entry. The field name should start with an ‘@’ sign, which indicates to MCINTY that it should not display any associated data when it is retrieved. In this case, this program will display each character of the field as a ‘?’ to indicate the length and presence of the field while not disclosing the contents. For example:

```
MCINTY ADD PROF(FRED) CLASS(USER) FIELDS(@PW) DATA(1234567)
```

IMPLEMENTATION

Installation

The program should be assembled and link-edited into an APF
authorized library (available in the linklist or TSO/E log-on proc) with AC=1, AMODE=31, RMODE=ANY, and NON-REENTRANT. It must be named as a COMMAND PROCESSOR in the IKJTSOxx member of SYS1.PARMLIB.

**Customization**

The following changes to default behaviour may be made if desired, and must be done before assembling the program.

**Message options**

The defaults for message and data display can be changed by modifying the value specified in the DEFAULT= parameter on the IKJKEYWD macros labelled KLIST (for data display) and KMSG (for data and messages).

**Authorization**

A System Special RACF user is normally allowed to perform any operation by the program, but you can disable this if required by uncommenting the instruction labelled SPECHK in subroutine SAUTHCHK and uncommenting the MNOTE that immediately follows it. To re-enable, simply comment out these two instructions again.

---

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field length</th>
<th>Field contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>usrcntl *</td>
<td>4</td>
<td>Length of USRCNT field (always=4)</td>
</tr>
<tr>
<td>USRCNT</td>
<td>4</td>
<td>Number of occurrences following</td>
</tr>
<tr>
<td>usrdlen *</td>
<td>4</td>
<td>Total length of ALL occurrences following</td>
</tr>
</tbody>
</table>

The following fields are repeated once for each occurrence:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Field length</th>
<th>Field contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>usrdcl *</td>
<td>4</td>
<td>Length of this occurrence</td>
</tr>
<tr>
<td>usrnm *</td>
<td>4</td>
<td>Length of USRNM field (always=8)</td>
</tr>
<tr>
<td>USRNM</td>
<td>8</td>
<td>Name of occurrence (installation defined)</td>
</tr>
<tr>
<td>usrdatl *</td>
<td>4</td>
<td>Length of USRDATA field (1-255)</td>
</tr>
<tr>
<td>USRDATA</td>
<td>1 to 255</td>
<td>Installation supplied data</td>
</tr>
<tr>
<td>usrflgl *</td>
<td>4</td>
<td>Length of USRFLG field (always=1)</td>
</tr>
<tr>
<td>USRFLG</td>
<td>1</td>
<td>Installation supplied value (0-255)</td>
</tr>
</tbody>
</table>

* Field names in lower case are used in MCINTY only and not defined in the RACF templates.

**Figure 2: Data returned by an ICHEINTY request**
Testing

During testing, it’s advisable to disable the authority MCINTY allows System Special users to ensure that the lower-level access checking is tested thoroughly (see the section on ‘Customization’ above). This is because the testing is likely to be done by a Special user, who would automatically be given access to everything without going through the access checking.

Use the TRACE option during testing to see the security checks that are being made. This is useful if you get the message “MCI07E: NOT AUTHORIZED TO class.segment.field”.

MESSAGES AND CODES

Return codes

Figure 3 gives details of the return codes. See the section on messages (below) for details of which return code is issued for a specific message.

Messages

All errors are accompanied by a message (which can be suppressed by the NOMSG option). Messages are in the format MCI\(nnx\) where \(nn\) is a number and \(x\) is I for Informational and E for Error. Return codes are shown in brackets, as in Error (rc).

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Request completed successfully</td>
</tr>
<tr>
<td>4</td>
<td>Profile not found</td>
</tr>
<tr>
<td></td>
<td>Userdata field not found</td>
</tr>
<tr>
<td>8</td>
<td>Insufficient authority for request</td>
</tr>
<tr>
<td>12</td>
<td>Command parse failed</td>
</tr>
<tr>
<td></td>
<td>Field name invalid</td>
</tr>
<tr>
<td></td>
<td>Segment name invalid</td>
</tr>
<tr>
<td></td>
<td>Syntax error in command parameters</td>
</tr>
<tr>
<td></td>
<td>Class invalid/inactive</td>
</tr>
<tr>
<td>16</td>
<td>ICHEINTY workarea too small (RACWA)</td>
</tr>
<tr>
<td></td>
<td>Other ICHEINTY error (message contains ICHEINTY reason)</td>
</tr>
<tr>
<td></td>
<td>Internal abend (message MCI99E is issued)</td>
</tr>
</tbody>
</table>

Figure 3: Return codes
Figure 4 shows the messages with their relevant errors and actions.

**MCI00I: USERDATA ‘action’**
Informational (0) – Indicates successful completion of ‘action’ requested.
Action – None required.

**MCI01E: PARSE FAILED RC=nnn**
Error (12) – Command parse failed. This is an internal error and indicates an 
error in the IBM TSO command parser. The return code from the parser is 
shown as nnn.
Action – Notify the Systems Programmer!

**MCI02E: CLASS INVALID OR INACTIVE**
Error (12) – A class specified in the CLASS parameter is either not defined 
to RACF or is not currently active.
Action – Use the SETROPTS command to verify the class is active.

**MCI03E: NO FIELD NAMES SPECIFIED**
Error (12) – A field name must be supplied in the FIELDS parameter for an 
ADD or REP request for USERDATA and for a GET request for non-
USERDATA.
Action – Correct the command parameters.

**MCI03E: ONLY 1 FIELD ALLOWED PER UPDATE**
Error (12) – Only one USERDATA field per invocation can be updated using 
ADD or REP.
Action – Correct the command parameters.

**MCI04E: PROFILE NAME MISSING**
Error (12) – For any request a PROFILE name must be supplied.
Action – Correct the command parameters.

**MCI04E: FLAG OUT OF RANGE (0-255)**
Error (12) – The value in the FLAG parameter, if specified, must be a 
number from 0 to 255 for a USERDATA update request.
Action – Correct the command parameters.

**MCI05E: PROFILE NOT FOUND**
Error (4) – The profile specified in the PROF parameter was not found.
Action – If it is a generic profile but does not contain any recognized generic 
characters then specify the GENERIC parameter.
Action – Check that the correct CLASS is specified or defaulted to.

**MCI05E: FIELD NAME INVALID**
Error (12) – The field name specified in FIELD is invalid or undefined. This 
only applies to a non-USERDATA request when a GET request for a field 
not defined in the RACF templates is attempted.
Action – Check the field name in Security Server – Macros and Interfaces .

*Figure 4 (part one): Messages, errors, and actions*
<table>
<thead>
<tr>
<th>Message ID</th>
<th>Message Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCI05E: SEGMENT NAME INVALID</td>
<td>Error (12) – Specified SEGMENT name not allowed for the specified profile type. Action – Check that the segment name is valid for the CLASS and PROFILE specified.</td>
<td></td>
</tr>
<tr>
<td>MCI05E: WORK AREA TOO SMALL, TRY FEWER FIELDS</td>
<td>Error (16) – The workarea in the program is too small for the amount of data requested. This is an internal limit set by the length of the getmained workarea which is currently 32K. Action – Either request less data to be returned or increase the size of the workarea in the program by changing the ORG statement immediately preceding the RACWAL label.</td>
<td></td>
</tr>
<tr>
<td>MCI05E: ICHEINTY RC=nnn REASON=nnn</td>
<td>Error (16) – An undetermined error occurred during a RACF database access request. Action – Check the code reported in the list of RETURN and REASON codes for the ICHEINTY interface in the <em>Security Server – Macros and Interfaces</em> manual.</td>
<td></td>
</tr>
<tr>
<td>MCI06E: NO USERDATA IN PROFILE</td>
<td>Error (4) – For a GET USERDATA request, none was found in profile named. Action – Put some in.</td>
<td></td>
</tr>
<tr>
<td>MCI06E: USERDATA FIELD NOT FOUND</td>
<td>Error (4) – For a GET or DEL USERDATA request a requested FIELD was not found in the profile. Action – If the field is expected to be there try listing all USERDATA to see if its name has been mis-spelled (omit the FIELD parameter).</td>
<td></td>
</tr>
<tr>
<td>MCI07E: NOT AUTHORIZED TO class.segment.field</td>
<td>Error (8) – The caller did not have sufficient access to perform the operation requested on the field named in the message. Action – Check they have the authority as described in the section entitled ‘Authorization’.</td>
<td></td>
</tr>
<tr>
<td>MCI07E: NOT ALL FIELDS RETURNED (ACCESS CHECK FAILED)</td>
<td>Error (8) – The caller did not have sufficient field level access to perform the operation requested on SOME of the fields, but no information is returned by RACF to indicate which particular fields. Action – Try one field at a time to determine which ones cause the error. These will produce the “NOT AUTHORIZED to class.segment.field” message.</td>
<td></td>
</tr>
<tr>
<td>MCI08E: USERDATA SUPPORTED IN BASE SEGMENT ONLY</td>
<td>Error (12) – USERDATA can be stored/retrieved only in the BASE segment of a profile. Action – Remove the SEGMENT parameter from the request.</td>
<td></td>
</tr>
</tbody>
</table>

*Figure 4 (part two): Messages, errors, and actions (continued)*
null
**THE CODE**

******************************************************************************
* DESCRIPTION *
******************************************************************************
* RACF PROFILE INFORMATION PROCESSOR COMMAND *
* *
* 1. UPDATE/RETRIEVE ENTRIES (OCCURRENCES) IN THE USERDATA *
* REPEAT-GROUP. *
* 2. RETRIEVE STANDARD RACF FIELDS FROM ANY SEGMENT. *
* *
* RETRIEVED DATA IS PUT INTO CLIST VARIABLES, AND OPTIONALLY *
* WRITTEN TO THE TERMINAL. *
* *
* AUTHORISATION CHECKING IS PERFORMED TO ENSURE THAT THE *
* CALLER IS AUTHORISED TO MAKE THE REQUEST, USING FIELD *
* LEVEL ACCESS OR AUTHORITY OVER THE OWNING GROUP. *
* *
******************************************************************************
* SYNTAX *
******************************************************************************
* INTY "FUNCTION" ABBREV. *
* PROF("PROFILE_NAME") PR *
* FIELDS("FIELD_NAME_LIST") FI *
* CLASS("RESOURCE_CLASS") CL *
* SEGMENT("SEGMENT_NAME") SEG *
* DATA("DATA_VALUE") DA *
* FLAG("USER_FLAG_VALUE") FL *
* USERDATA USR *
* RGROUP RG *
* NORGROUP NORG @MC7 *
* NLIST NOL *
* NOMSG NOM @MC2 *
* GENERIC GEN @MC2 *
* TRACE TR @MC5 *
* DEBUG DEB @MC6 *
* *
* "FUNCTION" - GET | ADD | REP | DEL *
* *
* GET - RETRIEVE RACF FIELDS OR USERDATA FIELDS. *
* ADD - ADD A USERDATA FIELD *
* REP - REPLACE A USERDATA FIELD (ADD IF NOT THERE) *
* DEL - DELETE A USERDATA FIELD *
* *
* | NOTE: A ‘GET’ OPERATION CAN REFERENCE MULTIPLE FIELD *
* | NAMES, WHEREAS THE UPDATE OPERATIONS CAN ONLY *
* | WORK ON ONE FIELD. *
* *
* "PROF" THE FULL PROFILE NAME
* "FIELDS" ONE OR MORE FIELD NAMES TO BE RETRIEVED/ALTERED IN
  THE PROFILE.
* FIELD NAMES MUST BE VALID EXISTING FIELD NAMES FOR
  NON-USERDATA REQUESTS.
* (SEE RACF SPL FOR FIELD NAMES)
* USERDATA REQUESTS CAN SUPPLY ANY FIELD NAME.
* FIELD NAMES CAN BE SUFFIXED WITH A ‘DATA-CONVERSION’
  CHARACTER TO CONVERT FIELDS HELD IN NON-CHARACTER
  FORMAT TO CHARACTER.
* VALID SUFFIXES ARE:-
  .X CONVERT FROM HEX TO CHARACTER
  .B CONVERT FROM BINARY TO CHARACTER
  .P CONVERT FROM PACKED TO CHARACTER
* EXAMPLE:-  FI(PASSDATE.P) RESULTS IN: 99365
  FI(PASSDATE.X) RESULTS IN: 99366F
  FI(FLAG2.B) RESULTS IN: 128
* "CLASS" MUST BE A VALID ACTIVE CLASS
  (DEFAULT=USER)
* "SEGMENT" MUST BE A VALID RACF SEGMENT NAME
  (DEFAULT=BASE)
  FOR ‘USERDATA’ OPERATIONS, ONLY THE BASE SEGMENT IS
  SUPPORTED.
* "DATA" USED FOR ADD/REP OPERATIONS ON USERDATA.
  THIS IS THE DATA TO BE ASSOCIATED WITH THE FIELD
  NAME. IT CAN BE A QUOTED STRING OR A SIMPLE STRING.
  ALSO USED ON ‘DEL’ OPERATIONS TO DELETE SPECIFIC
  OCCURRENCES WHEN THERE CAN BE MULTIPLE ENTRIES WITH
  THE SAME ‘USRNM’ VALUE.
  MAY ALSO BE USED TO RETRIEVE A SPECIFIC OCCURRENCE
  IN A REPEAT GROUP WHICH IS PART OF STANDARD RACF
  DATA (SEE ‘REPEAT GROUPS’ BELOW).
* "FLAG" USED FOR ADD/REP OPERATIONS ON USERDATA.
  THIS IS THE VALUE TO BE ASSIGNED TO THE USRFLG FIELD
  IT MUST BE A NUMBER FROM 0-255.
  (DEFAULT=X’00’)
* ‘USERDATA’ USE WITH THE ‘GET’ OPERATIONS ONLY.
  INDICATES THAT THE FIELD NAMES SPECIFIED ARE
  USERDATA FIELDS AND NOT PART OF THE STANDARD RACF
  TEMPLATE.
  THE CONTENTS OF THE ‘USRNM’ FIELD IS USED AS THE
  NAME OF THE USERDATA ENTRY.
* `RGROUP` USE TO FORCE A FIELD TO BE PROCESSED AS A REPEAT- @MC7
* GROUP. THIS OVERRIDES AUTOMATIC RECOGNITION AND IS @MC7
* NOT NORMALLY REQUIRED BUT IS PROVIDED TO COMPLEMENT @MC7
* THE `NORGROUP` PARAMETER. @MC7
* @MC7
* `NORGROUP` USE TO OVERRIDE AUTOMATIC REPEAT-GROUP RECOGNITION @MC7
* IN RARE CASES WHERE A RETURNED FIELD LOOKS LIKE A @MC7
* REPEAT-GROUP BUT IS NOT. @MC7
* WARNING: USING RGROUP/NORGROUP WRONGLY WILL GIVE @MC7
* ERROR MESSAGE MC110E OR PRODUCE INCORRECT @MC7
* OUTPUT. @MC7
* @MC7
* `NOLIST` SUPPRESS DISPLAY ON THE TERMINAL OF RETRIEVED DATA. * @MC7
* ERROR MESSAGES ARE STILL DISPLAYED. * @MC7
* @MC7
* `NOMSG` SUPPRESS DISPLAY ON THE TERMINAL OF ALL MESSAGES. @MC2
* @MC7
* `GENERIC` SEARCH FOR GENERIC PROFILE EVEN IF THE PROFILE @MC2
* NAME CONTAINS NO GENERIC CHARACTERS. @MC2
* @MC7
* `TRACE` GIVES DEBUGGING TRACE OF AUTHORISATION CHECKS. @MC5
* @MC7
* `DEBUG` TURNS OFF ESTAE FOR DEBUGGING PURPOSES. @MC6
* @MC7
*----------------------------------------------------------------------
* **AUTHORISATION**
* @MC7
* @MC7
* AUTHORISATION TO PERFORM AN OPERATION IS CHECKED AS FOLLOWS:
* @MC7
* @MC7
* 1. RACF ‘SPECIAL’ USERS CAN PERFORM ANY OPERATION.
* @MC7
* @MC7
* 2. READ/UPDATE ACCESS TO “AUTHPROF” IN CLASS ‘FIELD’ ENABLES
* THE CALLER TO READ A FIELD (STANDARD OR USERDATA) OR
* TO UPDATE A USERDATA FIELD, IN ANY RESOURCE PROFILE OF
* THAT CLASS.
* | “AUTHPROF” IS DESCRIBED BELOW.
* @MC7
* @MC7
* 3. THE FOLLOWING GROUP-AUTHORITY WILL ALLOW A USER TO
* PERFORM THE ACTIONS DESCRIBED:
* @MC4
* ‘GROUP-AUDITOR’ - READ ANY FIELD IN ANY PROFILE OWNED
* BY THE GROUP OR ANY SUBGROUP. @MC4
* ‘GROUP-SPECIAL’ - UPDATE ANY USERDATA FIELD IN ANY PROFILE
* OWNED BY THE GROUP OR ANY SUBGROUP. @MC4
* @MC4
* `AUTH=CONNECT` - ALLOWS THEM TO READ USER-PROFILES OWNED @MC5
* DIRECTLY BY THAT GROUP ONLY. @MC5
* - ALLOWS THEM TO READ THAT GROUP PROFILE ONLY
* @MC5
* “AUTHPROF”
THIS IS A SPECIAL PROFILE USED BY THIS PROGRAM TO CONTROL
ACCESS TO SPECIFIC FIELDS WITHIN ANY RESOURCE PROFILE.
THE FORMAT OF THE PROFILE IS:
“CLASS”.”SEGMENT”.”FIELD”
(EXAMPLE: USER.BASE.DFLTGRP)
“CLASS” - THE NAME OF THE CLASS IN WHICH THE RESOURCE
PROFILE IS DEFINED.
“SEGMENT” - THE NAME OF THE SEGMENT CONTAINING THE FIELD
TO BE ACCESSED.
FOR ‘USERDATA’ FIELDS, THE SEGMENT NAME IS
ALWAYS ‘USERDATA’.
“FIELD” - THE NAME OF THE FIELD TO BE ACCESSED.

GENERICS CAN BE USED TO ALLOW ACCESS TO A RANGE OF
SEGMENTS/FIELDS, BUT SPECIAL USE IS MADE OF THE
PROFILES “CLASS”.? AND “CLASS”.”SEGMENT”.?
ACCESS TO THESE PROFILES WILL ENABLE A USER TO PROCESS:-

“CLASS”.? - ALL FIELDS IN ANY SEGMENT IN THE SPECIFIC CLASS*
(This includes all Userdata fields)
“CLASS”.”SEGMENT”.? - ALL FIELDS IN THE SEGMENT AND CLASS*
(This excludes Userdata fields)
“CLASS”.USERDATA.? - ALL Userdata fields in the named class*

This is done to avoid the need for an administrator with
wide scope having to be put on the access list for all
individual fields.

Example: Read access to ‘dataset.base.?’ allows a user
to read any field in the base segment of any
dataset profile, even if a more specific
“authprof” exists, e.g. ‘dataset.base.owner’

&RACUID - placing &RACUID on the access list for an
authorisation profile in the field class is
supported, even if the class is not raclisted. *
| this is checked only when a user performs an *
| operation on their own user profile. *
| it can only be used to give users access to fields*
| (userdata or standard) in their own user profile, *
| either for read or update. *
Note: &RACUID does not work on generic auth profiles *
for userdata fields, e.g. ‘user.userdata.*’ (this *
is because ‘userdata’ is not recognised as a valid *
segment name in normal RACF processing and thus *
ot supported by ‘field level access’ checking as *
specified by ‘FLDACC=Yes’ on ICHEINTY) *

********************************************************************
SPECIAL FIELDS

USERDATA:

THE USERDATA FIELD IN A RACF TEMPLATE IS A REPEAT-GROUP WHERE
* EACH OCCURRENCE WITHIN THE REPEAT-GROUP IS MADE UP OF 3 FIELDS:
* USRNBM : 8 CHARACTERS - USED AS THE NAME OF THE ENTRY
* USRDATX : 1-255 CHARACTERS - CONTAINS THE DATA
* USRFGLS : 1 CHAR - CAN BE USED AS A FLAG

* THIS PROGRAM CAN BE USED TO MAINTAIN THESE FIELDS WHILE
* PROVIDING SELECTIVE CONTROL OVER WHO CAN READ/UPDATE WHICH
* INDIVIDUAL ENTRIES.

* ‘HIDDEN’ FIELDS:

* THESE ARE SUPPORTED TO ALLOW DATA TO BE STORED IN A USERDATA
* ENTRY THAT IS NOT TO BE DISPLAYED.
* THE FIELD NAME SHOULD START WITH AN ‘@’ SIGN.
* IN THIS CASE THIS PROGRAM WILL DISPLAY EACH CHARACTER OF THE
* FIELD AS A ‘?’ TO INDICATE THE LENGTH AND PRESENCE OF THE FIELD
* WHILE NOT DISCLOSING THE CONTENTS.

* EXAMPLE: INTY ADD PR(FRED) CL(USER) FI(@PW) DA(1234567)

* ‘REPEAT GROUPS’

* TO SELECT A SPECIFIC OCCURRENCE WITHIN A REPEAT-GROUP (E.G. TO
* RETRIEVE CONNECT INFORMATION FROM A USER PROFILE FOR A SPECIFIC
* RACF GROUP) THEN SPECIFY IN THE DATA() PARAMETER A VALUE TO BE
* COMPARED TO THE FIRST NAMED FIELD IN FIELD(). WHEN AN OCCURRENCE
* IS FOUND WITH THIS FIELD MATCHING THEN VALUES FOR THE SAME
* RELATIVE OCCURRENCE WILL BE RETRIEVED FOR ALL OTHER FIELDS NAMED
* IN FIELD(). THIS IS BEST EXPLAINED BY EXAMPLE!....

* GET PR(FRED) FI(CGGRPNM,CGAUTH,CGAUTHDA.P) DATA(SYSI) @MC7

* EACH OCCURRENCE OF THE ‘CGGRPNM’ REPEAT GROUP IS SCANNED FOR A
* MATCH WITH ‘SYS1’. WHEN ONE IS FOUND THE REMAINING REPEAT-GROUP
* FIELDS (CGAUTHOR AND CGAUTHDA) ARE SCANNED AND THE VALUES FROM
* THE SAME RELATIVE OCCURRENCE ARE RETRIEVED.
* THIS EFFECTIVELY RETRIEVES THE GROUP NAME, CONNECT OWNER AND
* CONNECT DATE (CONVERTED FROM PACKED TO CHAR) FOR THE CONNECT
* ENTRY ‘SYS1’ IN FRED’S USER PROFILE (DEFAULT CLASS=USER).
*

***************************************************************************
*                       OUTPUT                                           *
***************************************************************************
*  
*  CLIST VARIABLES:  
*  
*     'ORDINARY FIELDS'  
*     FOR EACH ORDINARY FIELD RETRIEVED A CLIST VARIABLE IS  
*     CREATED WITH THE SAME NAME AS THE FIELD.  
*     THE USRFLG ASSOCIATED WITH A USRDATA FIELD IS WRITTEN TO  
*     A VARIABLE WITH THE SAME NAME AS THE USRDATA, PLUS A  
*     SUFFIX OF 'F'.  
*     IF DUPLICATE FIELDS EXIST (E.G. IN USERDATA) THEN THE  
*     LAST FIELD FOUND IS THE ONE PUT INTO THE VARIABLE; BUT  
*     ALL FIELDS FOUND ARE LISTED AT THE TERMINAL AND CAN BE  
*     "SYSOUTTRAPPED" IF REQUIRED.  
*     
*     'REPEAT-GROUP FIELDS'  
*     WHERE A FIELD IS A MEMBER OF A REPEAT-GROUP (I.E. DEFINED  
*     AS SUCH IN THE IBM RACF TEMPLATES) A NUMERIC SUFFIX IS  
*     APPENDED TO THE VARIABLE NAME TO UNIQUELY IDENTIFY EACH  
*     OCCURRENCE OF THE REPEAT-GROUP, AND THE NUMBER OF SUCH  
*     VARIABLES CREATED IS PUT IN A VARIABLE WITH THE SAME NAME  
*     AS THE FIELD SPECIFIED.  
*     
*     'SPECIFIC OCCURRENCES '  
*     WHERE SPECIFIC OCCURRENCES OF A REPEAT-GROUP HAVE BEEN  
*     REQUESTED (USING DATA() PARAMETER) THEN NUMBERED VARIABLES*  
*     ARE STILL CREATED, EVEN IF ONLY ONE IS MATCHED.  
*     (SEE EXAMPLE UNDER 'REPEAT GROUPS' IN 'SPECIAL FIELDS'  
*     SECTION ABOVE).  
*     
*     EXAMPLES  1.IF THE COMMAND LINE INCLUDED 'FIELDS(NAME)' THEN  
*     THE 'NAME' FIELD OF A USER PROFILE IS PUT INTO A  
*     CLIST VARIABLE CALLED 'NAME'.  
*     
*     2.IF A USER IS CONNECTED TO 10 GROUPS AND THE COMMAND*  
*     LINE INCLUDED: 'FIELDS(CGGRPNM)'  
*     THE FOLLOWING VARIABLES WOULD BE CREATED:  
*     'CGGRPNM'  = 10 (THE NO. OF CGGRPNM OCCURRENCES )  
*     'CGGRPNM1'  = THE NAME OF THE 1ST CONNECT GROUP  
*     'CGGRPNM2'  = THE NAME OF THE 2ND CONNECT GROUP  
*     |  
*     'CGGRPNM10'  = THE NAME OF THE 10TH CONNECT GROUP  
*     
**************************************************************************  
*  RETURN CODES  
*  
*  0 - REQUEST COMPLETED OK  
*  4 - PROFILE NOT FOUND  
*  
**************************************************************************
* USERDATA FIELD NOT FOUND
* 8 - INSUFFICIENT AUTHORITY FOR REQUEST
* 12 - COMMAND PARSE FAILED
* FIELD NAME INVALID
* SEGMENT NAME INVALID
* SYNTAX ERROR IN COMMAND PARAMETERS
* CLASS INVALID/INACTIVE
* 16 - ICHENTRY WORKAREA TOO SMALL (RACWA)
* OTHER ICHENTRY ERROR (MSG CONTAINS ICHENTRY REASON)
* ALL ERRORS ARE ACCOMPANIED BY A MESSAGE (WHICH CAN BE
* SUPPRESSED BY THE ‘NOMSG’ OPTION)

******************************************************************
* DEPENDENCIES

******************************************************************
* AMODE=31, RMODE=ANY, AC=1
* NON-REENTRANT
* TSO COMMAND PROCESSOR
* AUTHORISED IN IKJTSOXX

******************************************************************
* CHANGE HISTORY

******************************************************************
* WHO WHEN DESCRIPTION ID?
* _ _ ____________________________ __
* MJC 160994 1.CORRECT OUTPUT WHEN RETRIEving REPEAT GRP WITH NO MC1
* OCCURRENCES (E.G. RACF GROUP WITH NO USERS). MC1
* 2.INCREASE RACF WORK AREA AND PUTLINE BUFFER SIZES. MC1
* MJC 201894 1.SET RC4 IF USERDATA FIELD NOT FOUND FOR DELETE. MC2
* 2.ADD ‘NOMSG’ OPTION TO SUPPRESS ALL MESSAGES. MC2
* 3.ADD ‘GENERIC’ OPTION TO FORCE GENERIC SEARCH. MC2
* 4.SUPPRESS LEADING ZEROS ON BINARY FIELD DISPLAY. MC2
* 5_ALLOW MULTIPLE SPECIFIC REPEAT-GROUP OCCURRENCE MC3
* RETRIEVAL INTO CLIST VARIABLES MC3
* MJC 121294 1.ADD AUTHORISATION CHECK TO ALLOW ACCESS TO PROFILE MC4
* IF CALLER HAS GROUP-AUTHORITY IN OWNING GROUP TREE MC4
* 2.SUPPORT &RACUID ON ACCESS LISTS OF ‘FIELD’ CLASS MC4
* PROFILES. MC4
* MJC 070195 1.CORRECTIONS TO ABOVE GROUP-AUTH CHECK. MC5
* 2.ADD ‘TRACE’ OPTION. MC5
* MJC 03101 1.ENHANCE TRACE INFORMATION MC6
* 2.ADD ERROR CHECKS AND ESTAE MC6
* 3.ADD 3RD BASE REG MC6
* 4.ADDED AUTOMATIC ‘REPEAT-GROUP’ RECOGNITION MC7
* 5.CREATE VARIABLE CONTAINING USRFLG WITH ‘F’ SUFFIX MC7

******************************************************************
* MACRO

@TRACE
LCLA &NDX,&L,&POS @MC5
LCLC &LIT @MC5
&POS SETA 8 LEAVE ROOM IN MSG FOR MSGID @MC5
.NXT ANOP @MC5
&NDX SETA &NDX+1 @MC5
AIF (&NDX GT N'&SYSLIST).END @MC5
&LIT SETC '=' @MC5
&PRM SETC '&SYSLIST(&NDX,1)' @MC5
&L SETA K'&PRM-2 @MC5
AIF ('&PRM'(1,1) EQ ' ').L02 @MC5
&L SETA &SYSLIST(&NDX,2) @MC5
&LIT SETC ' ' @MC5
.L02 ANOP @MC5
MVC TRTEXT+&POS.(&L),&LIT&PRM @MC5
&POS SETA &POS+&L @MC5
AGO .NXT @MC5
.END ANOP @MC5
MVI TRLEN,&POS LENGTH OF TRACE MSG @MC5
BAL R14,STRACE CALL TRACE ROUTINE @MC5
TR&SYSNDX DS 0H @MC5
MEND

* PRINT NOGEN
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8 COMMAND PARAMETERS
R9 EQU 9 3RD BASE
R10 EQU 10 SUBROUTINE LINKAGE
R11 EQU 11 2ND BASE
R12 EQU 12 1ST BASE
R13 EQU 13 SAVEAREA/WORKAREA POINTER
R14 EQU 14
R15 EQU 15
INTY AMODE 31
RMODE ANY
INTY CSECT
USING INTY,R15
SAVE (14,12),,'MCINTY V2.0' @MC6
LR R12,R15 LOAD BASE ADDR
DROP R15
USING INTY,R12,R11,R9
LA R11,4095(R12) LA R11,1(R11) LOAD 2ND BASE
LA R9,4095(R11)
LA R9,1(R9)       LOAD 3RD BASE       @MC6
LR R8,R1         SAVE PARM LIST ADDR   @MC1
L R0,=A(WORKLEN) GETMAIN R,OV=(Ø)
ST R1,8(R13)     OLD TO NEW
ST R13,4(R1)     NEW TO OLD
LR R13,R1        POINT TO OUR SAVEAREA
USING WORKAREA,R13 ADDR SAVE/WORK AREAS
USING CPPL,R8     ADDR PARAMETERS

*****************************************************************************************
*                                           *
* GET INFO ABOUT CURRENT USER               *
*                                           *
*****************************************************************************************
L R1,'X'224'       TO ASCB
USING ASCB,R1
L R1,ASCBASXB     TO ASXB
USING ASXB,R1
L R4,ASXBENV
USING ACEE,R4
ST R4,TSUACEE     SAVE ADDR USERS ACEE
MVC TSUSER,ACEEUSRI AND USERID
MVC TSUSERL,ACEEUSRL AND USERID LENGTH
MVC FLG1RAC,ACEEFLG1 AND USERS ATTRIBUTES
DROP R4

*****************************************************************************************
*                                           *
* PROCESS INPUT PARAMETERS                  *
*                                           *
*****************************************************************************************

*****************************************************************************************
*                                           *
* BUILD PARSE PARAMETER LIST                *
*                                           *

LA R2,LOCPPL      TO OUR LOCAL PPL
USING PPL,R2
L R1,CPPLUPT     TO UPT FROM PARMS
ST R1,PPLUPT
L R1,CPPLECT     TO ECT FROM PARMS
ST R1,PPLECT
XC LOCECB,LOCECB  CLEAR ECB FOR PARSER
LA R1,LOCECB     TO PARSE ECB
ST R1,PPLECB
L R1,=V(PCLPDL)  PARSE DESCRIPTOR LIST
ST R1,PPLPCL
LA R1,LOCANS     TO REPLY AREA FOR PARSER
ST R1,PPLANS
L R1,CPPLCPBUF   TO COMMAND BUFFER
ST R1,PPLCPBUF
XC PPLUWA,PPLUWA NO USER WORK AREA
DROP R2          DROP PPL        @MC6
* CALL TSO PARSER
* CALLTSSR EP=IKJPARS,MF=(E,(R2))
ST R15,SAVER15
LTR R15,R15
BNZ ERR1
DROP R8 DROP CPPL
* PROCESS RESULTS FROM PARSER *
PARSOK DS 0H
L R8,LOCANS TO Parsed Command
USING IKJPARMR,R8
OC KRG,KRG WAS ‘RGROUP’/’NORGROUP’ CODED? @MC7
BZ *+8 NO @MC7
OI KRG,X’80’ SET ‘USER OVERRIDE’ FLAG @MC7
* SET UP RECOVERY ENVIRONMENT @MC6
STM R8,R15,RECREGS SAVE REGS FOR ESTAE EXIT RETRY @MC6
OC KDEBUG,KDEBUG DEBUG MODE SPECIFIED? @MC6
BNZ ESTAEOK YES, OMIT ESTAE @MC6
LR R2,R13 POINT TO AREA TO PASS TO ESTAE @MC6
ESTAE ESTAEX,PARAM=(2) @MC6
LTR R15,R15 ESTAE SETUP OK? @MC6
BJ ESTAEOK YES @MC6
CVD R15,DWD1 RETURN CODE FROM ESTAE @MC6
OI DWD1+7,X’8F’ @MC6
UNPK EMG9,EMG9,DWD1+6(2) @MC6
LA R1,EMG9 @MC6
LA R0,L’EMG9 @MC6
BAL R14,SPUTMSG ESTAE FAILED, BUT CONTINUE @MC6
ESTAEOK DS 0H @MC6
* GET FUNCTION CODE (1ST OPERAND OF COMMAND) *
L R15,=V(PCLPDL)
USING PCLPDL,R15
LH R1,KFUNC+4 OFFSET TO FUNC PDE IN PDL
LA R1,IKJPARD(R1) ADDR FUNC PDE
LH R1,4(R1) GET ‘RESERVED WORD NO.’
STC R1,FCNCCODE SAVE IT
DROP R15
* GET CLASS NAME FROM PARMS, OR USE DEFAULT *
MVC RCLASSL,DFLTCLS USE DEFAULT CLASS LEN
MVC RCLASS,DFLTCLS+1 AND DEFAULT CLASS NAME
SLR R2,R2
ICM  R2,3,CLASS+4    LEN CLASSNAME
BZ  CLSØ1    USE DEFAULT
ICM  R1,15,CLASS    TO CLASS FROM PARMs
STC  R2,RCLASSL    STORE LEN CLASSNAME
BCTR  R2,0
MVC  RCLASS,='CL8' '
MVC  RCLASS(*-*),Ø(R1)
EX  R2,*-6    USE CLASS NAME FROM PARMs
CLSØ1  DS  ØH
*
* GET PROFILE NAME FROM PARMs
*
SLR  R2,R2
ICM  R2,3,PROF+4    LEN PROFILE FROM COMMAND LINE
BZ  ERR4    NONE, ERROR
MVC  PROFILENAME,BLANKS
STC  R2,PROFILENAME
ICM  R1,15,PROF    PROFILE ADDRESS FROM PARMs
BCTR  R2,0
MVC  PROFILENAME+1(*-*),Ø(R1)
EX  R2,*-6    MOVE TO OUR AREA
*
* GET SEGMENT NAME FROM PARMs, OR USE DEFAULT
*
MVC  RSEG,DLTSEG+1    USE DEFAULT SEGMENT NAME
SLR  R2,R2
ICM  R2,3,SEGNAME+4    LEN SEGMENT FROM PARMs
BZ  SEGØ1    NONE, USE DEFAULT
ICM  R1,15,SEGNAME    TO SEGMENT NAME FROM PARMs
BCTR  R2,0
MVC  RSEG,='CL8' '
MVC  RSEG(*-*),Ø(R1)
EX  R2,*-6    USE SEGMENT NAME FROM PARMs
SEGØ1  DS  ØH
CLI  FUNCODE,FUNCGET    CHECK FUNCTION
BE  SEGØ2    CAN RETRIEVE FROM ANY SEGMENT
CLC  RSEG,='CL8' 'BASE'    UPDATE FUNCTIONS CAN ONLY PROCESS
BNE  ERR8    USERDATA IN BASE SEGMENT
SEGØ2  DS  ØH
*
* SET UP BASIC ICHIENITY ACCORDING TO CLASS AND SEGMENT NAME
*
XC  RACWA,RACWA    CLEAR WORK AREA
L  R0,=A(RACWAL)    GET LENGTH   @MC1
ST  R0,RACWA    AND STORE IN WORK AREA
MVI  INTF+3,Ø    RESER ACTION COUNT FOR 'LOCATE' FLDEF
ICHEINTY ENTRY=PROFILENAME,CLASS=RCLASS,OPTIONS=(NPRO,NOEXEC), +
RELEASE=1.9,MF=(E,INTY1) POINT TO PROFILE AND CLASS @MC4
ICHEINTY LOCATE,TYPE='USR',RELEASE=1.9,OPTIONS=(NPRO,NOEXEC), +
SEGMENT=RSEG,    (ONLY NEED PUT SEGMENT IN ONCE) +
MF=(E,INTY1)
OC KGENERIC,KGENERIC WAS GENERIC FORCED ?  @MC2
B2 SEG03 NO. @MC2
ICHEINTY LOCATE,RELEASE=1.9,OPTIONS=(NOPRO,NOEXEC), +
   GENERIC=UNCOND, FORCE GENERIC SEARCH +
   MF=(E,INTY1)
SEG03 DS 0H @MC2
CLC RCLASS,=CLB'USER' IF CLASS=USER THEN ITS READY
BE CLSOK
ICHEINTY LOCATE,TYPE='GRP',RELEASE=1.9,OPTIONS=(NOPRO,NOEXEC), +
   MF=(E,INTY1) SET CLASS=GROUP
CLC RCLASS,=CLB'GROUP' IF CLASS=GROUP THEN ITS READY
BE CLSOK
ICHEINTY LOCATE,TYPE='DS',RELEASE=1.9,OPTIONS=(NOPRO,NOEXEC), +
   MF=(E,INTY1)
CLC RCLASS,=CLB'DATASET'
BE CLSOK
* CHECK FOR VALID GENERAL RESOURCE CLASS
RACROUTE REQUEST=STAT,CLASS=RCLASS,RELEASE=1.9, +
   WORKA=RACWA,MF=(E,RACSTATL)
LTR R15,R15
BNZ ERR2
ICHEINTY LOCATE,TYPE='GEN',RELEASE=1.9,OPTIONS=(NOPRO,NOEXEC), +
   CLASS=RCLASS,MF=(E,INTY1)
CLSOK DS 0H
MVI AUTHCODE,255 CALLER NOT ALLOWED YET
* @MC4
* CHECK IF CALLER HAS SUFFICIENT GROUP AUTHORITY OVER GROUP @MC4
* STRUCTURE THAT OWNS THE PROFILE. @MC4
* IF THEY DO, ALLOW THE REQUEST, OTHERWISE CONTINUE WITH @MC4
* FURTHER CHECKS AT FIELD CLASS-SEGMENT-FIELD LEVEL. @MC4
* @MC4
BAL R10,SCHOWN CHECK AUTHORITY TO PROFILE @MC4
STC R15,AUTHCODE SAVE RESULTS OF AUTH CHECK @MC4
LTR R15,R15 AUTHORISED ? @MC4
BZ AUTHOK YES, NO NEED FOR FURTHER CHECKS @MC4
*
* BUILD RESOURCE NAME FOR AUTHORISATION CHECK AND MAKE INITIAL CHECK
* THIS IS IN THE FORMAT 'CLASS.SEGMENT.FIELD'
* HERE WE CHECK THE 'CLASS.SEGMENT' PART, FIELD IS CHECKED LATER
* DURING FIELD PROCESSING IF REQUIRED.
*XC AUTHENTL,AUTHENTL RESET ENTITY LENGTH FIELD
MVC AUTHENT,BLANKS AND FIELD ITSELF (MUST BE BLANKS)
LA R0,L'AUTHENT LEN OF ENTITY NAME BUFFER
STCM R0,3,AUTHENTL STORE IN BYTES 0-1 OF ENTITY LEN
LA R3,AUTHENT TO START OF BUFFER
*
* USE SECURITY PREFIX IF ANY CODED
*
SLR  R2,R2
ICM  R2,1,AUTHPREF   DO WE USE A PREFIX
B2  AUTHC01   NO
MVC  Ø(8,R3),AUTHPREF+1  MOVE IN PREFIX (MUST INCLUDE DOT)
LA  R3,Ø(R2,R3)  PAST PREFIX
*
*   CHECK AUTHORITY TO ALL FIELDS IN CLASS
*  
AUTHC01 IC  R2,RCLASSL
MVC  Ø(8,R3),RCLASS   PUT CLASS NAME IN
LA  R3,Ø(R2,R3)   PAST NAME SO FAR
MVC  Ø(2,R3),=C'.'?  ADD SPECIAL PART
BAL  R10,SAUTHCHK   CHECK ACCESS TO 'CLASS.?'
STC  R15,AUTHCODE   SAVE RESULTS OF AUTH CHECK
*
*   CHECK AUTHORITY TO ALL FIELDS IN CLASS.SEGMENT
*  
LA  R1,AUTHUPRF+1   USE SPECIAL SEGMENT NAME TO MEAN
IC  R2,AUTHUPRF   ACCESS TO USERDATA
OC  KUSRDATA,KUSRDATA  WAS 'USERDATA' SPECIFIED ?
BNZ  AUTHC05  YES, SET RESOURCE NAME FOR USERDATA
CLI  FUNCODE,FUNCGET  IS IT AN UPDATE OPERATION
BNE  AUTHC05  YES, ALWAYS USERDATA
ICM  R1,15,SEGNAME  TO SEGMENT NAME FROM PARMS
ICM  R2,3,SEGNAME+4  GET LEN SEGMENT NAME
BNZ  AUTHC05  PRESENT, USE IT
LA  R1,DFLTSEG+1  USE DEFAULT SEGMENT NAME
IC  R2,DFLTSEG  GET LEN SEGMENT NAME
AUTHC05 BCTR  R2,Ø
MVI  Ø(R3),C'.'   DOT SEPARATOR
MVC  1(*-* R3),Ø(R1)
EX  R2,*-6
LA  R3,2(R2,R3)   PAST DOT AND SEGMENT NAME
MVC  Ø(2,R3),=C'.'?  ADD SPECIAL PART
BAL  R10,SAUTHCHK   CHECK ACCESS TO 'CLASS.SEGMENT.?'
STC  R15,AUTHCODE
*
AUTHC10 DS  ØH
LA  R0,AUTHENT  TO START OF ENTITY NAME
SR  R3,R0  LENGTH OF ENTITY NAME (LESS .?)
STH  R3,AUTHL  SAVE FOR LATER
ICM  R1,15,FIELDS  FIELD NAMES SPECIFIED IN COMMAND ?
BNZ  AUTHOK  YES, AUTH CHECK LATER FOR EACH FIELD
CLI  AUTHCODE,Ø   IS CALLER AUTHORISED TO CLASS/SEGMENT
BNE  ERR7   NO
AUTHOK DS  ØH
*
*   SET UP DATA FOR USERDATA WRITE
*  
SLR  R2,R2
XC  UDDATA,UDDATA

XC UDL LEN, UDL EN CLEAR TOTAL LEN OCCURRENCE @MC4
ICM R2,3, DATA+4 LEN DATA FROM COMMAND LINE
BZ NODAT NONE
STCM R2,15, UDDATA STORE LEN IN USERDATA
ICM R1,15, DATA DATA ADDRESS FROM PARMS
BCTR R2,0
MVC UDDATA(*-*), Ø(R1)
EX R2,*-6 MOVE TO OUR AREA
LA R1, UDDATA+1(R2) PAST UDDATA
MVC Ø(5,R1), =X’00000000100’ LEN_FLG+FLAG
LA R2, UDL+1(R2) LEN UDNAME+UDDATA+UDDATA+UDFLG ETC.
ST R2, UDL LEN TOTAL LEN OF OCCURRENCE

* GET FLAG VALUE IF ANY AND PUT INTO UDFLG (R1 -> UDFLG-4)

ICM R2,15, FLAG TO FLAG VALUE FROM PARMS
BZ *+8 NONE, LEAVE AS ZERO
L R2, Ø(R2) GET BINARY FLAG VALUE
C R2, =F’255’ CHECK
BP ERR4A OUT OF RANGE
STC R2,4(R1) PUT FLAG VALUE IN UDFLG

NODAT DS ØH

***************************************************************************
* BRANCH ACCORDING TO FUNCTION REQUESTED
***************************************************************************

CLI FUNCODE,FUNCGET ‘GET’
BE GET YES

***************************************************************************
* UPDATE FUNCTIONS ONLY
***************************************************************************
* BUILD ICHEINT REQUEST TO UPDATE USERDATA
***************************************************************************

***************************************************************************
* VERIFY FIELD NAME AND SET UP FOR ICHEINT
***************************************************************************

SLR R2,R2
ICM R2,3, FIELDS+4 LEN FIELD NAME
BNZ UPD10 PRESENT, SET IT UP AND CHECK
CLI FUNCODE, FUNCDEL IS IT A ‘DEL’ REQUEST
BE DELALL YES, DELETE ALL USERDATA

B ERR3 MUST SPECIFY FIELD NAME IF NOT DEL
UPD10 CLC =X’FF000000’, FIELDS+8 IS THERE ONLY ONE FIELD NAME
BNE ERR3A NO, ONLY ALLOWED 1 PER UPDATE

MVC UDNAME, BLANKS
ICM R1,15,FIELDS FIELD NAME ADDRESS FROM PARMS
BCTR R2,Ø
MVC UDNAME(*-*) ,Ø(R1)
EX R2, R2,*-6 SET FIELD NAME IN USRM
*
* DO AUTH CHECK FOR FIELD
*
LA R3, AUTHENT TO START OF AUTH ENTITY NAME
AH R3, AUTHL +LEN OF CLASS.SEG PART OF NAME
MVI Ø(R3), C'.' DOT SEPARATOR
MVC 1(8, R3), BLANKS ENSURE LAST FIELD NAME CLEARED
MVC 1(*-*, R3), Ø(R1) ADD FIELD NAME TO ENTITY NAME
EX R2, R2,*-6
AH R2, AUTHL NEW LEN INCLUDING FIELD NAME
LA R2, 2(R2) (+1 FOR EARLIER BCTR +1 FOR DOT)
STH R2, AUTHENTL+2 @MC4
BAL R1 Ø, SAUITHCHK CHECK ACCESS TO ‘CLASS.SEGMENT.FIELD’
LTR R15, R15
BNZ ERR7 NOT AUTHORISED TO FIELD
*
CLI FUNCODE, FUCNADD IS IT AN ‘ADD’ REQUEST
BE UPDADD YES, NO DELETE REQUIRED
******************************************************************************
*
* ‘DELETE’ A SPECIFIC OCCURRENCE
* (‘REP’ ALSO DOES DELETE FOLLOWED BY ADD)
*
******************************************************************************
DELSPEC DS ØH
*
* RETRIEVE ALL USERDATA AND DELETE REQUIRED OCCURRENCES FROM WORK
* AREA THEN REWRITE ENTIRE USERDATA.
* THIS IS DONE TO ENABLE MULTIPLE OCCURRENCES WITH THE SAME USRM TO
* BE STORED, ‘ICHEINTY DELETE’ DOES NOT WORK PROPERLY IN THIS CASE.
*
ICHEINTY LOCATE, ACTIONS=(ACTN2, ACTN2A), RELEASE=1.9, +
  OPTIONS=(ACTION), WKAREA=RACWA, +
  MF=E, INTY1
MVI FLG2GETU, FLG2DEL INDICATE DELETE TO SGETUDAT ROUTINE
BAL R1 Ø, SGETUDAT DELETE OCCURRENCES FROM WORK AREA
*
ST R3, AOCC SAVE ADDR LOCATED OCCURRENCE
STC R15, DELRC SAVE RC FROM DELETE
SLR R15, R15 RESET SO ‘CHKINTY’ DROPS THRO
CLI DELRC, Ø ANY FIELD FOUND AND DELETED?
BNE DELCHK NO, NOTIFY USER
ICM R1, 15, USRCNT ANY OCCURRNCES LEFT
BZ DELALL NO, DELETE THE LOT
*
* INSERT NEW USERDATA LENGTH INTO ACTN.
* REWRITE ALL USERDATA.
* L R2,USRDLLEN GET LEN OF ALL USERDATA
ICHEACTN FLDATA=((2),RELEASE=1.8.1, +
   MF=(E,ACTN2A) PUT LEN USERDATA LEFT INTO ACTN
ICHEINTY ALTER,ACTIONS=(ACTN2A),RELEASE=1.9, +
   OPTIONS=(ACTION), +
   MF=(E,INTY1)
B DELCHK
*******************************************************************************
*
* ‘DELETE’ ALL USERDATA
*
*******************************************************************************
DEALL DS 0H
ICHEINTY ALTER,ACTIONS=ACTN4,RELEASE=1.9, +
   OPTIONS=(ACTION,FLDEF), +
   MF=(E,INTY1) DELETE ALL USERDATA
DELCHK DS 0H
CLI FUNCODE,FUNCDEL IS IT A ‘DEL’ REQUEST
BE CHKINTY YES, ALL DONE, CHECK STATUS
*******************************************************************************
*
* ‘ADD’ (‘REP’ ALSO DOES ADD AFTER FIRST DELETING)
*
*******************************************************************************
UPDADD DS 0H
L R2,UDLEN LEN WHOLE OCCURRENCE
ICHEACTN FLDATA=((2),UDATA),RELEASE=1.8.1,MF=(E,ACTN3)
ICHEINTY ALTER,ACTIONS=(ACTN3),RELEASE=1.9, +
   ENTRY=PROFILENAME, +
   OPTIONS=(ACTION,FLDEF), +
   MF=(E,INTY1)
B CHKINTY
*******************************************************************************
*
* BUILD ICHEINTY REQUEST FOR ‘GET’
*
*******************************************************************************
*
* FOR NON-USERDATA FIELDS, RETRIEVE ONLY THOSE FIELDS NAMED
* THERE IS 1 ICHEINTY POINTING TO AN ICHEINTY-FLDEF WHICH IS A
* LIST OF POINTERS TO THE ICHEACTNS. WE BUILD 1 ICHEACTN FOR EACH
* FIELD REQUESTED AND POINT THE FLDEF LIST TO IT. THE COUNT OF
* ACTIONS IS SET IN THE FLDEF LIST WHEN WE HAVE BUILT THEM ALL.
* FOLLOWING EACH ACTN WE BUILD IS A 4 BYTE FIELD USED TO INDICATE
* ANY SPECIAL PROCESSING FOR THIS FIELD.
*
* FOR USERDATA RETRIEVAL WE JUST USE 2 FIXED ACTIONS TO READ ALL
* USERDATA AND THEN BREAK IT DOWN OURSELVES WITHIN THE BUFFER INTO
* INDIVIDUAL FIELDS. WE STILL BUILD THE FLDEF ACTION LIST JUST
* SO WE CAN USE THE SAME CODE LATER TO PROCESS THE FIELD-NAMES AND
* DATA RETURNED BY INTY.
*
GET  DS  0H
   XC FLDCOUNT,FLDCOUNT COUNT OF FIELDS REQUESTED
   LA  R5,FIELDS TO 1ST PDE FOR FIELD NAME LIST
   LA  R3,INTYF+4 START OF ACTION PTRS IN FLDEF LIST
   LA  R4,WACTNS AREA TO BUILD ICHEACTNS
FLDL1 DS 0H
   OC 0(4,R5),0(R5) TEST PTR TO FIELD NAME
   BZ  LOCI NO MORE
   MVC 0(LACTN,R4),ACTN1 ACTN BASE
   ICM  R1,15,0(R5) PTR TO FIELD NAME
   LH  R2,4(R5) GET LEN OF FIELD NAME
*
* CHECK FOR FORMAT SUFFIX IN FIELD NAME AND SET INDICATOR TO
* CONVERT DATA AFTER RETRIEVAL IF SUFFIX PRESENT.
*
   LA  R14,0(R2,R1) PAST END OF FIELD NAME
   BCTR R14,0 BACK TO LAST..
   BCTR R14,0 ..BUT ONE
   CLI 0(R14),C'.' IS SUFFIX PRESENT
   BNE FLDL10 NO
   BCTR R2,0 REDUCE LEN OF NAME..
   BCTR R2,0 ..BY SUFFIX LEN
   SLR R0,R0
   IC  R0,1(R14) GET DATA-TYPE CHAR. (SUFFIX)
   LA  R14,LACTN(R4) PAST ACTN TO OUR 4-BYTE FIELD IND.
   STCM R0,15,0(R14) SAVE DATA-TYPE CHAR. FOR LATER
FLDL10 DS 0H
   BCTR R2,0
   MVC 4(8,R4),=CL8' ' ENSURE BLANK FIELD NAME
   MVC 4(*-*,R4),0(R1) MOVE FIELD NAME INTO ACTN
*
* DO AUTH CHECK FOR FIELD
*
   LA  R14,AUTHENT TO START OF AUTH ENTITY NAME
   AH  R14,AUTHL +LEN OF CLASS.SEG PART OF NAME
   MVI 0(R14),C'.' DOT SEPARATOR
   MVC 1(8,R14),BLANKS ENSURE LAST FIELD NAME CLEARED
   MVC 1(*-*,R14),0(R1) ADD FIELD NAME TO ENTITY NAME
   EX  R2,*-6
   AH  R2,AUTHL NEW LEN INCLUDING FIELD NAME
   LA  R2,2(R2) (+1 FOR EARLIER BCTR +1 FOR DOT)
   STH R2,AUTHENTL+2 @MC4
   BAL R10,SAUTHCHK CHECK ACCESS TO 'CLASS.SEGMENT.FIELD'
   LTR R15,R15
   BZ  FLDOK AUTHORISED TO FIELD @MC4
   *
* IF CALLER STILL DOESN'T HAVE ACCESS TO THE FIELD, LET   @MC4
* ICHENITY USE STANDARD FIELD CHECKING AS THIS WILL ALSO  @MC4
* RECOGNISE &RACUID WHEN USED ON A GENERIC FIELD PROFILE @MC4
* DESCRIBING THE ‘USER’ CLASS, E.G. ‘USER.BASE.*’       @MC4
*                                                 @MC4
   ICHENITY FLDACC=YES,OPTIONS=(NOPRO,NOEXEC),      +
     RELEASE=1.9,MF=(E,INTY1) SWITCH ON ‘FIELD ACCESS’ @MC4
FLDOK DS @MC4
   ST R4,Ø(R3) PUT ACTION ADDR IN FLDEF LIST
   LH R1,FLDCOUNT
   LA R1,1(R1) INCR FIELD (ACTION) COUNT
   STH R1,FLDCOUNT
   LA R4,LACTN+4(R4) TO NEXT ACTION PTR IN FLDEF LIST
   ICM R5,15,8(R5) TO AREA FOR NEXT ACTION
   C R5,=X’FF000000’ ANY MORE
   BNE FLDL1 YES
   MVC INTYF+3(1),FLDCOUNT+1 SET ACTION COUNT IN FLDEF LIST
**************************************************************************
* RETRIEVE INFORMATION FROM RACF DATABASE
*
**************************************************************************
LOC1 DS @H
   OC KUSRDATA,KUSRDATA WAS ‘USERDATA’ SPECIFIED
   BZ LOC2 NO, USE DYNAMIC FLDEF LIST
* FOR USERDATA, RETRIEVE ALL USERDATA USING 2 FIXED ACTIONS
*
   ICHENITY LOCATE,ACTIONS=(ACTN2,ACTN2A),RELEASE=1.9,       +
     OPTIONS=(ACTION),WKAREA=RACWA,                          +
     MF=(E,INTY1)
B CHKINTY
*
* FOR NON-USERDATA, USE DYNAMIC FLDEF ACTION LIST
*
LOC2 DS @H
   OC FLDCOUNT,FLDCOUNT CHECK NO. OF FIELDS
   BZ ERR3
   ICHENITY LOCATE,RELEASE=1.9,                                 +
     OPTIONS=(ACTION),FLDEF=INTYF,WKAREA=RACWA,              +
     MF=(E,INTY1)
C R15,=F’88’ ‘FIELD-LEVEL-ACCESS’ SOME FAILED @MC5
BNE CHKINTY NO, CHECK ANY OTHER ERRORS @MC5
   LA R1,EMSG7A INDICATE NOT ALL FIELDS WERE @MC5
   LA R0,’EMSG7A RETURNED DUE TO ACCESS CHECK @MC5
   BAL R14,PUTMSG FAILURE. @MC5
   MVC RETCODE,=F’8’ INDICATE AUTH ERROR IN RETCODE @MC5
   SLR R15,R15 TREAT AS SUCCESSFUL @MC5

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
* * CHECK ICHEINTY RETURN CODE
* *
CHKINTY DS 0H
LTR R15,R15 CHECK RC FROM ICHEINTY
BNZ ERR5 FAILED, FIND OUT WHY
CLI FUNCODE,FUNCGET IS THIS A ‘GET’ REQUEST
BE GETINF YES, RETRIEVE INFO
*
* FOR UPDATE FUNCTIONS ONLY:
* NOTIFY USER OF UPDATE STATUS AND TERMINATE
*
UPDMSG DS 0H
SLR R1,R1
IC R1,FUNCODE
BCTR R1,0
MH R1,Y(L’MSG00P0) INDEX TO MSG0 ACTION TAKEN @MC2
LA R2,MSG00P0(R1)
CLI DELE,0 WERE DELETES OK (REP/DEL) ?
BE +10 YES, REPLACED/DELETED
LA R2,MSG00P1(R1) NO, ADDED, NOT DELETED
MVC MSG00DES,0(R2)
LA R0,L’MSG0
LA R1,MSG0
CLC =C’NOT FOUND’,0(R1) FAILED DELETE REQUEST @MC2
BE ERR6A FIELD NOT FOUND TO DELETE @MC7
* BNE +10 NO, LEAVE RC @MC7
* MVC RETCODE,F’4’ INDICATE FIELD NOT FOUND TO DELETE@MC7
BAL R14,SPUTMSG DISPLAY STATUS MESSAGE
B RETURN

MSG0 DC C’MC1001: USERDATA XXXXXXXXXX’
MSG0DES EQU *-10,10
MSG00P0 DC CL10’ ‘,CL10’REPLACED’,CL10’ADDED’,CL10’DELETED’
MSG00P1 DC CL10’ ‘,CL10’ADDED ‘,CL10’ADDED’,CL10’NOT FOUND’
*****************************************************************************
* * ‘GET’ FUNCTION
* *
* RETRIEVE RESULTS FROM WORK AREA AND PASS TO CALLER
* 
*****************************************************************************

GETINF DS 0H
MVC UDNAME,BLANKS SELECT ALL FIELDS BY DFLT
SLR R5,R5
ICM R5,1,INTF+3 GET ACTION (FIELD) COUNT
SLR R6,R6 INDEX TO ACTION POINTER
ZAP SOCCNO,=P’0’ SELECTED OCC. NO. IF DATA() SPECIFIED

GETFLDS DS 0H
MVI LINE,C’ ‘
MVC LINE+1(L’LINE-1),LINE CLEAR TO BLANKS

L R4,INTYF+4(R6) TO 1ST/_NEXT ICHEACTN
OC KUSRDATA,KUSRDATA WAS ‘USERDATA’ SPECIFIED
BZ GETFLD1 NO, PROCESS DATA FROM ICHEACTNS
******************************************************************************
* GET REQUESTED USERDATA FIELDS
******************************************************************************
LTR R5,R5 SPECIFIC FIELDS REQUESTED
BZ +10 NO, PROCESS ALL USERDATA
MVC UDNAME,4(R4) PASS FLDNM TO USERDATA ROUTINE
MVI FLG2GETU,0 DISPLAY OCCURRENCES
BAL RI0,SGETUDAT PROCESS USERDATA IF PRESENT
B *+4(R15)
B +12 FOUND IT, SEE IF ONE OR ALL
B ERR6A NAMED FIELD NOT FOUND
B ERR6 NO USERDATA IN PROFILE
LTR R5,R5 IF NO FIELD NAMES THEN ALL USERDATA
BZ ENDFLDS ALREADY LISTED BY SGETUDAT.
LA R6,4(R6) TO NEXT ACTION POINTER
BCT R5,GETFLD3 LIST NEXT USERDATA FIELD NAMED
B ENDFLDS ALL DONE
******************************************************************************
* NON-USERDATA FIELDS...
******************************************************************************
* GET DATA ASSOCIATED WITH THIS ICHEACTN
GETFLD1 DS 0H
MVC LINE(B),4(R4) GET FIELD NAME FROM ICHEACTN
MVC FLDNAME,4(R4) .. AND READY FOR CLIST VARIABLE
MVI FLDNAMEX,C’ ‘ RESET FIELD NAME SUFFIX @MC7
XC VALUELEN,VALUELEN RESET LEN OF CLIST VAR. DATA
ZAP OCCNO,=P’0’ CURRENT OCC. NO. FOR RPT GRPS @MC1
ZAP VARN0,=P’0’ VAR.NO. FOR RPT GRP FLDLS SELECTED@MC3
LA R0,B DFLT MSG LEN = LEN OF FIELD NAME
SLR R2,R2 @MC1
ICM R3,15,16(R4) GET ADDR DATA RETURNED
BZ FLDMSG NONE
ICM R2,15,12(R4) GET LENGTH DATA RETURNED
MVC FLDIND,LACTN(R4) GET DATA-CONV. CHAR IF ANY
CLI KRG,0 DID USER OVERRIDE RGROUP OPTION ? @MC7
BNE RGCHK2 YES, DON’T USE AUTO-RECOGNITION @MC7
* AUTOMATIC ‘REPEAT-GROUP’ RECOGNITION @MC7
* DETERMINE IF DATA RETURNED FOR THIS FIELD IS IN ‘REPEAT-GROUP’ @MC7
* FORMAT BY COMPARING THE ASSUMED LENGTH FIELDS WITH THE LENGTH OF @MC7
* DATA RETURNED AS INDICATED IN ICHEACTN+12. @MC7
MVI KRG+1,0 RESET RG INDICATOR @MC7
LA R1,0(R2,R3) PAST END OF RETURNED DATA @MC7
RGCHK1 SL R2,0(R3) SUB LEN OF POSSIBLE 1ST OCCURRENCE@MC7
BM RGCHK2 TOO LONG: NOT RG FORMAT @MC7
S R2,=F’4’ ALSO SUB LEN OF OCC. LEN FLD. @MC7
BM RGCHK2 GONE NEG: NOT RG FORMAT @MC7
A R3,0(R3) PAST DATA @MC7

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
LA R3,4(R3) ..AND PAST LEN TO NEXT OCC. @MC7
CR R3,R1 ARE WE AT END OF RETURNED DATA? @MC7
BM RGCHK1 NOT YET @MC7
BP RGCHK2 PAST IT: NOT RG FORMAT @MC7
MVI KRG+1,1 THIS FIELD IS A 'REPEAT-GROUP' @MC7
RGCHK2 ICM R3,15,16(R4) RESTORE ADDR DATA RETURNED @MC7
ICM R2,15,12(R4) RESTORE LENGTH DATA RETURNED @MC7
*
CLI KRG+1,1 IS FIELD A REPEAT-GROUP? @MC7
BNE GETF10 NO, SINGLE FIELD @MC7
*
* FOR REPEAT-GROUP FIELDS ONLY: @MC7
* BREAK DOWN DATA RETURNED TO FORMAT EACH OCCURRENCE.
* (NULL OCCURRENCES GO THROUGH ALL PROCESSES AS OCCURRENCE NUMBERS
* ARE STILL NEEDED FOR SELECTION AND COUNTING)
*
ICM R4,15,12(R4) TOTAL LEN OF ALL OCCURRENCES RETURNED
BZ FLDNXT NONE RETURNED @MC3
GETF05 DS $0H
AP OCCNO,="P'1' INCRR. OCCURRENCE NO.
ICM R2,15,0(R3) LEN OF THIS OCCURRENCE
LA R3,4(R3) PAST LENGTH FIELD TO DATA
*
GETF10 DS $0H
* CONVERT DATA IF FIELD NAME WAS SPECIFIED WITH CONVERSION SUFFIX.
* NOTE THAT THE POINTERS TO THE ORIGINAL (SOURCE) FIELD ARE RESTORED
* AFTER CONVERSION TO MAINTAIN OUR PLACE IN THE RACF BUFFER.
STM R2,R3,BUFTRS SAVE LEN AND ADDR OF SOURCE FIELD
CLI FLDIND+3,0 CONV. CHAR SPECIFIED?
BE ++8 NO
BAL R10,SCNVDAT CONVERT DATA
LTR R2,R2 NULL FIELD/OCCURRENCE @MC3
BZ GETF15 YES, DO NOT PUT IN OUTPUT LINE @MC3
CL R2,="F'256' DATA TOO LONG? @MC6
BP ERR10 YES, POSSIBLE 'RGROUP' SPEC WRONG @MC6
BCTR R2,0 LEN CONVERTED DATA
MVC LINE+9(*-),0(R3)
EX R2,*-6 MOVE CONVERTED DATA TO OUTPUT LINE
LA R2,1(R2) @MC3
GETF15 DS $0H @MC3
CLI KRG+1,1 IS FIELD A REPEAT-GROUP? @MC7
BNE OCCSEL NO, NO SELECTION ON DATA THEN @MC7
******************************************************************************
* FOR REPEAT GROUP OCCURRENCES, IF A SPECIFIC ONE WAS SELECTED BY
* THE DATA() PARAMETER, COMPARE THE CURRENT OCCURRENCE (1ST FIELD
* NAMED IN THE 'FIELDS' PARAMETER ONLY) FOR THE SPECIFIED VALUE.
* IF THIS MATCHES, SELECT THE SAME OCCURRENCE NUMBER(S) WHEN
* PROCESSING SUBSEQUENT FIELDS.
* UP TO 256 OCCURRENCES CAN BE SELECTED IN THIS WAY (THE MASK FIELD
* IS 256 BITS LONG).

OC  UDDATAL, UDDATAL  WAS DATA() PARAMETER SPECIFIED
BZ  OCCSELY  NO, RETURN ALL OCCURRENCES TO CALLER
LTR  R6, R6  FIRST FIELD IN FIELD() PARM?
BZ  OCCSEL1  YES, COMPARE VALUE WITH DATA() VALUE

*  @MC3
*  CHECK OCCURRENCE BIT MASK FOR 2ND AND SUBSEQUENT SELECTIONS  @MC3
*  @MC3
STM  R1, R3, SAVESUB  SAVE WORK REGS  @MC3
CVB  R1, OCCNO  CURRENT OCCURRENCE NO.  @MC3
LR  R2, R1  SAVE OCCNO  @MC3
SRL  R1, 3  DIV/8 FOR BYTE OFFSET IN MASK  @MC3
LR  R3, R1  SAVE BYTE OFFSET  @MC3
SLL  R1, 3  OCCNO ROUNDED DOWN TO 8  @MC3
SR  R2, R1  BIT OFFSET  @MC3
LA  R1, 'X'80'  BIT 1 OF 1 TO 8  @MC3
SRL  R1, 0(R2)  MOVE ACCORDING TO OFFSET  @MC3
LA  R3, SELMASK(R3)  INDEX INTO SELECTION MASK BYTE  @MC3
TM  Ø(R3), *-*  @MC3
EX  R1, *-*  TEST APPROPRIATE BIT IN MASK  @MC3
LM  R1, R3, SAVESUB  RESTORE WORK REGS  @MC3
BNZ  OCCSELY  SELECT THIS OCCURRENCE  @MC3
B  FLDNXT  NO, BYPASS THIS OCCURRENCE

*  @MC3
*  COMPARE OCCURRENCE DATA WITH THE SUPPLIED DATA BEFORE SELECTING IT
*  @MC3

OCCSEL1  DS  0H
LTR  R0, R2  SAVE LEN. CONVERTED DATA  @MC3
BNP  FLDNXT  NULL, CANNOT MATCH  @MC3
ICM  R2, 15, UDDATAL  GET LEN DATA FROM DATA() PARM  @MC3
CR  R2, R0  CHECK DATA LENGTH  @MC3
BP  FLDNXT  TOO LONG, CAN'T MATCH  @MC3
BCTR  R2, 0  @MC3
CLC  UDDATA(*-*), Ø(R3)  @MC3
EX  R2, *-*  COMPARE DATA() VALUE  @MC3
BNE  FLDNXT  DIFFERENT, BYPASS OCCURRENCE  @MC3
LR  R2, R0  RESTORE LEN CONVERTED DATA  @MC3

*  @MC3
*  INDICATE IN MASK THE OCCURRENCE NUMBER SELECTED, SO WE  @MC3
*  CAN OUTPUT THE SAME OCCURRENCE NUMBER FOR FOLLOWING FIELDS.  @MC3
*  @MC3

STM  R1, R3, SAVESUB  SAVE WORK REGS  @MC3
CVB  R1, OCCNO  CURRENT OCCURRENCE NO.  @MC3
LR  R2, R1  SAVE OCCNO  @MC3
SRL  R1, 3  DIV/8 FOR BYTE OFFSET IN MASK  @MC3
LR  R3, R1  SAVE BYTE OFFSET  @MC3
SLL  R1, 3  OCCNO ROUNDED DOWN TO 8  @MC3
SR  R2, R1  BIT OFFSET  @MC3
LA  R1, 'X'80'  BIT 1 OF 1 TO 8  @MC3
SRL  R1, Ø(R2)  MOVE ACCORDING TO OFFSET  @MC3
LA R3,SELMASK(R3) INDEX INTO SELECTION MASK BYTE @MC3
OI @(R3),*-* @MC3
EX R1,=*-4 SET APPROPRIATE BIT IN MASK @MC3
LM R1,R3,SAVESUB RESTORE WORK REGS @MC3

******************************
* WRITE OUT RETRIEVED DATA
******************************
OCCSELY DS @H FIELD SELECTED.
ST R2,VALUELEN PASS LEN OF CLIST VARIABLE DATA
ST R3,VALUEPTR POINT DATA FOR CLIST VARIABLE
LA R0,9(R2) ADD LEN FIELD DATA TO MSG LEN.
LM R2,R3,BUFPTRS RESTORE LEN AND ADDR OF SOURCE FIELD

FLDMSG DS @H
LA R1,LINE TO INFO TO WRITE OUT (R0=LEN)
CLI KLIST+1,2 WAS ‘NOLIST’ SPECIFIED’ @MC7
BE *+8 YES, DON’T DISPLAY DATA @MC7
BAL R14,SPUTMSG
BAL R10,SCVAR WRITE CLIST VARIABLE

FLDNXT DS @H
CLI KRG+1,1 IS FIELD A REPEAT-GROUP ? @MC7
BNE FLDNXTA NO, SINGLE FIELD @MC7
LM R2,R3,BUFPTRS RESTORE LEN AND ADDR OF SOURCE FIELD

* FOR REPEAT GROUP OCCURRENCES, PREPARE TO PROCESS NEXT OCCURRENCE.
* WHEN ALL ARE DONE, WRITE THE CLIST VARIABLE CONTAINING THE
* NUMBER OF OCCURRENCES (THE NUMBER OF CLIST VARIABLES CREATED).
* LA R3,0(R2,R3) PAST OCCURRENCE
LA R2,4(R2) INCL LEN-FLD IN LEN OF OCCURRENCE
SR R4,R2 DECR LEN LEFT TO PROCESS
BP GETFØ5 PROCESS NEXT OCCURRENCE

* ALL DONE, CREATE VARIABLE WITH COUNT IN.
LA R0,VARNUM POINT TO NUMBER OF OCCURRENCES
ST R0,VALUEPTR PASS ADDR TO CLIST VAR. ROUTINE
MVC VALUELEN,VARNUML .AND LENGTH OF IT
ZAP OCCNO,=P’0’ THIS FIELD IS NOT RPT GROUP DATA
BAL R10,SCVAR CREATE VARIABLE WITH COUNT IN IT
MVI VARNUM,C’0’ RESET COUNT VAR FOR NEXT FIELD @MC3
MVC VARNUML,=F’1’ @MC3

* THRO’ TO NEXT ICHEACTN (FIELD)
* PROCESS NEXT FIELD
*
FLDNXTA DS @H
LA R6,4(R6) TO NEXT ACTION POINTER
BCT R5,GETFLDS

ENDFLDS DS @H
*
**************************************************************************
* FREEMAIN STORAGE AND RETURN TO CALLER

* 

******************************************************************************
RETURN DS 0H
ESTAE Ø CANCEL OUR RECOVERY ROUTINE @MC6
RETURN1 DS 0H @MC6
L R10,RETCODE SAVE RC
LR R1,R13
L R13,4(R13)
*
DROP R13 WORKAREA LOST NOW
L R00=(WORKLEN) @MC1
FREEMAIN R,LV=(Ø),A=(1)
LR R15,R00 PASS BACK RC
RETURN (14,12),RC=(15)
*
******************************************************************************
* ERROR ROUTINES *
*
******************************************************************************
ERR1 DS 0H
CVD R15,DWD1 RETURN CODE FROM PARSE @MC7
OI DWD1+7,X’0F’ @MC7
UNPK EMSG1RC,DWD1+6(2) @MC7
LA R1,EMSG1
LA R0,L’EMSG1
MVC RETCODE,=F’12’
LA R14,RETURN
B SPUTMSG
EMSG1 DC C’MCIØ1E: PARSE FAILED RC=NNN’ @MC7
EMSG1RC EQU *,-3,3 @MC7
*
ERR2 DS 0H
LA R1,EMSG2
LA R0,L’EMSG2
MVC RETCODE,=F’12’
LA R14,RETURN
B SPUTMSG
EMSG2 DC C’MCIØ2E: CLASS INVALID OR INACTIVE’ *
ERR3A DS 0H
LA R1,EMSG3A
LA R0,L’EMSG3A
B ERR3Z
ERR3 DS 0H
LA R1,EMSG3
LA R0,L’EMSG3
ERR3Z MVC RETCODE,=F’12’
LA R14,RETURN
B SPUTMSG
EMSG3 DC C’MCIØ3E: NO FIELD NAMES SPECIFIED’
EMSG3A DC C'MCIØ3E: ONLY 1 FIELD ALLOWED PER UPDATE'
*
ERR4A DS ØH
    LA R1,EMSG4A
    LA RØ, L'EMSG4A
    B ERR4Z
ERR4 DS ØH
    LA R1,EMSG4
    LA RØ, L'EMSG4
ERR4Z MVC RETCODE, =F'12'
    LA R14, RETURN
    B SPUTMSG
EMSG4 DC C'MCIØ4E: PROFILE NAME MISSING'
EMSG4A DC C'MCIØ4E: FLAG OUT OF RANGE (Ø-255)'
*
* PROCESS ICHINTY ERRORS
*  
ERR5 DS ØH
    C R15, =F'92' 'FIELD-LEVEL-ACCESS' ALL FAILED?
    BE ERR7 YES, TREAT AS ACCESS FAILURE.
    C R15, =F'88' 'FIELD-LEVEL-ACCESS' SOME FAILED?
    BE ERR7 YES, TREAT AS ACCESS FAILURE.
    MVC RETCODE, =F'4' RC=4
    LR R2, RØ SAVE REASON CODE
    LA RØ, L'EMSG512
    LA R1, EMSG512
    C R15, =F'12' PROFILE NOT FOUND
    BE ERR5Z
    C R15, =F'36' ICHINTY FAILED
    BNE ERR5A
    MVC RETCODE, =F'12' RC=12
    LA R0, L'EMSG536A
    LA R1, EMSG536A
    C R2, =F'3' REASON=INVALID FIELD NAME
    BE ERR5Z YES
    LA R0, L'EMSG536B
    LA R1, EMSG536B
    C R2, =F'16' REASON=INVALID SEGMENT NAME
    BE ERR5Z YES
ERR5A MVC RETCODE, =F'16' RC=16
    LA R1, EMSG544
    LA R0, L'EMSG544
    C R15, =F'44' WORK AREA TOO SMALL
    BE ERR5Z
    CVD R15, DWD1 RETURN CODE
    OI DWD1+7, X'0F'
    UNPK EMSG5RC, DWD1+6(2)
    CVD R2, DWD1 REASON CODE
    OI DWD1+7, X'0F'
    UNPK EMSG5RS, DWD1+6(2)
LA   R1,EMSG5
LA   R0,EMSG5L
ERR5Z DS 0H
   LA   R14,RETURN
   B   SPUTMSG
*
EMSG512 DC 'MCIØ5E: PROFILE NOT FOUND'
EMSG536A DC 'MCIØ5E: FIELD NAME INVALID'
EMSG536B DC 'MCIØ5E: SEGMENT NAME INVALID'
EMSG544 DC 'MCIØ5E: WORK AREA TOO SMALL, TRY FEWER FIELDS'
*
EMSG5 DC 'MCIØ5E: ICHEINTY RC=NNN'
EMSG5RC EQU -*,3
   DC ' REASON=NNN'
EMSG5RS EQU -*,3
   EQU -*-EMSG5
*
ERR6A DS 0H
   LA   R1,EMSG6A
   LA   R0,L'EMSG6A
   B   ERR6Z
ERR6 DS 0H
   LA   R1,EMSG6
   LA   R0,L'EMSG6
ERR6Z MVC RETCODE,=F'4'
   LA   R14,RETURN
   B   SPUTMSG
EMSG6 DC 'MCIØ6E: NO USERDATA IN PROFILE'
EMSG6A DC 'MCIØ6E: USERDATA FIELD NOT FOUND'
*
ERR7 DS 0H
   MVC EMSG7RNM,AUTHENT INFORM WHAT NAME CHECKED     @MC4
   LA   R1,EMSG7
   LA   R0,L'EMSG7
   MVC RETCODE,=F'8'
   LA   R14,RETURN
   B   SPUTMSG
EMSG7 DC 'MCIØ7E: NOT AUTHORISED TO CCCCCCCC.SSSSSSSS.FFFFFFFF'
EMSG7RNM EQU -*,26,26      SPACE TO COPY 'AUTHENT' TO     @MC4
EMSG7A DC 'MCIØ7E: NOT ALL FIELDS RETURNED (ACCESS CHECK FAILED)'
*
ERR8 DS 0H
   LA   R1,EMSG8
   LA   R0,L'EMSG8
   MVC RETCODE,=F'12'
   LA   R14,RETURN
   B   SPUTMSG
EMSG8 DC 'MCIØ8E: USERDATA SUPPORTED IN BASE SEGMENT ONLY'
   @MC6
EMSG9 DC 'MCIØ9E: ESTAE SETUP FAILED RC=NNN'     @MC6
EMSG9RC EQU *-3,3  @MC6
*  @MC6
ERR10 DS 0H  @MC6
MVC EMSG10NM,FLDNAME INFORM WHICH FIELD FAILED  @MC6
LA R1,EMSG10  @MC6
LA R0, L'EMSG10  @MC6
MVC RETCODE, =F'12'  @MC6
LA R14, RETURN  @MC6
B SPUTMSG  @MC6
EMSG10 DC C'MCI10E: FIELD SPECIFICATION ERROR - XHHHHHHH'
EMSG10NM EQU *-8,8 NAME OF FIELD  @MC6
*

******************************************************************************
*
*  SUBROUTINE: WRITE A MESSAGE TO THE TERMINAL  *
*
*  ON ENTRY: R1 = ADDR MSG  *
*  R0 = LEN MSG  *
*  RETURNS VIA R14  *
*
*  WARNING!!! DO NOT USE @TRACE IN THIS SUBROUTINE AS A  @MC5
*  RECURSIVE LOOP WILL OCCUR.  @MC5
*
******************************************************************************

SPUTMSG DS 0H
LTR R0, R0 LEN DATA
BZR R14 NONE
CLI KMSG+1,2 WAS 'NOMSG' SPECIFIED  @MC7
BER R14 YES, NO MESSAGES AT ALL  @MC7
STM R14, R5, SAVESUB3 SAVE REGS ON ENTRY

SPUTTR DS 0H  @MC5
XC PUTHDR, PUTHDR CLEAR PUTLINE BUFFER HEADER
MVI PUTFBUF, C' '  @MC5
MVC PUTFBUF+1(L'PUTBUF-1), PUTFBUF
LR R2, R0
BCTR R2, 0
MVC PUTFBUF(*-4), R0(R1)
EX R2, -*6 MOVE TEXT TO OUTPUT BUFFER
LA R2, 5(R2) LEN BUFFER + HDR
STH R2, PUTHDR STORE LEN IN BUFFER
LA R5, LOCPPL TO OUR LOCAL PPL  @MC6
USING PPL, R5  @MC6
L R3, PPLUPT POINT TO UPT
L R4, PPLECT AND TO ECT
DROP R5 DROP PPL  @MC6
XC LOCECB, LOCECB CLEAR ECB
PUTLINE MF=(E, OURIOL), UPT=(3), ECT=(4), ECB=LOCECB, PARM=PUT11, +
OUTPUT=(PUTHDR, DATA)
LM R14, R5, SAVESUB3 RESTORE REGS

BR R14
EJECT

*************************************************************************************************
* @MC5
* SUBROUTINE: WRITE TRACE MESSAGE TO TERMINAL @MC5
* @MC5
* ON ENTRY: TRTEXT = MESSAGE TEXT @MC5
* TRLEN = MESSAGE LENGTH @MC5
* @MC5
* EXIT THROUGH SPUTMSG SUBROUTINE (VIA R14) @MC5
* @MC5
*************************************************************************************************

STRACE DS 0H @MC5
OC KTRACE,KTRACE WAS 'TRACE' SPECIFIED @MC5
BZR R14 NO, BYPASS TRACE @MC5
CLI TRLEN,0 LEN OF TRACE MESSAGE @MC5
BER R14 NONE @MC5
STM R14,R4,SAVESUB3 SAVE REGS @MC5
SLR R0,R0 @MC5
IC R0,TRLEN GET LEN OF TRACE MESSAGE @MC5
* MVC TRTEXT(0),=CLB"*TRACE*" MOVE IN TRACE MSGID @MC5
MVC TRTEXT(0),=CLB'MCI90I ' MOVE IN TRACE MSGID @MC6
LA R1,TRTEXT POINT TO MESSAGE @MC5
B SPUTTR WRITE MSG TO TERMINAL @MC5
* @MC5

*************************************************************************************************
* * SUBROUTINE: LOCATE USERDATA (ALL OR SPECIFIC FIELDS) *
* *
* ON ENTRY: UDNAME = BLANK- LIST ALL USERDATA *
* OR FIELDNAME- PROCESS FOR SPECIFIC ENTRY *
* *
* EXIT VIA R10 *
* *
* R1 = ADDR OF USERDATA OCCURRENCE IF MATCHED *
* AND REQUEST WAS TO LOCATE ONLY. (R15=0) *
* R15 = Ø FIELD PASSED BACK OR USERDATA LISTED/DELETED *
* R15 = 4 FIELD NOT FOUND *
* R15 = 8 NO USERDATA IN PROFILE *
* *
*************************************************************************************************

SGETUDAT DS 0H
SLR R0,R0 LENGTH OF DATA RETURNED
LA R15,8 NO USERDATA IN PROFILE
ST R10,SAVER10
STM R15,R8,SAVESUB
ZAP OCCNO,=P'0' NO NUMERIC SUFFIX FOR USERDATA VARS *

ICM R2,15,USRCNT GET NO. OF OCCURRENCES
BZ GETU99 NO USERDATA TO GET
LA R15,4 SPECIFIC USRNM NOT FOUND (YET)
ICM  R5,15,USRDLEN  TOTAL LEN OF ALL USRDATA
LA   R5,USRDLLEN+L'USRDLLEN(R5)  TO END OF USRDATA

*  LOOP THROUGH USRDATA OCCURRENCES, SELECT THOSE REQUIRED
*
LA   R3,USRDOCC  TO 1ST OCCURRENCE
USING USRDOCC,R3
GETU1  DS   0H
MVC  LINE,BLANKS  BLANK INFO LINE
ICM  R4,15,USRDTAL  LEN OF THIS USRDATA FIELD
*  SELECT PROCESSING
CLC  UDNAME,BLANKS  DO WE SELECT CERTAIN FIELDS ?
BE   GETU2  NO, SELECT ALL USRDATA
CLC  UDNAME,USRNM  IS THIS THE ONE
BNE  GETUNXT
GETU2  DS   0H
CLI  FUNCODE,FUNCREP  IS IT 'REPLACE'
BE   GETU4  YES, DON'T CHECK CURRENT VALUE
ICM  R2,15,UDDTAL  DO WE SELECT CERTAIN VALUES ?
BZ   GETU4  NO
CR   R2,R4  YES, CHECK DATA LENGTH
BNE  GETUNXT  WRONG LENGTH, CAN'T MATCH
BCTR  R2,0
CLC  UDDATA(*-*),USRDATA
EX  R2,.*-6  COMPARE DATA() VALUE
BNE  GETUNXT  DIFFERENT, TRY NEXT OCCURRENCE

*  OCCURRENCE SELECTED.
*  EITHER FORMAT DATA FOR DISPLAY, OR DELETE/RETURN ACCORDING TO FLG2
*
GETU4  DS   0H
SLR  R15,R15  AT LEAST ONE FIELD FOUND
LR   R1,R3  POINT TO OCCURRENCE MATCHED
TM   FLG2GETU,FLG2LOC  LOCATE ONLY?
B0   GETU99  YES, PASS BACK ADDR OCCURRENCE
*  TM   FLG2GETU,FLG2DEL  DELETE ?
*  B0   GETUDEL  YES, DO DELETE
*  (UNCOMMENT THE ABOVE 2 INSTR. TO OMIT DISPLAY OF DELETED FIELDS)
*
*  FORMAT FOR DISPLAY:
*  GET USRNM
MVC  LINE(L'USRNM),USRNM  USRNM TO MSG LINE
MVC  FLDNAME,USRNM  .AND FOR CLIST VARIABLE NAME
MVI  FLDNAMEM,C' '  RESET FIELD NAME SUFFIX @MC7
XC  VALUELEN,VALUELEN  RESET CLIST VAR. DATA LEN
LA   R0,L'USRNM  MSG LEN SO FAR
LTR  R2,R4  ANY USRDATA ?
BZ   GETU5  NO
*  GET USRDATA
BCTR  R2,0  LEN USRDATA LESS 1
MVC LINE+L'USRNM+1(*-*),USRDATA
EX R2,*-6 MOVE USRDATA TO MSG LINE
LA R1,USRDATA+5(R2) PAST USRDATA + FLAG_LEN TO USRFLG
* IF 'HIDDEN FIELD' THEN HIDE USRDATA
CLI USRMN,C'@' IS THIS A HIDDEN FIELD?
BNE GETU4H NO, NO NEED TO HIDE USRDATA VALUE
MVI LINE+L'USRNM+1,C'? OVERWRITE 1ST CHAR USRDATA IN MSG
LTR R2,R2 WAS DATA ONLY 1 CHAR LONG
BZ GETU4H YES, NO NEED TO PROPAGATE
BCTR R2,0 DECR LEN BY 1 MORE FOR PROPAGATE
MVC LINE+L'USRNM+2(*-_),LINE+L'USRNM+1
EX R2,*-6 PROPAGATE ? THROUGH FIELD
LA R2,1(R2) RESTORE TO LEN-1
* FORMAT USRFLG
GETU4H DS @H
LA R2,1(R2) REAL LEN OF USRDATA
SLR R0,R0
IC R0,0(R1) GET USRFLG
CVD R0,DWD1
OI DWD1+7,X'0F'
LA R1,LINEL+L'USRNM+3(R2) TO PLACE FOR FLAG
UNPK @(3,R1),DWD1+6(2)
*
* DISPLAY USRDATA AND WRITE TO CLIST VARIABLE
*
LA R0,L'USRNM+6(R2) LEN USRMN-USRDATA(R2)+FLAG+SPACES
GETU5 DS @H
LA R1,LINE POINT TO INFO TO DISPLAY
CLI KLIST+1,2 WAS 'NOLIST' SPECIFIED? @MC7
BE */8 YES, DON'T DISPLAY DATA @MC7
BAL R14,SPUTMSG LIST USRDATA OCCURRENCE
* CREATE USRNM VARIABLE
LA R1,LINEL+L'USRNM+1 POINT TO USRDATA PART IN DISPLAY LINE
ST R1,VALUEPTR PASS TO VAR. WRITE ROUTINE
ST R2,VALUELEN AND LEN OF USRDATA
BAL R10,SCVAR WRITE CLIST VARIABLE
* CREATE USRFLG VAR, SAME NAME AS USRNM WITH 'F' SUFFIX @MC7
LA R1,LINEL+L'USRNM+3(R2) TO USRFLG IN DISPLAY LINE @MC7
ST R1,VALUEPTR PASS TO VAR. WRITE ROUTINE @MC7
MVC VALUELEN,=F'3' LEN OF USRFLG FOR 'SCVAR' @MC7
L R2,NAMELEN REAL LEN OF LAST VAR NAME CREATED @MC7
LA R2,FLDNAME(R2) PAST END OF USRNM VAR. NAME @MC7
MVI @(R2),C'F' USRNM SUFFIX FOR FLAG VAR. NAME @MC7
BAL R10,SCVAR WRITE CLIST VARIABLE @MC7
MVI FLDNAMEEX,C' ' RESER FIELD NAME SUFFIX @MC7
*
SLR R0,R0 RESET LEN CURRENT MSG
TM FLG2GETU,FLG2DEL DELETE ?
BNO GETUNXT,NO, PROCESS NEXT OCCURRENCE
*
* DELETE A SINGLE OCCURRENCE
* THIS IS DONE BY MOVING BACK ALL OCCURRENCES FOLLOWING THE ONE
* TO DELETE AND ADJUSTING THE OVERALL LENGTH AND OCCURRENCE COUNT.
*
GETUDEL DS 0H
LR R6,R3 ADDR CURRENT OCC ('TO' ADDR)
LR R4,R3 @MC6
A R4,USRDATA1 @MC6
LA R4,USRDOCC1(R4) ADDR NEXT OCC ('FROM' ADDR) @MC6
LR R0,R4 @MC6
SR R0,R6 LEN CURRENT OCC.
*
L R7,USRDLN
SR R7,R0
ST R7,USRDATA LEFT
L R7,USRDATA
BCTR R7,0
ST R7,USRDATA ADJUST NO. OF OCCURRENCES LEFT
LTR R7,R7 ARE WE DELETING ONLY ENTRY LEFT
BNP GETU99 YES, RETURN
*
LR R7,R5 TO END OF ALL USERDATA
SR R7,R4 LEN FOLLOWING CURRENT OCCURRENCE
BNP GETU99 NONE, LAST ONE, NO MOVE NEEDED
LR R1,R5 * SAVE R5 OVER MVCL @MC6
LR R5,R7 LEN FOLLOWING CURRENT OCCURRENCE @MC6
MVCL R6,R4 SHUFFLE BACK OVER CURRENT OCC @MC6
LR R5,R1 * RESTORE R5 @MC6
SR R5,R0 NEW END ADDR OF ALL USERDATA
B GETU1 SEE IF ANY MORE TO DELETE
*
* PROCESS NEXT OCCURRENCE IF ANY LEFT
*
GETUNXT DS 0H
A R3,USRDATA1
LA R3,USRDOCC1(R3) TO NEXT USERDATA OCCURRENCE
CR R3,R5 END YET ?
BM GETU1 NO
*
GETU99 L R10,SAVER10
LM R2,R8,SAVESUB+12 LEAVE R0,R1,R15
BR R10
DROP R3 @MC4

**********************************************************************************************
*  *
* SUBROUTINE: CHECK CALLERS AUTHORISATION TO REQUESTED FUNCTION *
*  *
* ON ENTRY: 'AUTHENT' SHOULD BE SET UP WITH THE NAME TO CHECK, *
* IN THE FORMAT 'CLASS.SEGMENT.FIELD' *
*  *
**********************************************************************************************
* EXIT VIA R10

* R15 = Ø USER AUTHORISED

* R15 = 4 USER NOT AUTHORISED

* NOTE: IF THE RESOURCE IS NOT DEFINED ACCESS IS NOT ALLOWED.

**************************************************************************

SAUTHCHK DS ØH
STM R1,R8,SAVESUB
SLR R15,R15
AUTHCODE,Ø
AUTHRET YES, IMMED RETURN
TM FLGIRAC,ACESPEC RACF SPECIAL CAN DO ANYTHING
BNO SAUTHØ5 NOT SPECIAL @MC6

*SPECHK B SAUTHØ5 DISABLE SYSTEM-SPECIAL OVERRIDE @MC6
* MNOTE 1,’SYSTEM-SPECIAL SUPPORT DISABLED’ @MC6
* @MC6

@TRACE ‘AUTHORISED BY SYSTEM-SPECIAL’ @MC6
SLR R15,R15
INDICATE AUTHORISED @MC6
B AUTHRET AND RETURN @MC6

SAUTHØ5 DS ØH @MC6
LA RØ,AUTHENT CALCULATE... @MC4
LA R1,AUTHENT+L’AUTHENT REAL LENGTH... @MC4
TRT AUTHENT,TRTAB2 OF... @MC4
SR R1,RØ FIELD... @MC4
STH R1,AUTHENTL+2 PROFILE NAME @MC4

LA R2,2 ‘READ’ ACCESS
CLI FUNCODE,FUNCGET IS IT ‘GET’ FUNCTION
BE ++Ø YES, READ ACCESS REQUIRED
LA R2,4 UPDATE REQUIRED FOR ANYTHING ELSE
@TRACE ‘CHECKING ACCESS TO: ‘,(AUTHCLS+1,B),’ ‘,(AUTHENT,26)

* IF CALLER’S OWN USER PROFILE SEE IF &RACUID HAS REQUIRED ACCESS

CLC RCLASS,=CLB’USER’ IS IT A USER PROFILE?
BNE SAUTHØ0 NO, &RACUID NOT APPLICABLE
CLC TSUSER,PROFNAME+1 IS CALLER’S OWN?
BNE SAUTHØ0 NO, &RACUID NOT APPLICABLE
MVC ACLUDER,=CLB’&RACUID’ USERID TO CHECK FOR ON ACCLST
@TRACE ‘ TRYING &RACUID’ @MC5
ICHEINTY LOCATE,TYPERGEN,CLASS=AUTHCLS+1,ENTRYX=AUTHENTL, +
ACTIONS=ACTN5,OPTIONS=(ACTION,TESTM),WKAREA=RACWA, +
RELEASE=1.9,MF=(E,INTY2)
LTR R15,R15 WAS INTY OK?
BNZ SAUTHØ0 NO, USE NEXT CHECK
CLI TEST5+1,Ø WAS &RACUID ON ACCESS LIST?
BNE SAUTHØ0 NO, DO NEXT CHECK
@TRACE ‘ &RACUID FOUND ...’ @MC5
CLC ACLENT1,RACWA+32 DID WE GET ACL ENTRY AS EXPECTED
BNE SAUTH10 NO, DO NEXT CHECK
LR R0,R2 ACCESS LEVEL REQUIRED
SLL R0,3 SAME FORMAT AS IN ACL ENTRY
CLM R0,1,RACWA+48 COMPARE REQD. LEVEL TO &RACUID LEVEL
BNP AUTHRET OK, &RACUID COVERS IT, RETURN R15=0.
@TRACE ‘&RACUID INSUFFICIENT LEVEL.’ @MC5
SAUTH10 DS 0H
*
* CHECK CALLER'S ACCESS TO ‘CLASS.SEGMENT.FIELD’
* NOTE: NO LOGGING IS DONE ON RACHECK
*
RACROUTE REQUEST=AUTH,CLASS=AUTHCLS,RELEASE=1.9, +
   ATTR=(2),ENTITYX=AUTHENTL,LOG=None,
   WORKA=RACWA, MF=(E,RACHECKL)
MVC WORK1(6),=CL6'OK’ @MC5
LTR R15,R15 @MC5
BZ **+Ø @MC5
MVC WORK1(6),=CL6'FAILED’ @MC5
@TRACE ‘ RACHECK ‘,(WORK1,6) @MC5
*
PASS BACK R15 FROM RACHECK
XC RACWA,RACWA REINSTATE WORK AREA
L R0,=A(RACWAL) FOR ICHEINTY TO USE @MC1
ST R0,RACWA ..... AUTHRET DS 0H
LM R1,R8,SAVESUB BR R1Ø
******************************************************************************
*
* SUBROUTINE: CHECK CALLERS AUTHORITY IN GROUP (OR ANY OF ITS
* OWNING GROUPS) THAT OWNS THE PROFILE.
*
* ‘GROUP-SPECIAL’ - ALLOWS USERDATA UPDATE FUNCTIONS TO ANY PROFILE
* OWNED BY THE GROUP OR ANY OF ITS SUB-GROUPS
* ‘GROUP-AUDITOR’ - ALLOWS THEM TO READ ANY PROFILE OWNED BY THE
* GROUP OR ANY OF ITS SUB-GROUPS
* AUTH=CONNECT - ALLOWS THEM TO READ USER-PROFILES OWNED
* DIRECTLY BY THAT GROUP ONLY.
* - ALLOWS THEM TO READ THAT GROUP PROFILE ONLY.
*
* ON ENTRY: ‘PROFNAME’ SHOULD BE SET UP WITH THE PROFILE NAME.
* (1ST BYTE = LENGTH)
* EXIT VIA R1Ø
* R15 = Ø USER AUTHORISED
* R15 = 4 USER NOT AUTHORISED
*
******************************************************************************
SCHKOWN DS 0H @MC4
STM R1,R8,SAVESUB @MC4
XC RACWA,RACWA CLEAR WORK AREA @MC4
L R0,=A(RACWAL) GET LENGTH @MC4

ST R0,RACWA AND STORE IN WORK AREA @MC4
@TRACE 'CHECKING AUTHORITY OVER ',(RCLASS,8),' ', @MC5 +
(PROFNAME=1,44) @MC5
* IF PROFILE IS A GROUP, CHECK AUTHORITY WITHIN GROUP FIRST @MC5
MVC OWNER,PROFNAME+1 GROUPNAME=PROFILE NAME @MC5
MVC RESOWNER,PROFNAME+1 FOR "AUTH=CONNECT" CHECK ONLY @MC5
CLC RCLASS,=CL8'GROUP' IS PROFILE A GROUP-PROFILE @MC5
BE SCHKO5 YES, CHECK CALLER'S AUTH IN IT @MC5
* GET OWNER OF REQUESTED PROFILE (ASSUME OWNER IS A GROUP) @MC4
ICHEINTY LOCATE,ACTN=(ACTN6),RELEASE=1.9, @MC4 +
OPTIONS=(ACTION),WKAREA=RACWA, @MC4 +
MF=(E,INTY1) @MC4
MVC RESOWNER,RACWA+32 SAVE RESOURCE OWNER @MC5
MVC OWNER,RACWA+32 SET UP FOR GROUP TREE CHECK @MC5
@TRACE 'PROFILE OWNER=',(OWNER,8) @MC5
* GET CONNECT INFO FOR CALLER FROM (ASSUMED) OWNING GROUP @MC5
SCHKO5 DS 0H @MC5
MVC CONGROUP,OWNER GET OWNER OF GROUP/RESOURCE @MC5
MVC ACLUSER,TSUSER SET UP TSO USERID FOR ACTN8 @MC5
LA R0,CONGROUP CALCULATE... @MC5
LA R1,CONGROUP+L'CONGROUP REAL LENGTH... @MC5
TRT CONGROUP,TRTAB2 OF... @MC5
SR R1,R0 GROUP... @MC5
STH R1,CONGPL+2 NAME @MC5
@TRACE 'GETTING CALLERS AUTHORITY IN GROUP ',(CONGROUP,8) @MC5
ICHEINTY LOCATE,TYPE='GRP',ENTRYX=CONGPL,
ACTIONS=(ACTN6,ACTN8),
OPTIONS=(ACTION,TESTM),WKAREA=RACWA,
RELEASE=1.9,MF=(E,INTY2) @MC5
C R15,F'12' DOES GROUP EXIST? @MC5
BE SCHKORC4 NO, TOP OF GROUP TREE @MC5
MVC OWNER,RACWA+32 GROUP OWNER (NEXT 1 UP TREE) @MC5
@TRACE ' (GROUP OWNER= ',(OWNER,8),')' @MC5
CLI TEST8+1,0 IS CALLER CONNECTED TO GROUP? @MC5
BNE SCHKO5 NO, KEEP GOING UP GROUP TREE @MC5
@TRACE ' CALLER CONNECTED TO ',(CONGROUP,8) @MC5
MVC USERACS,RACWA+44 SAVE USER'S "AUTH" IN GROUP @MC5
* ONLY WHEN PROFILE CLASS IS 'USER'.... @MC5
* CHECK FOR AUTH=CONNECT IN USER'S OWNING GROUP @MC5
CLC RCLASS,=CL8'USER' IS IT USER-PROFILE REQUESTED? @MC5
BE SCHKO8 YES, CHECK FOR AUTH=CONNECT @MC5
CLC RCLASS,=CL8'GROUP' GROUP-PROFILE REQUESTED? @MC5
BNE SCHK10 NO, NORMAL CHECKING @MC5
SCHKO8 DS 0H @MC5
@TRACE ' CHECKING IF ',(CONGROUP,8),' IS PROFILE OWNER'
CLC CONGROUP,RESOWNER IS THIS THE USER'S OWNING GROUP? @MC5
* (OR THE GROUP ITSELF IF CLASS=GROUP) @MC5
BNE SCHK10 NO, NORMAL CHECKING @MC5
@TRACE ' YES, CHECKING IF AUTH=CONNECT...' @MC5
CLI USERACS,X'40' AUTH=CONNECT AT LEAST? @MC5
* SUBROUTINE: CONVERT DATA FROM INTERNAL FORMAT *

**Implementation Details:**

```
BM SCHK10 NO, CONTINUE CHECKING @MC5
@TRACE ' YES, CHECKING IF READ REQUEST...' @MC5
CLI FUNCODE,FUNCGET IS IT READ OPERATION? @MC5
BE SCHKORC0 YES, ALLOW @MC5
@TRACE ' NOT READ, AUTH=CONNECT NOT ENOUGH.' @MC5
* GET GROUP CONNECT INFO FROM CALLER'S USER PROFILE @MC4
SCHK10 DS @H @MC5
@TRACE ' GETTING INFO FOR ',(TSUSER,8),' CONNECT TO ', @MC5 +
(CONGROUP,8),'.
ICHEINTY LOCATE,TYPE='USR',ENTRY=TSUSERL, @MC4 +
  ACTIONS=(ACTN7,ACTN7A,ACTN7B),TESTS=(TEST7), @MC4 +
  OPTIONS=(ACTION,TESTM),WKAREA=RACWA, @MC4 +
  RELEASE=1.9, MF=(E,INTY2) @MC4
* CHECK USERS AUTHORITY IN GROUP @MC4
@TRACE ' CHECKING GROUP CONNECT ATTRIBUTES' @MC5
C R15,=F'52' DID TESTS FAIL @MC4
BE SCHK05 YES, GO BACK UP GROUP TREE @MC4
LTR R15,R15 OTHER ERROR? @MC4
BNZ SCHKORC4 USER PROFILE NOT FOUND PERHAPS? @MC4
CLI TEST7+1,Ø WAS USER CONNECTED TO GROUP? @MC4
BNE SCHK05 NO, DO NEXT CHECK @MC4
CLI RACWA+44,X'80' GROUP-AUDITOR? @MC4
BNE SCHK20 NO @MC4
MVI GRPAUTH,GRPAUD INDICATE GROUP AUDITOR @MC4
@TRACE ' GROUP-AUDITOR FOUND' @MC5
SCHK20 CLI RACWA+49,X'80' GROUP-SPECIAL? @MC4
BNE SCHK30 NO @MC4
MVI GRPAUTH,GRPSPEC INDICATE GROUP SPECIAL @MC4
@TRACE ' GROUP-SPECIAL FOUND' @MC5
SCHK30 DS ØH @MC5
@TRACE ' CHECKING IF GROUP ATTRBS ENOUGH...' @MC5
CLC GRPAUTH,FUNCODE GROUP AUTHORITY ENOUGH FOR FUNC? @MC4
BL SCHK05 NO, BACK UP TREE @MC4
SCHKORC0 DS ØH @MC5
@TRACE ' YES, ACCESS ALLOED BY GROUP: ',(CONGROUP,8)
SLR R15,R15 @MC5
B SCHKORET @MC5
SCHKORC4 DS ØH @MC5
@TRACE 'ACCESS NOT ALLOWED BY GROUP'
LA R15,4 @MC5
SCHKORET DS ØH @MC4
XC RACWA,RACWA REINSTATE WORK AREA @MC4
L R0,=(A(RACWA) FOR ICHEINTY TO USE AGAIN @MC4
ST R0,RACWA .....
LM R1,R8,SAVESUB @MC4
BR R10 @MC4
```

* ON ENTRY: R3 = ADDR DATA
  * R2 = LEN DATA
  * FLIND = CONVERSION CHAR (P/X/B)
* ON EXIT : R3 = ADDR CONVERTED DATA
  * R2 = LEN CONVERTED DATA
* RETURNS VIA R10
*
*********************************************************************
SCNVDAT DS ØH
STM R2,R8,SAVESUB
LTR R0,R2
LEN DATA
BZR R10
NONE TO CONVERT @MC2
LR R1,R3
ADDR DATA
BCTR R2,0
CLI FLIND+3,C'P'
PACKED DECIMAL ?
BNE SCNVB
NO
*
* CONVERT PACKED DEC.
*
UNPK WORK1,Ø(*-*),R1
EX R2,*-6
OI WORK1+L'WORK1-1,X'F0'
LR R2,R0
INPUT LEN
SLL R2,1
OUTPUT LEN = (INPUT_LEN*2)-1
BCTR R2,0
LA R3,WORK1+L'WORK1
PAST END OF WORK FIELD
SR R3,R2
BACK TO START OF CONVERTED DATA
B SCNVRET
SCNVB DS ØH
CLI FLIND+3,C'B'
BINARY ?
BNE SCNVX
NO
*
* CONVERT BINARY
*
XC DWD1,DWD1
MVC DWD1(*-*),Ø(R3)
EX R2,*-6
L R1,DWD1
LA R2,4
MAX NO. BYTES
SR R2,R0
LESS ACTUAL = BYTES TO SHIFT RIGHT
SLL R2,3
*B = NO. BITS TO SHIFT RIGHT
SRL R1,Ø(R2)
RIGHT ALIGN IN REG1
CVD R1,DWD1
MVC WORK1,=15X'20'
MVI WORK1+13,X'21'
LA R1,WORK1+14
IN CASE ZERO @MC2
EDMK WORK1(15),DWD1
LA R2,WORK1+15
SR R2,R1
LEN OF SIG. RESULT @MC2
LR R3,R1
TO 1ST SIG DIGIT. @MC2
SCNVX DS 0H
CLI FLDIND+3,C’X’ HEX ?
BNE SCNVRET NO
*
* CONVERT HEX
*
LA R2,1(R2) INCL. DUMMY BYTE AT END OF SOURCE
UNPK WORK1,Ø(*-*,R1)
EX R2,*-6
TR WORK1,TRTAB1 TRANSLATE TO EBCDIC
LR R2,RØ INPUT LEN
SLL R2,1 OUTPUT LEN = INPUT_LEN*2
LA R3,WORK1+L’WORK1-1 PAST END OF TRANSLATED DATA
SR R3,R2 BACK TO START OF CONVERTED DATA
SCNVRET LM R4,R8,SAVESUB+8
BR R10
******************************************************************************
*
* SUBROUTINE: WRITE DATA TO CLIST VARIABLE
*
* ON ENTRY: VALUELEN = LEN OF DATA TO WRITE
* VALUEPTR = ADDR OF DATA TO WRITE TO VARIABLE
* FLDNAME = NAME OF FIELD
* VARNO = VARIABLE NO. (TO SUFFIX FLDNAME)/ OR Ø
* IF VALUELEN=Ø THE VARIABLE IS SET TO NULL.
* ON EXIT: DATA WRITTEN TO CLIST VARIABLE
* VARNO INCREMENTED +1 IF RPT.GRP.OCCURRENCE (OCCNO=Ø)
* RETURNS VIA R10
******************************************************************************
SCVAR DS 0H
STM R15,R4,SAVESUB2
MVC VARNAME,BLANKS ENSURE NO RESIDUE FROM LAST TIME
MVC VARNAME(L’FLDNAME+L’FLDNAMEX),FLDNAME NAME+SUFFIX @MC7
LA R3,VARNAME+L’VARNAME-1 TO END OF VARN NAME FIELD
LA R2,L’VARNAME MAX LEN VARIABLE NAME
SCVAR5 DS 0H
CLI Ø(R3),C’ ‘ SCAN BACK FOR LAST CHAR OF NAME
BNE SCVAR10 FOUND IT
BCTR R3,Ø TO PREV CHAR
BCT R2,SCVAR5 DECR LEN AND SCAN
ABEND 99,DUMP SHOULD NEVER HAPPEN
SCVAR10 DS 0H
ST R2,NAMELEN PUT NAME LENGTH IN PARMS
CP OCCNO,=P’Ø’ IS THIS A RPT GRP OCCURRENCE
BE SCVAR20 NO: VARIABLE NAME IS READY
MVC WORK1(5),=5X’20’ SET UP EDIT MASK
AP VARNO,=P’1’ INCR. FOR NEXT VAR. NO.

EDMK WORK1(5),VARNO EDIT AND NOTE 1ST SIG. CHAR.
LA R2,WORK1+5 PAST EDITED VALUE
SR R2,R1 LEN SIG. RESULT CHARS
BCTR R2,Ø LESS 1 FOR EX
MVC 1(*-,R3),Ø(R1) MOVE NUMBER IN AS SUFFIX
MVC VARNUM,Ø(R1)
EX R2,*-6 SAVE FOR COUNT VARIABLE LATER
LA R2,1(R2) RESTORE LEN
ST R2,VARNUM SAVE LEN OF NUMBER
A R2,NAMELEN ADD NUMERIC PART TO NAME
ST R2,NAMELEN UPDATE
SCVAR2Ø DS ØH
LA R1,CT441PRM PARGS FOR IKJCT441
L R15,16 CVT
L R15,CVTTVT-CVTR15 TSVT
ICM R15,15,TSVTACC-TSVTR15 IKJCT441
BZ SCVLNK
BASR R14,R15
B SCVARET
SCVLNK LINK EP=IKJCT441
SCVARET DS ØH
LM R15,R4,SAVESUB2
BR R1Ø
EJECT

******************************************************************************
*                                  @MC6
*          E S T A E     E X I T       @MC6
*                                  @MC6
* IF RTM DID NOT SUPPLY AN SDWA THEN JUST PERCOLATE THE ABEND.   @MC6
* ELSE GO TO RETRY ROUTINE TO ISSUE BASIC DIAGNOSTIC MESSAGE    @MC6
* BEFORE TERMINATING.                                            @MC6
*                                                             @MC6
* IF AN SDWA IS PROVIDED, R1 POINTS TO IT.                       @MC6
* IF NO SDWA, R1=ABEND CODE, R2=ADDR ESTAE PARM LIST.           @MC6
*                                                             @MC6
******************************************************************************
PUSH USING
DROP R12 DROP MAIN BASE REGS
ESTAEX DS ØH
USING SAVEAREA,R13 ADDRESS MAINLINE WORK AREA
LR R13,R2 AND POINT TO IT (IF NO SDWA)
LR R12,R15 LOAD BASE ADDR FOR ESTAE EXIT
USING ESTAEX,R12
LR R11,R1 POINT TO SDWA
USING SDWA,R11 ADDRESS SDWA IF WE HAVE ONE
CH R0,NOSDWA DID RTM GET AN SDWA
BE ESTAPERC NO, DO WITHOUT
L R13,SDWAPARM GET ADDR MAIN WORK AREA
MVC SDWASRSV,RECREGS SET REGS AS SAVED BEFORE ABEND
* MVC ABCODE,SDWAICD1  SAVE PGM IRPT CODE FOR RETRY RTN.
MVC ABCMPC,SDWACMPC  ABEND COMPLETION CODE
MVC ABPSW,SDWAEC1    PSW AT ABEND
MVC RECREGS,SDWAGRSV  REGS AT ABEND
L  R2,SDWANXT1     NEXT INSTR.
ICM R0,15,SDWAEP A  EPA OF PGM IF NOT SPVR STATE
BNZ ESTA10   USE IT
ICM R1,15,SDWARBAD ADDR ABENDING RB IF SPVR STATE
BZ ESTA10  MUST BE PROB PGM
L  R1,12(R1)  CDE FOR ABENDING RB
L  R0,16(R1)  EPA OF ABENDING PGM
   ESTA10 SR R2,R0  OFFSET INTO PROGRAM
   CH R2,LENPGM  IS OFFSET OUTSIDE OUR PGM ?  @MC6
   BNP +*6  NO, PROBABLY VALID  @MC6
   SLR R2,R2  ABEND NOT IN OUR CSECT  @MC6
   STCM R2,15,ABOFFS PASS TO RETRY
*
* RETURN TO RTM TO ATTEMPT RETRY
*
   ICM R2,15,RETRYADR GET RETRY ADDRESS
   SETRP RC=4, RC FOR RETRY +
      DUMP=NO, +
      RETADDR=(2), ADDR TO RETRY +
      FRESDWA=YES, FREE SDWA +
      RETREGS=YES, RESTORE REGS FROM SDWASRSV +
      WKAREA=(11) ADDR SDWA
      BR R14 RETURN TO ATTEMPT RETRY
*
* RETRY WILL NOT BE ATTEMPTED, CONTINUE WITH ABEND
*
   ESTAPERC DS 0H
      SETRP RC=0, CONTINUE WITH ABEND +
      WKAREA=(11) ADDR SDWA
      BR R14 RETURN TO CONTROL PGM.
   SPACE
   RETRYADR DC A(RETRYRTN) RESUME ADDR IN MAINLINE CODE
   NOSDWA DC H'12' R0 CONTENTS IF NO SDWA PROVIDED
   LENPGM DC Y(INTYEND-INTY) LEN OF OUR PGM  @MC6
   POP USING
   EJECT

*******************************************************************************
*  ROUTINE ENTERED WHEN THE ESTAE HAS SPECIFIED ‘RETRY’
*  ** THIS CODE IS PART OF THE MAINLINE PROGRAM **
*
*  INFORM USER OF ABEND AND TERMINATE PROGRAM
*
*******************************************************************************
RETRYRTN DS 0H
```assembly
ESTAE Ø CANCEL RECOVERY ROUTINE
MVI FLDIND+3,C'X' TELL SCNVDAT TO CONVERT TO EXT. HEX
*
LA R3,ABCMP C POINT TO ABEND COMPLETION CODE
LA R2,L'ABCMP C LENGTH OF COMPLETION CODE
BAL R10,SCNVDAT CONVERT TO DISPLAYABLE
MVC EM99CMP,Ø(R3)
*
LA R3,ABCOD C POINT TO PGM IRPT CODE
LA R2,L'ABCOD C LENGTH OF PIC
BAL R10,SCNVDAT CONVERT TO DISPLAYABLE
MVC EM99PIC,Ø(R3)
*
LA R3,ABPSW C POINT TO ABEND PSW
LA R2,4 C LENGTH OF 1ST HALF OF PSW
BAL R10,SCNVDAT CONVERT TO DISPLAYABLE
MVC EM99PSWA,Ø(R3)
LA R3,ABPSW+4 C POINT TO ABEND PSW BYTES 4-7
LA R2,4 C LENGTH OF 2ND HALF OF PSW
BAL R10,SCNVDAT CONVERT TO DISPLAYABLE
MVC EM99PSWB,Ø(R3)
*
DC ABOFFS,ABO C WAS ABEND IN OUR CSECT ?
B2 RETRYØ5 C NO, NO OFFSET TO REPORT THEN
LA R3,ABO C POINT TO ABEND OFFSET
LA R2,L'ABO C LENGTH OF OFFSET
BAL R10,SCNVDAT CONVERT TO DISPLAYABLE
MVC EM99OF,Ø(R3)
RETRYØ5 DS ØH
*
MVC RETCODE,=F'16' TPUT EMS99,LEMSG99 B RETURN C RETURN WITHOUT ESTAE CANCEL
*
EMSG99 DC C'MCI99E: ABEND S'
EM99CMP DC C'XXX','C' PIC'
EM99PIC DC C'XX','C' AT +'
EM99OF DC C' N/A ',C' PSW '
EM99PSWA DC CL8' '
EM99PSWB DC CL8' '
DC C' ' PAD
LEMSG99 EQU ~*LEMSG99
EJECT
** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
** STORAGE AREAS **
** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
LTORG ,
**********************************************************************************************************
```
* CONSTANTS

****************************************************************************************************
*
* !!! WARNING !!! ... ENSURE THE LENGTHS ARE ALSO CHANGED WHEN
*  CHANGING ANY OF THE FOLLOWING FIELDS
*
DFLTCLS DC AL1(4),CL8'USER' DEFAULT CLASS
DFLTSEG DC AL1(4),CL8'BASE' DEFAULT SEGMENT NAME
AUTHCLS DC AL1(5),CL8'FIELD' CLASS FOR AUTH CHECKS
AUTHPRF DC AL1(0),CL8' ' PREFIX FOR AUTH. RESOURCE NAME
  (AUTHPRF MUST INCLUDE TRAILING DOT)
AUTHUPRF DC AL1(8),CL8'USERDATA' PREFIX FOR USERDATA CHECKING
ACLEN1 DS 0CL16 CHECK FOR &RACUID ON ACL
  DC AL4(8),CL8'&RACUID',AL4(1)
BLANKS DC CL80' ' DC '0' $123456789abcdef' (MUST IMMED. PRECEDE TRTAB1)
TRTAB1 EQU *.256,256 HEX TRANSLATE TABLE
TRTAB2 DC XL256'0' @MC4
  ORG TRTAB2+C' ' @MC4
  DC C' ' @MC4
  ORG , @MC4

****************************************************************************************************
*
* NON-REENTRANT WORK AREA

****************************************************************************************************
*
* MAIN ICHEINTY, USED FOR ALL FIELD REQUESTS, (LOCATE AND ALTER)
*
INTY1 ICHEINTY LOCATE,ACTIONS=(*-*,**), GENERIC=NO, +
  RELEASE=1.9,MF=L
LINTY EQU *.INTY1
*
* INTY USED FOR AUTHORISATION CHECKS
*
INTY2 ICHEINTY LOCATE,ACTIONS=(*-*,**,**),TESTS=(*-*,**,**), +
  GENERIC=NO,RELEASE=1.9,MF=L
*
* ACTION USED AS TEMPLATE WHEN BUILDING FOR FIELDS REQUESTED
*
ACTN1 ICHEACTN FIELD=DUMMY,RELEASE=1.8.1,FDATA=(0,0),MF=L
LACTN EQU *.ACTN1
*
* CREATE LIST OF ACTION POINTERS (FLDEF)
*
INTYF ICHEINTY FLDEF,ACTIONS=(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0), +
  RELEASE=1.9,MF=L
*
* RETRIEVE/REPLACE ALL USERDATA
*
ACTN2 ICHEACTN FIELD=USRCNT,FDATA=(*-*,**), +

RELEASE=1.8.1
ACTN2A ICHEACTN FIELD=USRCNT,FLDATA=(*-*,*-*) , GROUP=YES, +
          RELEASE=1.8.1
          *
          * ADD USERDATA OCCURRENCE
          *
ACTN3 ICHEACTN FIELD=USERDATA,FLDATA=(*-*,*-*) , +
          RELEASE=1.8.1
          *
          * DELETE ALL USERDATA
          *
ACTN4 ICHEACTN FIELD=USRCNT,FLDATA='DEL',GROUP=YES, +
          RELEASE=1.8.1
          *
          * GET ACCESS LIST ENTRY
          *
ACTN5 ICHEACTN FIELD=ACL,TESTS=(TEST5) , +
          RELEASE=1.8.1
TEST5 ICHETEST FIELD=USERID,FLDATA=(8,ACLUSER), +
          RELEASE=1.8.1
          *
          * FORMAT OF DATA RECEIVED FROM PRECEDING ACTN5.
          * AL4(8),CL8'USERID'
          * @MC4
          *
          * GET OWNER
          *
          *
ACTN6 ICHEACTN FIELD=OWNER, @MC4 +
          RELEASE=1.8.1 @MC4
          *
          * @MC4
          *
          * GET CONNECT ENTRY INFORMATION
          *
ACTN7 ICHEACTN FIELD=CGAUTHOR,TESTS=TEST7, +
          @MC4 +
          RELEASE=1.8.1 @MC4
ACTN7A ICHEACTN FIELD=CGGRPaud,TESTS=TEST7, +
          @MC4 +
          RELEASE=1.8.1 @MC4
ACTN7B ICHEACTN FIELD=CGFLAG2,TESTS=TEST7, +
          @MC4 +
          RELEASE=1.8.1 @MC4
TEST7 ICHETEST FIELD=CGGRPnm,FLDATA=(8,CONGROUP), +
          @MC4 +
          RELEASE=1.8.1 @MC4
          *
          * FORMAT OF DATA RECEIVED FROM PRECEDING ACTN7.
          * AL4(8),CL8'CGAUTHOR',AL4(1),XL1'CGGRPaud',AL4(1),XL1'CGFLAG2' @MC4
          *
          * GET CONNECT ENTRY INFORMATION FROM GROUP PROFILE
          *
ACTN8 ICHEACTN FIELD=USERACS,TESTS=TEST8, +
          RELEASE=1.8.1
TEST8 ICHETEST FIELD=USERID,FLDATA=(8,ACLUSER), +
          RELEASE=1.8.1
          *
          * FORMAT OF DATA RECEIVED FROM PRECEDING ACTN8, AT RACWA+28
          * AL4(1),XL1'USERACS'
          * @MC4
* MISCELLANEOUS ICHEACTN DATA FIELDS
*
ACLUSER DS CL8 USER ON ACCESS LIST
CONGRPL DC H'8',H'0' LEN CONGROUP (ENTRYX FORMAT) @MC4
CONGROUP DS CL8 USER CONNECTED TO GROUP @MC4
*
PRINT NOGEN
RACSTATL RACROUTE MF=L,RELEASE=1.9,REQUEST=STAT
RACHECKL RACROUTE MF=L,RELEASE=1.9,REQUEST=AUTH
********************************************************************
* IKJCT441 PARAMETER LIST
********************************************************************
CT441PRM DC A(ECODE) ADDR OF ENTRY CODE
DC A(NAMEPTR)
DC A(NAMELEN)
DC A(VALUEPTR)
DC A(VALUELEN)
DC X'80000000' TOKEN (+END OF LIST)
*
ECODE DC A(TSVEUPDT) UPDATE/CREATE VARIABLE
NAMEPTR DC A(VARNAM) ADDR OF VARIABLE NAME
NAMELEN DC A(*-*) LEN OF VARIABLE NAME
VALUEPTR DC A(*-*) ADDR OF VARIABLE VALUE
VALUELEN DC A(*-*) LEN OF VARIABLE VALUE
FLDNAM DC CL8' ' FIELD NAME
FLDNSMEX DC C' ' FIELD NAME SUFFIX (FOR USRFLG) @MC7
VARNAM DC CL5'0' CLIST VARIABLE NAME
VARNUM DC CL5'0' RPT GROUP OCCURRENCE NUMBER
VARNUML DC F'1' LEN. OF ABOVE FIELD
*
********************************************************************
* AREA TO CREATE USERDATA FOR WRITING TO PROFILE
********************************************************************
UDLEN DS F LEN OCCURRENCE
UDATA EQU *
DC AL4(8) LEN OF USRNM
UDNAME DC CL8' ' NAME OF OCCURRENCE (USRNM)
UDDATAL DC AL4(*-*) LEN OF UDDATA (USRDATA)
UDDATA DS OCL255 DATA (USRDATA)
DC AL4(1) LEN OF FLAG
UDFNG DC X'0' USRFLG (UNUSED)
UDL EQU *-UDATA LEN OCCURRENCE (LESS UDDATA)
ORG UDATA+UDL+L'UDDATA ENSURE WE ARE PAST END OF AREA
********************************************************************
* PARSE PARAMETER LIST
********************************************************************
PUSH PRINT
PRINT GEN
PCLPDL IKJPARM
KFUNC IKJTERM 'FUNCTION',TYPE=CNST,RSVWD=FUNC,
PROMPT='FUNCTION CODE',
HELP='GET (RETRIEVE DATA) REP/ADD/DLM (UPDATE)'

KCLASS IKJKEYWD
IKJNAME 'CLASS', ALIAS='CL', SUBFLD=CLSS

KPROF IKJKEYWD
IKJNAME 'PROF', ALIAS='PR', SUBFLD=PRF

KFLDS IKJKEYWD
IKJNAME 'FIELDS', ALIAS='FI', SUBFLD=FLDS

KSEG IKJKEYWD
IKJNAME 'SEGMN', ALIAS='SEG', SUBFLD=SEGNM

KDATA IKJKEYWD
IKJNAME 'DATA', ALIAS='DA', SUBFLD=DAT

KFLG IKJKEYWD
IKJNAME 'FLAG', ALIAS='FL', SUBFLD=FLG

KUSRDATA IKJKEYWD
IKJNAME 'USERDATA', ALIAS='USR'

KRG IKJKEYWD
IKJNAME 'RGROUP', ALIAS='RG' MUST BE 1ST UNDER KRG @MC7
IKJNAME 'NORGROUP', ALIAS='NORG' MUST BE 2ND UNDER KRG @MC7

KLST IKJKEYWD DEFINED='LIST' @MC7
IKJNAME 'LIST' MUST BE 1ST UNDER KLST @MC7
IKJNAME 'NOLIST', ALIAS='NOL' MUST BE 2ND UNDER KLST @MC7

KMSG IKJKEYWD DEFINED='MSG' @MC7
IKJNAME 'MSG' MUST BE 1ST UNDER KMSG @MC7
IKJNAME 'NOMSG', ALIAS='NOM' MUST BE 2ND UNDER KMSG @MC7

KGENERIC IKJKEYWD,
IKJNAME 'GEN', ALIAS='GEN' @MC2

KTRACE IKJKEYWD,
IKJNAME 'TRACE', ALIAS='TR' @MC5

KDEBUG IKJKEYWD,
IKJNAME 'DEBUG', ALIAS='DEB' @MC6

* END OF MAIN PART OF LIST

FUNC IKJRSVDW

* KEEP THE FOLLOWING FUNCTION IKJNAMES IN ORDER; THE ORDER THEY
* APPEAR IN IS THE SAME AS THE RELATED 'EQU' VALUES FOLLOWING.

IKJNAME 'GET' GET STD OR USERDATA
IKJNAME 'REP' REPLACE, OR ADD IF NOT THERE
IKJNAME 'ADD' ADD, EVEN IF SAME USRNME EXISTS
IKJNAME 'DEL' DELETE

Funct GET EQU 1
Funct REP EQU 2
Funct ADD EQU 3
Funct DEL EQU 4

GROUPAUD EQU FuncGET GROUP-AUDITOR ALLOWE BCHALC 'GET' @MC4
GROUPSPEC EQU 255 GROUP-SPECIAL ALLOWE ALL UPD. @MC4

CLASS IKJSUBF
CLASS IKJIDENT 'CLASS', MAXLENTH=8, FIRST=ALPHA, OTHER=ALPHANUM
PRF IKJSUBF
PRF IKJIDENT 'PROFILE NAME', CHAR, MAXLENTH=44, FIRST=ANY, OTHER=ANY

FLDS IKJSUBF
AUTHENT DS CL26 ENTITY NAME FOR AUTH CHECK
FLGIRAC DS X COPY OF ACEEFLAG1
FUNCODE DS X FUNCTION CODE (1ST OPERAND IN CMD)
FLG2GETU DS X FLAG FOR SGETUDAT SUBROUTINE
FLG2LOC EQU X'80' LOCATE WITHOUT DISPLAY
FLG2DEL EQU X'40' DELETE OCCURRENCE(S)
OCCNO DS 0D,PL8 OCCURRENCE NO. OF REPEAT GROUPS
SOCCNO DS 0D,PL8 OCCURRENCE NO. SELECTED VIA DATA()
VARNO DS PL3 VARIABLE NO. FOR RPT GRP FIELDS
SELVAR DS XL32 OCCURRENCE SELECTION MASK @MC3
USERACS DS X USERS AUTH IN GROUP @MC5
OWNER DS CL8 @MC5
RESOWNER DS CL8 OWNER OF REQUESTED PROFILE @MC5
ABCMPC DS XL3 ESTAE - PGM IRPT CODE @MC6
ABOFFS DS AL4 ESTAE - ABEND OFFSET @MC6
ABPSW DS XL8 ESTAE - ABEND PSW @MC6
*
WACNTS DS 40XL(LACNTN) AREA FOR BUILDING ICHEACTNS
*
* RACF WORK AREA FOR RETURNED DATA
*
DS OF
RACWA DS 0XL256 ICHEGNT WORK AREA @MC1
ORG RACWA+28 PAST HEADER TO FIELD VALUE AREA
USRCNTL DC AL4(4) LEN OF USRCNT (4)
USRCNT DC AL4 NO. OF USERDATA OCCURRENCES
USRDLEN DS AL4 TOTAL LEN OF ALL USERDATA
*
FOLLOWING REPEATED ONCE PER OCCURRENCE
USRDOCC EQU * START OF OCCURRENCE
USRDOCL DS AL4 LEN OF THIS OCCURRENCE
USRMML DC AL4(8) LEN OF USRMN
USRNM DS CL8 USRMN FIELD
USRDATAL DS AL4 LEN USRDATA FOR THIS USRNM
USRTDA DS OCL256 USRDATA
USRFLGL DC AL4(1) LEN OF USRFLG
USRFLG DS X USRFLG
USRDOCLL EQU *.USRDOCC LEN OF OCCURRENCE (LESS USRDATA)
ORG WORKAREA+(32*1024) EXTEND RACWA UP TO 32K BOUNDARY @MC1
RACWAL EQU *.RACWA LEN OF RACF WORK AREA
WORKLEN EQU *.WORKAREA TOTAL LEN OF WORKAREA
*
PRINT NOGEN
IKJCPPL, IKJIOPL, IKJPPL,
PPLLEN EQU *.PPL IKJTSVL, IHAAPA, IHAASCBL,
Mick Covington
Systems Programmer (UK) © Xephon 2002
Inside IBM – IBM mainframe security since October 2000

This article follows on from the ‘Inside IBM’ that appeared in the last issue of *RACF Update* (issue 27, February 2002, pp 22-32). The article begins by reviewing subsequent updates to RACF and security enhancements to z/OS and z/VM. It ends with a look at recent news in other IBM, Tivoli, and Lotus security offerings.

**z900**

The new PCI Cryptographic Accelerator (PCICA) is a dedicated encryption processor, optimized for Secure Sockets Layer (SSL) protocol; it does nothing else. On the zSeries 900 (z900), each PCICA feature contains two cryptographic accelerator cards and can support up to 2100 SSL handshakes/second, but is limited by the CPU cycles available to perform the software portion of the SSL handshake. Current performance measurements with z/OS suggest that, on a 16-way z900, the maximum rate attainable is 3850 SSL handshakes per second.

The Integrated Cryptographic Services Facility (ICSF) and System SSL functions within z/OS Version 1.2 and above support the PCICA. z/OS HTTP server (and WebSphere), tn3270 server, LDAP server, CICS Transaction Gateway server, and other applications that use System SSL, as well as applications that call ICSF directly for clear key RSA encryption operations, will all transparently use the PCICA.

Likewise, the Resource Measurement Facility (RMF) feature of z/OS 1.2 is the first to report on the PCICA. But 1.2 does require PTF UW99368 for APAR OW49808.

Linux for zSeries supports the PCICA through PKCS#11 (Public Key Cryptography Standards) API support. The Integrated Facility for Linux will support the PCICA by mid-year.

There is also a promise to enhance OSA-Express (Open Systems Adapter) to support IPv6, but only for Queued Direct Input/Output (QDIO) mode.
z800

The z800 is a new smaller eserver zSeries below the z900 in capacity. The first model to be announced was part of the zSeries Offering for Linux, intended to run large numbers of Linux servers under z/VM. Subsequent models can also run z/OS, z/OS.e, OS/390 2.8 and above, z/VM, VM/ESA 2.4.0, VSE/ESA 2.4 and above, and Transaction Processing Facility (TPF) 4.1. z/OS and z/OS.e must be run in 64-bit mode. The z800 does not support any operating system running in 370 mode.

The Crypto coprocessor hardware is optional on the z800. It is a prerequisite for the PCICA and the PCI Cryptographic Coprocessor (PCICC) features. The PCICC is not available for the Linux-only z800 model.

z/OS.e

z/OS.e is a customized version of z/OS that runs only:

- On the z800 and non-IBM equivalents.
- New technology e-business transaction processing and data management workloads.

It is priced lower than z/OS, but follows the same release schedule. For example, both z/OS.e 1.3 and z/OS 1.3 were first available on 29 March. As such, the descriptions of security enhancements that follow, for z/OS 1.4, 1.5 and beyond, also apply to z/OS.e.

z/OS 1.4

The PKI (Public Key Infrastructure) Services component of SecureWay Security Server includes support for:

- 4758 Cryptographic Coprocessor generation of private keys.
- Sysplex enablement of PKI services.
- e-mail notification for completed certificate requests and expiration warnings.
- MAIL, STREET, and POSTAL CODE distinguished name qualifiers.
PKCS#7 certificate chains.
Even though IPv6 support has been added, all existing IPv4 functions still work, and applications not capable of IPv6 can continue to use IPv4 interfaces. IPv6 interfaces are implemented on the zSeries server with the OSA-Express adapter configured in QDIO mode for Fast Ethernet or Gigabit Ethernet networks.

tn3270 adds Transport Layer Security (TLS), while still providing SSL. ftp gets improved activity logging with a more consistent interface to security-related exit points, including the ability for the exit points to exchange data with each other. Distributed File Service (DFS) includes additional workstation domain-user-ID to z/OS-user-ID mapping options.

To make z/OS Unix more consistent with other platforms:

- An unused User or Group ID (UID or GID) value can be automatically assigned to a user or group.

- A system-wide setting prevents the assignment of a UID or GID value which is already in use; with the proper authorization, it is now possible to assign a shared UID/GID.

- The SEARCH command can now be used (by an administrator) to list the users/groups assigned to a UID/GID.

- The group owner of a new Hierarchical File System (HFS) file can now be automatically assigned using the effective GID of the creating process.

Sysplex-wide dynamic Virtual IP Addresses (VIPAs) for TCP/IP connections can now have the same single IP address appearance for application instances initiating outbound connections within a sysplex as Sysplex Distributor provides for inbound connections.

z/OS 1.5 AND BEYOND
Statements of Direction promise enhanced IPv6 support and new Enterprise Identity Mapping (EIM) services using Project eLiza technology. The goal of Project eLiza is to make all eservers, storage, and software, especially zSeries, a self-managing system, automating much of the system management function. The project was named
after the mid-1960s project to develop seamless communication between people and machines, perhaps best known for the program of the same name that played the role of an analyst by asking you questions based, in part, on your previous statements.

EIM will address the issue of multiple heterogeneous security registries existing in and between organizations. By managing the relationship between identities that are identified within multiple applications, platforms, and middleware, EIM services allow an application to use one registry for user authentication while using a different registry to associate users with resource access control rules.

CICS TS 2.1
CICSPlex System Manager began providing support for Enterprise Java Beans (EJB) technology in Version 2.1 of CICS Transaction Server for z/OS (CICS TS). After entering a user ID and password, CICS Web clients can register SSL client certificates to their ID in the RACF database. CICS TS can be set to allow only registered client certificates to be used on a connection, or to use HTTP basic authentication regardless of whether SSL is also used.

This EJB support includes EJB containers, which create and manage enterprise bean instances. Each container provides the services required by each enterprise bean running in it, including security. The enterprise bean does not need to authenticate users or check authorization rules. These functions are performed by the container on its behalf.

An EXEC CICS SIGNON or SIGNOFF command no longer modifies the user ID and security characteristics of the transaction issuing the command. The CICS Transaction Server for z/OS Migration Guide documents a temporary migration aid that restores the previous behaviour, which can cause unpredictable behaviour in a running transaction.

For many manuals, including the CICS RACF Security Guide, the only hardcopy available is by printing the Adobe Acrobat PDF file. Other IBM-recommended reading is the June 2000 redbook Securing Web Access to CICS (SG24-5756).
CICS TS 2.2

CICS TS Version 2.2 includes a full implementation of the EJB 1.1 security specification to provide method authorization checks through the isCallerInRole API. The API is used to determine whether a user is in a role that is authorized to execute a given method on an enterprise bean by accessing RACF through the System Authorization Facility (SAF) interface. APARs OW46859 and OW49190 must be applied to OS/390 or z/OS.

The need to understand EJB architecture is lessened by the fact that the method request executed by the enterprise bean runs under a CICS transaction ID and is associated with a standard CICS user ID, and can therefore be treated just like a procedural CICS transaction. A new Java-based CICS utility is provided for defining RACF profiles.

Version 2.2 also includes the Java 2 security policy mechanism, by which user-customized security policies can be used to control the new persistent, reusable Java Virtual Machine (JVM) within CICS TS. Users accessing CICS over Internet Inter-ORB Protocol (IIOP) are authenticated, providing secure interoperability between CICS and other IBM and non-IBM systems using the SSL client authentication protocol.

A new getCallerPrincipal method returns a Principal object whose getName method returns the distinguished name of the EJB client. When the client is authenticated with an SSL certificate, the distinguished name is extracted directly from that certificate; otherwise it is generated from a user-replaceable module, DFHEJDNX.

IMS V8.1

Version 8 of IMS includes enhancements to Database Recovery Control (DBRC). Recovery Control Dataset (RECON) Command Authorization Support allows users to control RECON access/update via DBRC batch commands or via the High Availability Large Database (HALDB) Partition Definition Utility. Security criteria can be customized and an audit trail maintained through a user exit.

MQSERIES GETS A NEW NAME

WebSphere MQ is the new name for MQSeries and is being phased in
gradually with each new release of an MQ product. WebSphere MQ has always implemented its own level of security beyond that provided by the many platforms it supports – for example RACF and the rest of SecureWay Security Server on z/OS. WebSphere MQ provides access control of queues and authorization identification between message queue managers.

New to Version 5.2 of MQSeries for OS/390 was the ability to qualify WebSphere MQ resource names in security profiles by a queue-sharing group name and/or a queue manager name. Version 2.1 of WebSphere MQ Integrator for z/OS added a Control Centre security exit.

First introduced just over a year ago, MQSeries Integrator Agent for CICS Transaction Server (MQSI Agent for CICS) was intended to replace Message Driven processor (MDp) from Early, Cloud & Company, integrating MQ with CICS and IMS applications. The runtime component of its MQSI Agent for CICS component runs as an application under CICS TS, using the security, auditability, and control facilities provided by CICS. Support by an External Security Manager for Front End Programming Interface (FEPI) pass tickets is also used.

MQSERIES EVERYPLACE
MQSeries Everyplace extends MQ to an ever-growing number of lightweight or mobile platforms and devices. Authentication, compression, and encryption are used to bring reliability and security to network connections that would otherwise be open to data errors and electronic eavesdropping.

MQSeries Everyplace provides message-level, queue-level, and end-to-end security. Up to 128-bit encryption is provided by MARS, DES, triple DES, RC4, and RC6. There is also Wireless Transport Layer Security (WTLS) standards compliance.

MQSERIES WORKFLOW 3.3
MQSeries Workflow databases are protected by DB2 security. MQ security provides access control to MQ Workflow queues. Users must
be authorized via Flow Definition Language (FDL) and MQ Workflow Buildtime to access MQ Workflow resources, such as processes, and to administer the system.

Auditing is provided via operating system security logs and MQ Workflow audit log. The Windows NT unified log-on option is supported for log-on.

User authentication is provided by means of an MQ Workflow user ID and password. Passwords are not transmitted over the line, at user log-on, or when users change their passwords. No clear-text password is stored in the MQ Workflow server databases.

WEBSPHERE

Version 4.0 of WebSphere Application Server for z/OS and OS/390 creates a secure Web deployment environment with Kerberos as the backbone and SSL at the endpoints. It provides automated authorization checking, and offers authentication and authorization service to clients, automatically checking the security credentials of all clients accessing WebSphere Application Server services. Both basic and certificate-based authentication are supported. As with previous versions, its security services are derived from the information provided by the hosting IBM HTTP Server for z/OS.

Version 4.0 of WebSphere Application Server, Advanced Edition for Linux, runs on zSeries mainframes. Both it and HTTP Server now support hardware crypto accelerators and smart cards to improve the performance of protected client/server and server/server communications. Smart cards allow users to carry their certificates with them.

Crypto hardware increases server throughput. It can also be used with the storage feature to store private keys in dedicated hardware while in use and encrypting them when idle. Private keys never leave the module unencrypted.

The Linux edition also offers improved Java security APIs in the distributed security model. Commands formerly provided through Tool command language (Tcl) scripts can now be performed using the new Java API.
It includes an interface for applications to interact with the WebSEAL component of Tivoli Policy Director. An upgrade of the LDAP client interface is also included for accessing directory services throughout the network.

IBM Bank Teller 4.0.2 implements the Interactive Financial eXchange (IFX) server infrastructure using the IFX Connector implemented by Version 4.1.1 of WebSphere Business Components (WSBC) Composer. IFX is an open Finance Industry standard specification for data formatting, connectivity, and security (SSL).

OTHER IBM SOFTWARE
Beyond what RACF provides for the Fault History File dataset, a new security subsystem within Version 2 of Fault Analyzer for z/OS and OS/390 provides finer access control of fault entry write and deletion.

Likewise, Version 7.1 of Content Manager OnDemand for z/OS and OS/390 provides more choices in defining security, including the ability to distribute security by department or groups of users with associated reports.

Even though Personal Communications, WebSphere Host On-Demand (HOD), and Screen Customizer are all now part of Version 2.0 of Host Access Client Package for Multiplatforms, each component retains its own version number. Version 5.5 of Personal Communications for Windows includes smart card support, allowing a certificate to be stored in a dedicated security device, such as a smart card.

In Version 7.2, DB2 Server for VSE & VM requesters can encrypt the password and the server can decrypt it. The CONNECT IDENTIFIED BY statement can now be issued over Distributed Relational Database Architecture (DRDA).

TIVOLI AND IBM SOFTWARE SUPPORTING IT
Tivoli security software is listed at:
It includes:
• Tivoli Policy Director
• Tivoli Policy Director for MQSeries
• Tivoli User Administration
• Tivoli Risk Manager
• Tivoli Identity Director
• Tivoli Privacy Manager
• Tivoli Security Manager
• Tivoli Global Sign-On
• Tivoli Public Key Infrastructure.

Of course, there is other Tivoli software with security features, as well as IBM software that has been built strictly to work with Tivoli security software.

Note that there have been a lot of product name changes as Tivoli continues to inherit IBM software products, and the SecureWay name has been dropped from all Tivoli products.

TIVOLI SECURITY SOFTWARE

Although Version 3.8 of Tivoli Policy Director neither runs on nor supports the mainframe, Policy Director Authorization Services for z/OS and OS/390 is free IBM software that provides an authorization daemon, pdacld, that extends Tivoli Policy Director to include z/OS. OS/390 2.10 is also supported. Both products provide centralized, policy-driven security authorization facilities. Previously, Version 3.7 had added cross-domain Web single sign-on, delegated user administration, authorization API entitlement service, and support for Lotus Domino registry, Java 2 security, and Windows 2000.

Version 3.8 of Tivoli Policy Director for MQSeries is the first version to include direct support for the mainframe, for both z/OS and OS/390 2.10, but requires the free Policy Director Authorization Services for z/OS and OS/390. It provides a single security management solution for MQ that covers MQ messages as they traverse across both mainframe and distributed servers. Previously, Version 3.7 had added
access control for MQ queues, and protection for data while in queue and on the wire.

Tivoli User Administration continues to run on both z/OS and OS/390, as well as a broad range of distributed platforms. It provides an automated, secure way to centrally manage user attributes and user services across multiple platforms, including centralized password management and a single view of user account data.

Version 3.8 includes a toolkit that customizes user records with additional fields. There are also additional application management capabilities demonstrated by sample code that manages Oracle database user account information.

Version 3.7 improved the performance, scalability, and password management tools. It was the first to support Tivoli Policy Director, Windows 2000, additional attributes in Windows NT, and group profile in Unix Tivoli Management Agent (TMA) endpoints, porting the capability to handle user groups in Unix to the scalable three-tier TMA architecture.

Version 3.8 of Tivoli Risk Manager and Version 1.1 of Tivoli Identity Director run only on AIX, NT 4.0, and Sun Solaris. Tivoli Privacy Manager also supports Windows 2000.

Tivoli Security Manager provides a role-based, centralized mechanism for managing and implementing access control policy from PCs to mainframes. Version 3.7.1 continues to support z/OS and OS/390 RACF as a client, adds OS/390 role template populate capabilities, and allows Windows 2000 resources to be included in the role-based access control model. The Unix security engine has been replaced with one based on Tivoli Policy Director.

Version 3.7 added Windows 2000 access control management to the role-based model. Role-Based Populate makes role-based access control easier, and has been expanded from NT and Unix to NetWare, OS/400, and OS/390. When Tivoli Security Manager and Tivoli User Administration are used together:

- It is now easier to combine user management and access control for Windows domains using separate User Account Domains and Resource Domains in a trust model.
• The Tivoli User Administration wppasswd command checks Tivoli Security Manager password policy to verify a password change request.

Tivoli Global Sign-On (GSO) supports a broad range of platforms, including 3270 mainframe applications. Version 3.7 adds Sun Solaris and Windows 2000 as clients. GSO requires:

• Tivoli Managed Framework
• Tivoli User Administration
• Tivoli Security Manager
• AIX, Sun Solaris, or NT 4.0 as a server platform.

Tivoli Public Key Infrastructure runs only on AIX.

SECURITY ENHANCEMENTS TO OTHER TIVOLI PRODUCTS

Version 5.1 of Tivoli NetView for z/OS can automatically log suppressed operator commands, submit TSO commands from NetView using SAF surrogate authority for TSO commands, be set to authorize a particular command only within a specified command list, and be used to specify which NetView operators have authority to log on to the NetView Management Console (NMC). The NMC Topology Server now has a customizable XML log that provides a record of console and server activity, including commands to be executed, command responses, and NETCONV communication start and stop. The Web interface has been completely redesigned to authenticate the operator’s NetView user ID and password, and provide authorization for specific functions through standard NetView-based security.

Tivoli Workload Scheduler (TWS) is the new name for Tivoli Operations Planning and Control (OPC) on the mainframe and Maestro on other platforms. With Version 8.1, TWS for z/OS inherits scheduling agent technology from TWS, and the non-mainframe implementations of TWS have been made more OPC-like. All implementations get a new Java-based GUI known as the Tivoli Job Scheduling Console, and changes made to the TWS database or plan are now logged to a log file for audit purposes. But most of the security and auditability capabilities are derived from the operating system and the requisite Tivoli Management Framework (TMF).
Tivoli Business Systems Manager (TBSM) monitors availability and performance of z/OS, OS/390, Windows NT/2000, AIX, HP-UX, OS/400, and Sun Solaris systems. Rather than rely on the Windows registry, Version 1.5 does its own authentication to validate log-ons and passwords. Users log on to the client specifying their domain-qualified Windows log-in and password. This information is encrypted and sent to the Application Server, which attempts to authenticate the user and return an indication of either a log-in failure, such as an expired password, or the user’s authorization to the client.

The new TBSM Java-based application server can be set up with Windows groups for TBSM operators and administrators. Users can then log in to their groups and perform TBSM functions.

Tivoli Data Exchange is a bulk data transfer product that operates using protocols supported by MQ, such as SNA and TCP/IP. It supports servers running z/OS, OS/390, TPF, OS/400, Windows NT/2000, AIX, HP-UX, Sun Solaris Versions 2.6 and 2.7, and OS/2, and clients running Windows 9x/2000.

Version 1.2 allows status messages to be offloaded directly to a relational database through a new exit that can be used to extract status messages from the status queues and place them into a database, enterprise console, or custom application. It could, for example, be used to create a real-time-accessible security audit log of all data transfers.

Despite the fact that Version 3.1 was announced in June 2001, Version 2.2 of Tivoli Manager for Domino remains the only release to support OS/390, and is the only platform where Version 2.2 support does not end on 29 June 2002. New to Version 2.2 is monitoring of Lotus Notes Access Control Lists and unsuccessful log-in attempts. Tivoli Management Solution for Domino Version 3.2, a newly architected bundle that includes Tivoli Manager for Domino 3.1, does not support z/OS or OS/390.

Tivoli NetView Performance Monitor (NPM) provides four levels of security:

- Minimal – NPM checks the operator ID to see whether it is identified to NPM and not already logged on.
• Normal – NPM checks operator ID, password, and profile; the profile can limit what the operator is allowed to do.

• RACF – NPM checks the profile and RACF is called by use of the SAF interface to check operator ID and password.

• User – an NPM user exit routine provides special ID and password validation, either directly in the routine or by a call to a non-IBM security product.

Not to be confused with NPM, Tivoli NetView Performance Monitor for TCP/IP (NPM/IP) is a separate product at a different version level. Version 1.2 added RACF support to centralize access rights. This was done through a new SAF interface in Version 3.1 of CLEVER TCP/IP.

LOTUS

The concept behind Lotus Domino for IBM z/OS and OS/390 is to replace large numbers of NT or Unix servers running Domino with a single mainframe Domino server. Reduced Total Cost of Ownership (TCO) is the main selling point, although scalability, availability, and performance are better, too. Several associated products enhance security.

Lotus Domino for IBM HTTP Server, sometimes referred to as Web Connector for the IBM OS/390 Web Server, stores X.509 digital certificates in RACF and other security products. IBM Document Connect for Lotus Domino for z/OS allows administrators to define data integrity and security of the building blocks, through protected text blocks and Lotus Notes hierarchical access control lists.

Symantec acquired IBM’s anti-virus business in the late 1990s. Because all operations of Symantec AntiVirus 2.5 for Lotus Notes/Domino are completed in native Notes format, it also runs on the mainframe.

CONCLUSIONS

And, of course, there is Linux, where IBM has invested heavily to tune performance and overcome scepticism about reliability and security.
The concept is to realize the economies of scale possible by consolidating large numbers of Intel-based Linux servers on to a single zSeries mainframe.

And finally, IBM’s main security page is at http://www.ibm.com/security. As well as providing recent IBM security-related announcements, the left sidebar makes a good starting point for access to additional IBM security information.

Jon E Pearkins
(Canada)

© Xephon 2002

---

**Contributing to RACF Update**

Although the articles published in Xephon *Updates* are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others.

Many have discovered that writing an article is not the daunting task that it might appear to be at first glance. Often, just a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties with RACF, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon’s wide range of products and services, simply by telling us all about it.

More information about contributing an article to a Xephon *Update*, and an explanation of the terms and conditions under which we publish articles, can be found at http://www.xephon.com/nfc. Alternatively, please write to the editor, Fiona Hewitt, at any of the addresses shown on page 2, or e-mail her at fionah@xephon.com
RACF restructuring: coding

The second article in our four-part series on RACF restructuring concentrates on coding. For part one of this series, see RACF Update 27, February 2002, pp 8-22.

WHAT WE’RE DOING TODAY...

This article reviews a new, and hopefully more logical, group structure, and also offers advice on user ID and CICS resource naming conventions. It segregates groups into two main structures: organizational (based on your company’s org chart), and system (which segregates RACF and OS/390 functions into a hierarchy). This structure can be much more efficient and effective in controlling system security.

There are also hints and tips on generating the JCL by using MS Word and MS Access. It’s a neat trick that can save you several hours of repetitive and mindless keyboard work, and cuts down on the errors that invariably crop up during coding.

For the more advanced or adventurous, we also examine the rebuilding of RACF ISPF input and display screens into something a bit more formatted and functional.

There are hints and tips on how best to create your test LPAR and the general size requirements, and advice on the order of building the database from the JCL you’ve created, and on how to do the initial database configuration.

Finally, we look at some of the Pentland Utilities, which can help you develop new structures by recording and reviewing your current database (see http://www.nigelpentland.co.uk).

GROUP STRUCTURES

Groups are the mainstay of the overall functioning of RACF. User access, dataset access, virtually the entire structure of RACF is based on groups and their interconnections. So that’s where we’ll start to develop our new database. And we start from line 10.
When a new RACF database is created, there are only three groups defined by default: SYS1, the highest group, and VSAMDSET and SYSCTLG, which are owned by SYS1 (see Figure 1).

Under no circumstances should you change these names. I’ll admit, I’d prefer that SYS1 could be renamed to something else, so that it could be used simply as a High Level Qualifier (HLQ) group for datasets. However, that’s not really feasible, because many RACF functions point directly at the SYS1 group. Changing that would require far too much re-coding of RACF, and would make patches and updates of the software extremely difficult to implement without problems. So we keep SYS1 as the primary group.

However, we want to segregate two major functions within the group structure: business and RACF/operating system functions. To do this, you simply create two new groups, named BUSINESS and SYSTEMS. Here’s a sample of the JCL you need to create such a structure:

```
ADDGROUP BUSINESS -
DATA('HIGHLVL GROUP FOR BUSINESS OPERATION PROFILE -
DEFINED SYSTEM GROUP OWNER - SYS1-
******************************************************************************
AUTH1:MICKEY MOUSE 555-1212 X2000-
AUTH2:DONALD DUCK 555-1212 X2015-
ADDED BY DOC FARMER 15/05/2002') -
OWNER(SYS1) -
SUPGROUP(SYS1)-
OMVS(GID(10001000))
ADDGROUP BUSINESS -
DATA('HIGHLVL GROUP FOR SYSTEMS OPERATION PROFILE -
DEFINED SYSTEM GROUP OWNER - SYS1-
******************************************************************************
```

Figure 1: Three default groups
You’ll note that the installation data is set up in a particular way: 45 characters per line for five lines, and a maximum of 30 characters for the sixth line. This is to make the output more readable on RACF screens and reports. By including owner information as well as an adequate description of the function and use of the group, you make it more understandable to RACF Admin and Analyst alike. This is the installation data that should be used for all groups, and is structured as follows:

- **Line 1.** Brief description of the job, function, or group, followed by the department name.

- **Line 2.** Description of the group, followed by all of the group identifiers preceding it (from highest to lowest).

- **Line 3.** Further description area. If none is needed, fill with asterisks.

- **Line 4.** Name and phone number of primary authorizer (the owner of the group).

- **Line 5.** Name and phone number of secondary authorizer.

- **Line 6.** Date, time, and name of person who added or last updated the group profile.

For subsequent levels, we create a specific naming convention for groups in each category (BUSINESS and SYSTEMS). BUSINESS groups are structured as follows:

AAA$BBBn

- Where AAA is the primary group identifier.
- $ is the BUSINESS identifier.
- BBB is the secondary group identifier.
- n is the number 1 through 9, describing an individual group function within the primary/secondary combination.
The highest-level qualifiers must conform to the structure under BUSINESS on the company’s organization chart:

- Administration Group (ADG)
- Investment Group (INV)
- Corporate Services Group (CSG).

For example, the Administration Group is a primary group, and uses ADG as the identifier. For the highest level, the Group Name would be ADG$. Under that group are six divisions or departments, as defined on the company’s organization chart:

- Credit Division (CRD)
- Finance Division (FCD)
- Human Resources (HRD)
- Internal Audit (IAD)
- Planning (PLN)
- Risk Management (RSK).

So you would have ADG$CRD, ADG$FCD, ADG$HRD, ADG$IAD, ADG$PLN, and ADG$RSK. Now, if you have (for example) two departments in Internal Audit – Financial Auditing and Technical Auditing – you would come up with separate three-letter identifiers for each: IAF and IAT. Their group names would then be IAD$IAF and IAD$IAT.

Each department/division/function on the organization chart should have its own unique three-letter identifier. This helps in tracking the function-to-group within the system. It also creates a naming convention, which lays out your company’s organization chart within RACF itself. This structure is also quite useful in other systems (LDAPs or Novell naming conventions for access, for example). It can even help in the development of an overall system access database.

Figure 2 illustrates the high-level structure of a sample BUSINESS profile.

Now, what about the SYSTEMS structure? Well, I’ve come up with
Figure 2: High-level structure of a sample BUSINESS profile
a high-level division structure for those functions required by RACF and OS/390. They work out as follows:

AAA#BBBn

- Where AAA is the primary group identifier.
- # is the SYSTEMS identifier.
- BBB is the secondary group identifier.
- n is the number 1 through 9, describing an individual group function within the primary/secondary combination.

You’ll note that in all BUSINESS profiles the identifier is $, while all SYSTEMS profiles use a #. This is mainly for immediate identification of what the group profile represents. It would be virtually impossible to remember hundreds or even thousands of group names. This naming convention gives the viewer instant knowledge of which part of the RACF database they’re dealing with.

The highest-level qualifiers must conform to the structure under SYSTEMS within an OS/390-RACF system:

- Dataset Name Profiles (DSN)
- General Resource Profiles (GEN)
- User Profiles (USR).

Below each of the HLQs are a number of system or RACF functional groups, as illustrated in Figure 3.

Now, you may be asking yourself why you should come up with unique three-character identifiers for each department or system prefix. Because I SAID SO, THAT’S WHY (oops, sorry, just slipped into parent mode there). Actually, it’s because you can quickly identify any system, subsystem, division, department, etc, just with that code. Overlapping the system prefixes (owner, then area) automatically provides you with a kind of visual tree structure. That tree structure can be listed in alpha order by three-character identifier so that you can instantly determine the owner(s) of the group. And three characters give you a lot of leeway. Granted, the first character
Figure 3: System or RACF functional groups
must be alpha (a throwback to the old HASP days – remember those?), but, even with that, you have a potential 33,696 combinations! That offers you a lot of flexibility.

USER ID STRUCTURES

As with group IDs, you need to have proper user ID naming conventions for easier identification and greater control. User IDs for actual human users are relatively straightforward. For CICS-only users, I prefer to use a full eight-character ID (which can be all numeric), to differentiate them from TSO/CICS users, who are restricted to only seven characters, the first of which must be an alpha character. (Note to IBM: HASP is dead. Deal with it and give the OS/390 and RACF community more flexibility in naming conventions, please.)

In the company where I work, the employee ID is seven digits (the last one being a check digit). For CICS-only users, this is quite straightforward: a one-character prefix, which can identify them as permanent, temporary, consultant, etc, followed by the employee ID. For TSO it gets a bit mucky, but there’s a solution: replace the first number with a corresponding letter – so, 1=A, 2=B, 3=C, and so forth.

Now, what if you don’t have a seven-digit employee ID? Well, try the last seven digits of an employee’s Social Security Number. These also incorporate a check digit (nifty, eh?) and can be used without fear of duplication.

When creating new user IDs, you should remember to use the installation data to ‘fill in the gaps’. After all, the name field in RACF uses only 20 characters. That’s fine if your name is John Smith, but doesn’t work so well if your name is Abdullah Muhammed bin AbdulAziz Al-Harbi Al-Hassan (there are people in the world whose names run to 50 characters). This can be dealt with in two ways. First, have IBM expand the name field. Two chances there: Slim and None, and Slim just left town (thank you, Dan Rather). Second, use the first line of the installation data to include the full name. Here’s an example of how the JCL might appear:

ADDUSER A234567 -
You’ll note that the format of the installation data for user IDs is different from that for groups. For these, the first five lines are 49 characters in length, and the last line is five characters. This gives you more space for data, such as the people who are the authorizers of that employee. It also gives you space to include the full name, which can be quite helpful in doing a search for a partial name in a flat file or a Pentland Utility report.

But what about system-required IDs like started tasks, CICS regions, etc? A separate, yet consistent, naming convention should be used for each. Now, you may get some resistance from your Technical Support or Production Control people. After all, they’ll have to live with the new naming conventions on a 24/7 basis. Also, they’ll have to make some significant changes to JCL and internal tables to reflect the new names. Be patient, allow them input in developing the naming conventions, but don’t let them run the show. Instead, show them how the new naming conventions will make problem tracking faster and easier. And bring doughnuts.

Let’s start with a naming convention for CICS regions. The general format should look like this:

CIC#xaaa

- Where CIC is the CICS Identifier (constant)
• # is the SYSTEMS identifier (constant)
• x is the single letter prefix corresponding to the type of region:
  – B = Business
  – D = Development
  – I = Integration
  – P = Production
  – Q = Quality Control
  – T = Training
  – U = User Test
  – X = IS Security Test
• aaa is the application prefix corresponding to the type of region (this one is determined jointly by Technical Support and Production Control).

For started tasks, it’s a bit simpler:

STJ#aaaa
• Where STJ is the Started Task Job Identifier (constant)
• # is the SYSTEMS identifier (constant)
• aaaa is the started task prefix corresponding to the type of region (this one is determined jointly by Technical Support and Production Control).

For either of these, the installation data field is important in that it can provide vital descriptive information as well as ownership. Here’s what the installation data field should look like for a CICS region:

```sql
DATA ('PRODUCTION INVESTMENT TRACKING SYSTEM CICS-
DESCRIPTION OF REGION FUNCTION GOES HERE -
DESCRIPTION OF REGION FUNCTION GOES HERE -
OWNER1:DOC FARMER B345678-
OWNER2:MOHAMMED ALI AL-MOGBIL C456789-
!!!!!')
```

Note the five exclamation marks at the end of the description. This separates it from normal user IDs, and can be a great search tool in a flat file.

Now let’s take a look at the installation data field for a started task:

```
DATA('TSO - TIME SHARING OPTION - IBM PRODUCT       STJ-
DESCRIPTION OF STARTED TASK FUNCTION GOES HERE      -
DESCRIPTION OF STARTED TASK FUNCTION GOES HERE      -
OWNER1: DOC FARMER    B345678-
OWNER2: MOHAMMED ALI AL-MOGBIL C456789-
&&&&&')
```

The five ampersands at the end flag this as a started task, which as above can be used in flat file searches. You’ll also note that on all installation data fields, we’ve included an owner or authorizer. This is quite important when determining who is allowed to grant or deny access to a function or user, and should be used on all installation data fields (user, group, dataset, general resource profiles, etc).

CHEATING FOR FUN AND PROFIT

Okay, now that you’ve got your entire group structure figured out, with all the hierarchies and ownership issues, and ensuring that your three-character codes and OMVS GIDs and UIDs are all unique, you’ve got to code the JCL. At this point, most people invoke a personal deity in a colourful fashion, because they know that they’re in for a lot of typing. For example, let’s assume that your new group structure will require 1,000 groups. That’s daunting enough, until you realize that you will need to write ten lines of code for each and every group. That’s 10,000 lines of JCL to type manually. And you’ve got to make sure the RACF commands are consistent, and that installation data fields are aligned properly, and that you’ve got all the owner names and extensions correct, etc, etc. That is, to put it mildly, a pain.

But, instead of dealing with a few hundred thousand keystrokes, you can cut down your input by creating a database within MS Access for input of the raw data into a mail merge document in Microsoft Word. That mail merge document would contain the JCL template, which would then insert the database materials into formatted, structured JCL ready for processing on the mainframe.
Creating the database in Access is pretty straightforward. You can even create structured input screens so that you can farm out the work to administrative or secretarial staff. Provided you give them accurate guidelines, and you place size limits on the number of characters per field, you can get quite good results. One note: you should remember to pad your name and data fields with blank spaces to the right of the data, so that they always come out on the database with the proper number of characters. This is especially important when formatting the installation data fields in MS Word (through the mail merge process). If you neglect to do this, you’ll end up with a rather jumbled looking display when you do a List Group or List User once you’ve run the JCL.

Another important note: when you do your group inputs, BE SURE TO DO THEM IN ORDER! It is VITAL that these are processed in highest to lowest order. If you try to define a group, and you haven’t defined the OWNER/SUPGROUP for it yet, you’ll get a rejected item. My advice here is to input the groups into the database using the business and/or system organization charts as a guide. As long as you don’t re-sort the database later, you’ll be in good shape. An alternative to that is to create several smaller databases: one for highest-level groups, one for secondary groups, one for tertiary groups, etc. You may have a few more mail merges and JCL runs, but in the long run it’s a bit safer.

One more advantage to the database/mail merge option of JCL creation is that you now have a separate database from which to review access and authorization names. Since RACF’s reporting capabilities are less than ideal (as has been discussed ad nauseam), this can give you the opportunity to keep track of who the authorizers are for departments, groups, etc. Also, if you have a change of authorizers for a department or system function, you can update the database and generate new JCL to revise the installation data fields for any affected IDs, thereby keeping the RACF database up to date. If you take this route, be sure to keep accurate records of what new IDs, groups, etc, have been added by your Administration staff in order to keep your Access database in sync.
ISPF SCREENS – THE CASE FOR NEW ONES

Let’s face it: IBM doesn’t appear to have put a whole lot of thought or planning into its ISPF screens for RACF. Installation data input is free-format, so you’ve got to count how far along you are if you want to have the List User, List Group, List Dataset, etc looking correct. And IBM, while we’re at it, what possessed you to change the installation data format for all of these items? I mean, 45 characters per line for groups, 50 for users, 79 for datasets and general resource profiles, etc—a little standardization would’ve been more convenient.

But be that as it may, we’re stuck with IBM’s ISPF screens. Or are we? If your Technical Support team is of sufficient quality (and I’m betting that it is) and of sufficient intestinal fortitude, new ISPF screens can be designed which will allow for better, more logical, and more efficient input of RACF data by your administrative team. This can keep your installation data formats up to date without having to resort to JCL every time you want to add or change a user, group, or dataset profile.

I’m not going to go into the specific coding requirements of the ISPF screens here for a variety of reasons. First and foremost, I’m not a programmer capable of creating such code. Second, when it comes right down to it, each site must come up with the screens which will fit their desired use of the installation data field—formatting and content issues are quite individual and must conform to your own security requirements. Finally, Technical Support people love a good challenge. Developing new ISPF screens for RACF, with sufficient controls to ensure security of data and continuity of naming standards, will give them quite a lot of enjoyment.

“HERE’S YOUR LPAR – WHERE ARE THE DOUGHNUTS?”

Well, you’ve got the JCL all neat and tidy in your text files on your PC. You’ve developed all your group structures, recreated your CICS region profiles, copied the dataset profiles, made all new user IDs, all tucked away in ASCII character sets on your hard drive. You’ve spent a lot of time and effort getting this ready. But it’s not doing you much good in a Windows environment. You need a test LPAR to do the initial builds and tests.
If you remember from the previous article, I stated that this process should be run as a project, inclusive of many different operational and business areas. One of the most critical ones this phase are your Systems Programmers, or Technical Support, or ‘techies’ in the vernacular. These guys and gals are the ones who will create the actual test environment. But what should you be asking for? This depends on your processor environment (and processor capacity).

Generally, your first ‘build’ LPAR should be pretty small. Don’t ask for more than 10% of processing capacity, and limit yourselves to two disk packs. You can get away with one, but two will allow you to do some initial testing on a ‘system’ and ‘non-system’ basis.

WARNING! Be ABSOLUTELY sure that your Tech Support staff understand that you want a new LPAR, not a copy of an existing one. In other words, they should start from scratch. This will prevent them from copying over an existing RACF database. You don’t want one of those at all. You want the new LPAR to start up with a blank security file. OS/390 and RACF will automatically default to the security structure noted in Figure 1.

It’s also a good idea (in fact, it’s vital) that you read the RACF installation manual (GC28-1920-00) and follow the actions specified. You’ll want to create a TSO user ID (after you’ve logged in as IBMUSER, of course), which gives you SPECIAL, OPERATIONS, and AUDITOR access. You’ll need those for the initial set-up work you’ll be doing. Make sure your Internal Auditors have passed on this, and explain to them that this is temporary access for the build only. Otherwise, they’ll say (quite rightly) that this is too much functionality for an individual to have.

Note that this is definitely a point in time where bribery is vital to the success of your RACF database restructuring project. A local doughnut emporium is an absolute necessity (if there isn’t one within walking distance, find one that delivers). Get a wide variety, don’t skimp on the jam-filled, and get extra napkins to keep the powdered sugar off the CPU.

Once you’ve confirmed that you can log on under the new ID, start feeding in the JCL from your PC in the usual manner. Then you’ll
begin the actual build process. But what order should you follow for this? Well, I’ve found that the following structure seems to work:

1. Groups (highest to lowest order)
2. User IDs
3. Datasets

You might be able to swap 3 and 4, but groups MUST be installed first. The rest of the RACF structure hangs on those groups, and you’ll get error upon error if you try to install user or dataset or GenRes profiles before their creation. Also, make sure that you review the output from all your JCL, in order to find any items that “didn't make it” in the process. Go back to correct those errors (in your main JCL file) and then make the correction via your TSO interface. Also make sure you

---

**Figure 4: Pentland Utilities tests**

<table>
<thead>
<tr>
<th>RACF00</th>
<th>RACF52 SYS1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACF01</td>
<td>RACF56</td>
</tr>
<tr>
<td>RACF02</td>
<td>RACF58 ALL</td>
</tr>
<tr>
<td>RACF03</td>
<td>RACF59</td>
</tr>
<tr>
<td>RACF04</td>
<td>RACF64</td>
</tr>
<tr>
<td>RACF05</td>
<td>RACF65 GCICSTRN</td>
</tr>
<tr>
<td>RACF07 SYS</td>
<td>RACF68 DSM01.txt</td>
</tr>
<tr>
<td>RACF12 GCICSTRN D</td>
<td>RACF69</td>
</tr>
<tr>
<td>RACF18</td>
<td>RACF72</td>
</tr>
<tr>
<td>RACF19 TCICSTRN GCICSTRN</td>
<td>RACF77</td>
</tr>
<tr>
<td>RACF20 TCICSTRN GCICSTRN D</td>
<td>RACF79</td>
</tr>
<tr>
<td>RACF22 TCICSTRN</td>
<td>RACF80</td>
</tr>
<tr>
<td>RACF24</td>
<td>RACF85 TCICSTRN GCICSTRN</td>
</tr>
<tr>
<td>RACF25 TCICSTRN GCICSTRN D</td>
<td>RACF86</td>
</tr>
<tr>
<td>RACF28 TCICSTRN GCICSTRN C</td>
<td>RACF87</td>
</tr>
<tr>
<td>RACF30</td>
<td>RACF88</td>
</tr>
<tr>
<td>RACF32</td>
<td>RACF89</td>
</tr>
<tr>
<td>RACF37</td>
<td>RACF90</td>
</tr>
<tr>
<td>RACF38</td>
<td>RACF91 TCICSTRN GCICSTRN</td>
</tr>
<tr>
<td>RACF42</td>
<td>RACF92</td>
</tr>
<tr>
<td>RACF48 TCICSTRN GCICSTRN</td>
<td>RACF33 TCICSTRN GCICSTRN</td>
</tr>
<tr>
<td>RACF49 TCICSTRN GCICSTRN</td>
<td>RACF34 TCICSTRN GCICSTRN</td>
</tr>
</tbody>
</table>
note ALL the changes you made. This will be vitally important when you start the testing phase (in the next article in this series).

Once you’ve (re)created the profiles, create a flat file using IRRDBU00 and run a DSMON report (with output in both paper and flat file). If you wish, you can download the flat files and do some Pentland Utilities tests at this stage (see http://www.nigelpentland.co.uk). The tests you should perform (in order) are shown in Figure 4.

You should also take the paper copy of the DSMON report to your Tech Support staff, to verify the started task list and to check over the tree structure. If you’re so inclined, you may want to run an IRRUT200 (Index and Map) to check the database structure in detail. Just be sure you have no SYSUT1, because you’re not making a copy of the file itself. You just want the analysis. Go over this with your techies as well, and ensure that the structure seems to be intact.

IN OUR NEXT EXCITING EPISODE

You’ll laugh, you’ll cry, you’ll kiss your career goodbye when you...

- Create a full-sized test LPAR!
- Develop detailed test scripts and schedules!!
- Build a RACF database – over and over and over!
- Generate progress and technical review reports for management!

Okay, seriously.

The next article will show you how to set up the testing environment from the mini-LPAR you created, how to transfer the database into a separate development or test LPAR, and how to safeguard those testing regions from damage. It will describe who should be involved in the testing, processes you should follow, and the general length and breadth of the tests. It will also give some insight into your handover process from mere testing to a live environment process.

(Doc Farmer would welcome comments and suggestions on this article. He can be contacted at Doc.Farmer@sbm.net.sa.)

Doc Farmer
Manager and Senior IS Security Analyst (Middle East) © Xephon 2002
Converting from ACF2 to RACF

Following on from the article in RACF Update 26 (November 2001, pp 32-35) which presented a direct comparison of RACF and ACF2, this article looks in detail at an ACF2 to RACF conversion. Note that although this article will obviously be of interest to anyone considering moving from ACF2 to RACF (particularly RACF sites that have inherited ACF2 systems through mergers or acquisitions), some of the material will also be useful to those converting from TOP-SECRET to RACF.

This article is based on an ACF2 to RACF security software conversion that took place in Toronto, Canada. The conversion, including planning, preparation, translation of all the security information, testing, and implementation took about five months.

WHO WILL BE AFFECTED?

In most installations, security is generally transparent to users, and the conversion to RACF will therefore make little or no difference to the vast majority of the user community. However, it will affect various people involved with mainframe security. In particular, the security administrators, technical support staff, and IS internal auditors will find that working with RACF is quite different from working with ACF2. For these groups, converting to RACF will mean a major change in the way they do things.

SOME MAJOR DIFFERENCES

The way security is implemented in ACF2 is quite different from the way it is implemented in RACF. Here are some major differences:

- The ACF2 term for data protection is called a ‘rule’; its RACF equivalent is called a ‘profile’.
- ACF2 rules are compiled; RACF profiles are not.
- There are three separate databases in ACF2 for storing its security information; RACF keeps everything in one database.
- ACF2 ‘source’ rules can be kept and viewed in a partitioned
dataset (PDS). RACF has no such concept. However, RACF definitions can be kept and viewed in a ‘flat’ file.

- There is no concept of groups in ACF2, but it uses the concept of UID (user identification) strings to define a user’s ‘grouping’; RACF uses the concept of groups, and users are connected (or belong) to groups.

- For non-dataset resources (such as disk volumes) ACF2 uses ‘resource types’; RACF has resource classes for this purpose.

- ACF2 does not allow a user to belong to more than one group. In RACF, all users are connected to at least one group; they can optionally be ‘connected to’, or associated with, additional groups.

There are other differences as well. ACF2 keeps all of a user’s privileges (that is, what the user is allowed to do) with the user’s profile definition. Although RACF also does the same for some of its privileges, for others it has separate ‘resource classes’. Information such as who has the privilege is contained in a resource class. An example of this is the ‘Bypass Tape Label Processing’ (BLP) privilege. ACF2 stores this as a ‘flag’ in the user profile; RACF has a profile that lists all users who have the BLP privilege. (See RACF Update 26, November 2001, pp 32-35 for a more detailed comparison of RACF and ACF2).

THE CONVERSION TEAM

The first step in our conversion process was to form a RACF conversion team. The core team members worked very closely together, and met frequently, both formally and informally. They included:

- The MVS System Programmer (20% time commitment to this project). This individual would create the MVS test image for testing, install the RACF software, and work on all system-software related tasks.

- The Security Administrator (70% time commitment to project work). The individual knew the current security set-up very well, and was instrumental in guiding us in choosing various RACF options.
The RACF conversion specialist (100% time commitment to project). This person had worked on all aspects of ACF2 and RACF, including installation, planning, security administration, and reporting.

Other team members participated to a lesser extent – the Database Administrator, the Internal Auditor, and the Manager of Technical Support.

SCOPE
The first task of the conversion team was to scope the project. The installation had about 700 userids and about 3,000 ACF2 rules. The security administration was centralized, so there was less RACF training required. There were no ACF2 exits in place; only standard functions were being used. (If there are ACF2 exits in place, they would need to be reviewed. Are they still necessary? Will RACF provide an equivalent function without an exit? In some cases, exits may have to be re-written to fit RACF.)

We prepared a project plan, and identified all the activities. We reviewed the security reporting that was in place – violations monitoring, invalid log-ons, use of special privileges, and so on. These reports were replicated for a RACF environment.

Since the two security products cannot co-exist on the same MVS system, the best way to convert was to build the RACF database on a test MVS system, and do the testing off-hours. Of course, this meant we would need to do a lot of weekend testing.

CLEAN-UP OF RULES AND USERIDS
All installations have security rules that were useful at one time, but are now obsolete. The same goes for userids. A security conversion project is a good time to re-visit the security set-up and do some clean-up. In fact, even those installations who are thinking of converting to RACF at a future date should spend the time now to do as much clean-up as possible, and get this chore out of the way.
We spent a fair bit of time on this activity, but the effort was well worth it. We not only deleted obsolete rules and userids, but also simplified some rules – without compromising security, of course. Cleaning up things meant there was less security information to carry forward to RACF.

RACF GROUP STRUCTURE

Next, we started looking at ways to carry out the conversion exercise itself and build the RACF database from scratch. In simple terms, the RACF database contains mostly ‘profiles’:

- User profiles and group profiles tell RACF ‘who’ needs security access.
- Dataset profiles and resource profiles define ‘what’ needs to be protected.

The first thing we did was to define the RACF group structure. This is a tree-like chart that defines a hierarchy of user groups and closely resembles the functional units and departments in an organization.

In ACF2, a portion of the UID string is generally used to store information on departments and divisions of the organization. The ACF2 UID strings provided us with a starting point, showing us how users were ‘grouped’ several years ago, based on information current at that time. In some cases, this grouping did not reflect the current organizational structure. This can happen when, for example, two departments have merged, but the security administration staff did not have the time to reflect this in the security database.

Again, a security conversion project is a good time to improve on the corporation’s organization chart, in security terms. We built the RACF group structure so that it more closely resembled the current organization.

A lot of planning and preparation went into defining the group structure. It’s an important conversion activity and lays the foundation for all future administration of RACF within the installation. It’s easy to change userids and profiles later, but very difficult to redefine groups afterwards.
BUILDING THE RACF DATABASE

Before starting to build the RACF database, we had to impose a ‘freeze’ on changes to the ACF2 database. However, since this was a production database, a total freeze was impossible and emergency changes to ACF2 were allowed. There was a ‘cut-over’ point to security changes, after which we itemized all changes for later transfer to RACF.

Once the RACF group structure was finalized (on paper), we proceeded to define to RACF the various profiles – groups, userids, dataset, and resource – in that order. This order is very important – without groups, you cannot build userids, because userids require you to specify the default group for the user. And without userids and groups, you cannot define profiles – the access lists in profiles require userid and group information.

We made a lot of use of the EDIT function of ISPF/TSO to define the RACF profiles. However, although ISPF/TSO edit functions (and CLISTs) are sufficient for smaller installations, larger installations may prefer to use programming languages such as SAS to build RACF profiles from the ACF2 database. The benefit of this method is that you make fewer typographical errors.

The idea is to dump all ACF2 information into ‘flat’ files, and use this as input, to come up with RACF commands in a flat file (output). These RACF commands can then be executed in batch to build the RACF database.

To define the group profiles, we used listings containing all unique UID strings, and then used the grouping information provided therein. This, together with the group structure described above, provided the material to build a list of ADDGROUP commands for RACF. These commands were processed in batch, on the test machine that had the RACF database.

Similarly, to define all user profiles, we obtained a list of userids in ACF2, using one of the ACF2 reporting utilities. We then edited this list to derive a list of ADDUSER commands for input to RACF, in batch.
The biggest challenge was to translate ACF2 rules into RACF dataset and resource profiles. Again, we produced a flat file of all the ACF2 rules defining access to datasets and resources. This process is fairly simple in ACF2 using the ACF2 DECOMPILE command. The result goes to a Partitioned Data Set (PDS), but ISPF/TSO can be used to copy this PDS to a sequential or ‘flat’ file. The translation process itself is fairly involved, since the way ACF2 processes and interprets security access rules is quite different from the way it is done in RACF.

The following example illustrates the translation process from ACF2 rules to RACF commands.

Suppose that all programmers in DEPT1 are allowed full access to the DEPT1 test datasets. Further, assume that such datasets all begin with DEPT1.TEST. The ACF2 rule for this would look, in part, like this:

```
```

We edited and ‘massaged’ the above data using ISPF/TSO edit to come up with its RACF equivalent:

```
```

There were thousands of such translations to be done, so ‘massaging’ all of them at once, using edit commands, really helped. Again, SAS may be better suited for larger installations.

This approach also meant that we had to use RACF CLISTs (Command Lists) and commands to build our initial RACF database, and not the RACF ISPF panels. Of course, once the database is built and you go live with RACF, it’s a matter of preference whether to use commands or RACF panels.

We were keeping track of all the security changes that occurred since the date we started the translation process. These changes were carried forward to RACF just before going ‘live’.

RACF GLOBAL OPTIONS

Another important conversion activity is specifying RACF ‘global options’. These are high-level parameters that determine how RACF will function at the installation in overall or ‘global’ terms. They are
used to enforce password controls, specify auditing options, activate, and deactivate resource protection, and so on.

We didn’t worry about RACF global options at the very beginning of our project. By and large, we made do with defaults supplied with RACF. Only later, when the time came to begin the testing, did we start customizing the global options.

TESTING
We drew up test plans, with detailed lists of things to do during each test slot. The very first test, for example, was just to see whether the system came up with RACF in it. Then, progressively, we tested more and more software products, until, at the end, we conducted a full-blown test. During this final test, many programmers and operations staff were asked to participate, to see whether they could spot any problems.

All the software products were examined and tested under RACF, to make sure their interfaces to security, if any, still worked. For those products that had an external security interface, we looked at the product’s installation manual. This manual usually told us what changes were needed to make them work under RACF. In some cases, the manual told us to re-assemble a module or two.

Automation software, which handled operator console replies to ACF2 messages, had to be modified to support RACF messages.

We prepared a list of helpful ‘Hints and Tips’ for the user community as we progressed with the testing, and itemized all possible changes that we could think of. For example, we included such things as: “All RACF messages start with ‘ICH’. Under ACF2, the messages were prefixed by ACF.” These notes were distributed to the user community a week before the ‘live’ conversion date.

SOME SURPRISES
One of the things that caught us by surprise was the fact that all RACF userids must have a password – even those that will never be used to sign on to TSO or CICS (surrogate userids). Even ‘started-task’
userids needed passwords! In such special cases, ACF2 allowed userids to not have a password.

For these surrogate and started-task userids, we simply gave passwords that were hard to guess, and forgot about them. We later learned that there are RACF add-on packages that will take care of passwords for these userids. We were also told that IBM has been requested to provide for no-password userids.

We also found that the OPERATIONS attribute of RACF is not as powerful as its ACF2 counterpart, the NON-CANCEL attribute. Because of this, during testing, we found userids with OPERATIONS attribute failing on some accesses. Most of these failures were resolved during the tests.

GOING LIVE WITH RACF

Finally, the day came to go live with RACF. We chose a Monday, so we had the weekend to do last-minute preparations.

On this first day, the biggest headache was not any technical difficulty, but password changes! We had assigned new passwords to all userids. Also, the RACF sign-on screens were slightly different. This caused confusion among some infrequent users of the system, and they had problems signing on to the system that first day. RACF revoked their userids after they reached the maximum number of allowable sign-on attempts. The Help Desk was kept busy helping these people sign on correctly.

To ease this situation somewhat, we increased the maximum allowable sign-on attempts in RACF, just for that first day. This gave the users extra chances to sign on correctly that first day.

SUMMARY

Since security software is at the heart of the operating system, this type of conversion should be handled with care. Backing out of the change is difficult, and would affect the entire user community. For this reason, a lot of planning and testing is required. But it can be done.

Dinesh Dattani (Canada)
Remote security – inexpensive firewalls

Most organizations still rely solely on VPN and RACF to protect mainframe access from remote workstations connected via commercial high-speed Internet – despite the fact that those workstations are unprotected from hackers gaining access to them from the Internet, and going on to take over a logged-on RACF session.

A random check of a major telco confirmed that it doesn’t even recommend, let alone require, a firewall for employees connecting to the mainframe from high-speed Internet at home. Ironically, it tells its residential customers to install a firewall because, like virtually all ISPs, the telco’s firewall protects its staff, not its customers.

In our continuing look at SOHO hardware firewalls for the small remote office or home mainframe user, we switch to cable/DSL gateway routers with built-in firewalls and VPN, instead of the much more expensive firewall with built-in hub. Of course, software firewalls are even cheaper, or free with the operating system, as in Windows XP. But they leave the operating system itself unprotected against a direct attack.

BELKIN

The Belkin 4-Port Cable/DSL Gateway Router (F5D5230-4) is an unusual shape, sits vertically, and has a docking ring to physically attach (side-by-side) with other Belkin products. Although the small installation manual suggests answering some networking questions first and installing set-up software on your workstation, you’ll find it faster to just connect your modem and workstation(s), and then power up the router and see if it works. It worked for me with my ADSL modem, and Windows XP Professional and Windows 2000 Server workstations, but I:

- Use dynamic IP addresses (DHCP)
- Do not use PPPoE
- Previously had the two workstations communicating with each other through the hub of a SOHO firewall.
The CD-ROM included is common to this router and Belkin’s 5- and 8-port network switches. An install.exe file in the root directory installs the NetSetup and NetShare software common to all three products. A Manuals subfolder on the CD-ROM contains the same installation manuals available on the Belkin Web site and included on paper with each product. An Acrobat Reader subfolder installs Adobe Acrobat Reader, which is required to read these manuals.

INSTALLATION MANUAL

The installation manual does its job but, like many new products, there are some errors and problems:

- It is written for Windows 9x/ME, most notably for winipcfg instead of NT/2000/XP’s ipconfig, which even labels some fields differently. For example, MAC Address is labelled Adapter Address instead of Physical Address.

- ‘Basic Parts’ (p 11) incorrectly states “2 networking cables for each computer, one for the Modem-to-Router connection; and one for the PC-to-Router connection.”

- Page 12 refers to “one of the ports on the rear of the Gateway Router labelled LAN”, when nothing is labelled LAN.

- There is no indication as to whether the Belkin-supplied software works on anything but Windows 9x/ME.

Although the manual implies that Intrusion Detection is off by default, the Web-based set-up utility gives the impression that it is on (which it is). But neither is clear. The manual doesn’t indicate how to add additional workstations later, but powering everything off first seems logical given specific instructions to do that during the initial install. Nonetheless, hot-plugging a second workstation worked flawlessly when tested.

Until they are on, the LAN port status lights look as if they’re unnumbered. But when they’re lit, you can see the port number at the bottom of each oblong light. Although I didn’t experience any problems related to it, the FAQs on the Belkin Web site tell you to turn off Windows XP’s built-in software firewall:
http://web.belkin.com/support/faq_qa.asp?pid=12&cid=1#1053

With lots of helpful photos, there’s also a Set-up Guide available only on the Web site at:

http://www.belkin.com/networking/setup/guide.html

WEB-BASED SET-UP UTILITY

Like the other firewalls we’ve looked at, the Belkin has its own IP address: 192.168.2.1. Point a Web browser at it from any workstation connected to its hub and you can manage the firewall with what Belkin calls a Web-Based Set-up Utility. But, unlike the previously-reviewed firewalls, the Belkin prevents simultaneous usage.

Even if you close the browser or shut down and power off the workstation, you’ll still get an error if you try to log on from another workstation:

Duplicate Administrator
This device is managed by 192.168.2.33 currently!!

Because there is no logout function, all you can do is wait for the relatively short session timeout. But, for some reason, you may then have to enter your password twice.

Once logged on, you’ll see the Status display with Initial Set-up, Utilities, Status, and Help listed in the left sidebar. Initial Set-up is where almost everything is, even the firewall’s log of refused access attempts, under Security/Security Log.

Although not the Internet Explorer (IE) default, I always force IE to check every Web page for currency:

- From the IE menu bar, Tools/Internet Options.
- On the General tab, push the Settings button in the Temporary Internet files section of the dialogue box.
- Select Every visit to the page.
- Click OK twice.

Some Web sites don’t work properly with this setting, but Belkin’s Web-Based Set-up Utility works best this way. Otherwise, hitting the IE Back button can display out-of-date information.
<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum users</td>
<td>253</td>
</tr>
<tr>
<td>LAN ports</td>
<td>4 x 10/100Mbps (autosense)</td>
</tr>
<tr>
<td>LAN port status lights</td>
<td>One per port: amber for 100, green for 10, flashes for activity</td>
</tr>
<tr>
<td>WAN port status lights</td>
<td>One: green for connection, flashes for activity</td>
</tr>
<tr>
<td>Other status lights</td>
<td>Ready</td>
</tr>
<tr>
<td>Power switch</td>
<td>None</td>
</tr>
<tr>
<td>VPN</td>
<td>Included with PPTP and IPSec pass-through</td>
</tr>
<tr>
<td>Autodial back-up</td>
<td>No</td>
</tr>
<tr>
<td>Back-up throughput</td>
<td>Not applicable</td>
</tr>
<tr>
<td>Remote management</td>
<td>Yes, but default is off</td>
</tr>
<tr>
<td>List price (US)</td>
<td>$138</td>
</tr>
<tr>
<td>Lowest street price (US)</td>
<td>$70</td>
</tr>
<tr>
<td>Firmware updates</td>
<td>Part of lifetime warranty (free)</td>
</tr>
<tr>
<td>Size (imperial)</td>
<td>7.6&quot; x 7.5&quot; x 1.9&quot;</td>
</tr>
<tr>
<td>(metric)</td>
<td>192 x 189 x 46 mm</td>
</tr>
<tr>
<td>Weight (imperial)</td>
<td>15 oz</td>
</tr>
<tr>
<td>(metric)</td>
<td>423 g</td>
</tr>
<tr>
<td>Shipping size (imperial)</td>
<td>12.2&quot; x 9.9&quot; x 3.1&quot;</td>
</tr>
<tr>
<td>(metric)</td>
<td>310 x 250 x 77 mm</td>
</tr>
<tr>
<td>Shipping weight (imperial)</td>
<td>2 lb</td>
</tr>
<tr>
<td>(metric)</td>
<td>1 kg</td>
</tr>
<tr>
<td>Power transformer size</td>
<td>3.1&quot; x 2&quot; x 1.25&quot;</td>
</tr>
<tr>
<td>(metric)</td>
<td>79 x 51 x 32 mm</td>
</tr>
<tr>
<td>Polarized power plug?</td>
<td>No</td>
</tr>
<tr>
<td>Power cord length</td>
<td>6 ft</td>
</tr>
<tr>
<td>(metric)</td>
<td>1.84 m</td>
</tr>
<tr>
<td>Power transformer output</td>
<td>5 volts, 2.4 amps</td>
</tr>
<tr>
<td>LAN cables included</td>
<td>None</td>
</tr>
<tr>
<td>Firewall IP address</td>
<td>192.168.2.1</td>
</tr>
<tr>
<td>LAN IP address</td>
<td>192.168.2.n, assigned randomly</td>
</tr>
<tr>
<td>Log format</td>
<td>See Figure 3</td>
</tr>
<tr>
<td>Log display order</td>
<td>Chronological</td>
</tr>
</tbody>
</table>

**Figure 1: Specifications**
INSTALLING THE SOFTWARE

Double clicking the install.exe program on the CD-ROM installs the NetSetup and NetShare programs. In Windows XP, Start/All Programs/ Belkin SOHO Networking/Belkin NetSetup starts NetSetup. Despite the fact that the router was first released in August, before Windows XP was released, the software works properly in Windows XP. For example, NetSetup correctly initiates the XP Network Set-up Wizard rather than the Belkin software you would see on Windows 9x/ME. Of course, the dialogue boxes look nothing like the ones in the installation manual.

EVALUATION

Figure 1 shows the same specifications as were used in the last issue (RACF Update 27, February 2002, pp 54-55) for the Symantec and WatchGuard stand-alone SOHO firewalls. Figure 2 shows the Belkin’s log file format. Figure 3 compares all three firewalls by a few new specifications. Note that all size measurements in Figures 1 and 3 were made with the Belkin’s removable foot removed.

Figure 4 shows the results of tests performed. Explanations of some of these tests were included in last issue (pp 56-58).

<table>
<thead>
<tr>
<th>Belkin Router</th>
<th>Symantec Firewall/VPN 100</th>
<th>WatchGuard SOHO</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD-ROM included?</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Hot pluggable</td>
<td>Not recommended</td>
<td>Yes</td>
</tr>
<tr>
<td>Reset button</td>
<td>On rear</td>
<td>None</td>
</tr>
<tr>
<td>Footprint (imperial)</td>
<td>7.25&quot; x 1.6&quot;</td>
<td>11&quot; x 5.5&quot;</td>
</tr>
<tr>
<td>(metric)</td>
<td>184 x 41 mm</td>
<td>280 x 140 mm</td>
</tr>
</tbody>
</table>

Figure 2: Comparing some new specifications

Figure 3: Belkin log

2002/03/06 15:08:12 **Unauthorized HTTP Access** <TCP> Source IP: 208.179.251.103 Port: 61781 Dest IP: 161.184.156.86 Port: 88
2002/03/06 15:32:03 **TCP SYN Flooding** <TCP> Source IP: 208.179.251.103 Port: 61780 Dest IP: 161.184.156.86 Port: 1723
CONCLUSION

Since 11 September, interest in security has created a huge potential market for SOHO hardware firewalls. Firewalls have begun to appear in high-speed Internet gateway routers, at prices 80% below stand-alone SOHO firewalls.

But don’t let the prices fool you. The Belkin router demonstrates that a new industrial-strength generation is emerging, replacing the toy firewalls we’ve seen in routers in the past. These routers (especially those with VPN support built in) are ideal for the small remote office or home user accessing the mainframe via high-speed Internet.

---

Jon E Pearkins  
(Canada)  
© Xephon 2002

---

E-mail alerts

Our e-mail alert service will notify you when new issues of RACF Update have been placed on our Web site. If you’d like to sign up, go to http://www.xephon.com/racf and click the ‘Receive an e-mail alert’ link.
IBM SECURITY HOME PAGE – http://www.ibm.com/security

As mentioned in this issue’s ‘Inside IBM’ article, the IBM security home page is the place to start for information on IBM’s products and services relating to security and privacy. The page is divided into lead stories, security news, white papers, and events.

The left sidebar is effectively the security main menu, with the following options: news, services, products, case studies, library, education, standards, research, partners, privacy, and events.

At the bottom of the sidebar, there are also the following related links: warranty information, IBM software, Identrus, PC security, privacy, Tivoli security, and wireless.

The Products link takes you to a Web page that seems to do little more than categorize and list major IBM products with security connections (see http://www.ibm.com/security/products). The categories are: cryptography, Lotus, middleware, PC security, intrusion detection, secure servers, and security management.

Many popular mainframe products are listed even though their primary purpose is not security. And when you click on them, you’re taken to a security-specific page for that product, not the product’s home page.

An interesting exception is the last link under Secure Servers: Evaluated Products. This lists the testing and certification of IBM security products by international standards organizations.

The Library link on the Security home page leads to a wealth of information. So much, in fact, that the Library Web page is little more than links to each of the categories: White papers, Redbooks, security brochures, journals, magazines, and newsletters, books, Web sites, FAQs, mailing lists/archives, features archive, glossary, and government security topics.
There’s also a lot of information from the Research link on the Security home page. And the Privacy link is the starting point if privacy is your area of interest.

SOMETHING NEW TO WORRY ABOUT – http://applied-math.org

Although it won’t appear in print until August, you can read Information Leakage from Optical Emanations now in compressed PostScript or Adobe Acrobat PDF.

Although it may not be obvious from the title, the paper discusses the very real possibility of reading the data going through modems and other communications devices simply by monitoring the status lights. Given a clear line of sight, the advanced optics used in reporters’ cameras should allow viewing from over a mile (1.6 km) away, though the authors speak mostly about across the street.

This eavesdropping ability lies in one property of the Light-Emitting Diodes (LEDs) used for status lights. They can go on and completely off very, very quickly – at speeds measured in tens of nanoseconds. The authors successfully and accurately read data at speeds as high as 56Kbps, but they theorize that 10Mbps is completely possible. They also found some stand-alone data encryption devices, as well as modems and other devices with built-in encryption, that displayed the unencrypted data on their status lights!

UNIVERSITY OF TENNESSEE RACF PAGE – http://utkvm1.utk.edu/racf.html

If you’re looking for a single Web page documenting common RACF commands and functions, including RACF Report Writer, as well as an overview of RACF concepts, this might fit your needs quite well. Like all well-designed very long Web pages, this one begins with a table of contents in the form of links you can click on to get further down the page to the section you’re looking for. The audience is RACF administrators in a z/OS environment, and there is some terminology used that’s local to the site, such as project director, project administrator, User Services consultants, Request for Services form, and UTCC. MVS is used to refer to z/OS.
One step above, at http://utkvm1.utk.edu, you’ll find more mainframe documentation and information, such as the fact that UTCC stands for the 42-year-old University of Tennessee Computing Centre. Of more practical interest is *The IBM User’s Guide*, with a Table of Contents as large as the RACF page, covering both z/VM and z/OS. Chapter 2, Policies and Procedures, is concerned mostly with RACF/user ID administration.

**UNIVERSITY OF CINCINNATI** – http://www.cas.ucit.uc.edu/security

Despite the fact that it is a CA-ACF2 site, the University of Cincinnati Office of Information Technologies (UCit) Core Application Services (CAS) provides an interesting example of access control procedures for a large number of short-term employees, ie students. The Policies and Procedures link lists a dozen sensible rules; FAQs and Information answers some common questions; and the Mainframe Access link documents and includes links to on-line forms.

*Jon E Pearkins  
(Canada)*  
© Xephon 2002

**Need help with a RACF problem or project?**

Maybe we can help:

- If it’s on a topic of interest to other subscribers, we’ll commission an article on the subject, which we’ll publish in *RACF Update* – it won’t cost you anything.

- If it’s a more specialized, or more complex, problem, you can advertise your requirements (one-off projects, freelance contracts, permanent jobs, etc) to the hundreds of RACF professionals who visit *RACF Update*’s home page every month. This service is also free of charge.

Visit the *RACF Update* Web site

http://www.xephon.com/racf

and follow the link to *Opportunities for RACF specialists*. 

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
CA’s eTrust PKI 2.0 focuses on rapid deployment of Public Key Infrastructure (PKI) and can also be used by eTrust Single Sign-On to provide access to the mainframe and many other platforms.

Although eTrust Directory supports both LDAP V3 and X.500, it has been tuned to outperform even LDAP-only solutions. The DXLink feature enables any LDAP-compliant server to be incorporated into the eTrust Directory backbone.

For further information contact:
Tel: (631) 342 6000.
URL: http://www3.ca.com/Solutions/Collateral.asp.

***

Mainstar has upgraded its back-up and recovery software with new enhancements, including a disaster recovery report with RACF functions to support primary and line commands.
For further information contact:
Mainstar Software, PO Box 4132, Bellevue, WA 98009-4132, USA.
Tel: (425) 455 3589.

***

Candle’s Version 210 of MQSecure incorporates RSA BSAFE Cert-C PKI software to further enhance the end-to-end security it provides for MQ (WebSphere MQ, formerly MQSeries) networks. Candle is also using RSA BSAFE Crypto-C libraries in the product. BSAFE recently passed the US Federal Information Processing Standards (FIPS) 140-1 Cryptographic Module Validation Program.

For further information contact:
Candle, 201 N Douglas St, El Segundo, CA 90245, USA.
Tel: (310) 535 3600.

***

SAM Jupiter is Systor’s name for the next version of Security Administration Manager (SAM). It includes a more user-friendly user interface, an optimized workflow, and a business process oriented design.

SAM provides distributed security administration in a single repository for access control data, and supports Windows NT/2000, NetWare, Unix, RACF, CA-Top Secret, CA-ACF2, DCE, SAP R/3, Lotus Domino, Oracle, and DB2, with Connectors for LDAP and application security.

For further information contact:
Systor Security Solutions, 6411 Ivy Lane, Suite 610, Greenbelt, MD 20770 USA.
Tel: (301) 486 4600.

***