E-SRF

EKC Security
Event Reporting Facility

Release 2.1

Masterfile and Data Dictionary Reference

E-SRF™ is a proprietary product
developed and maintained by

EKC Inc.
10400 West Higgins Road
Rosemont, Illinois 60018
USA

(847) 296-8010

Technical Support:
(847) 296-8035

EKC, Inc. provides only software program products, which fully comply with, and maintain MVS integrity.

The vendor hereby warrants that:

1) E-SRF™ ("Software") performs only those functions which are described in the published specifications;

2) there are no methods for gaining access to the Software or other computer resources or data of Licensee (such as a master access key, ID, password, or trap door) other than set forth in the published specifications;

3) the Software does not introduce any MVS integrity exposures. The program code, with the exception of one utility, runs totally in non-authorized, problem state. The one utility, EKCRXCAT, requires APF-authorization to read the MVS System Catalogs. A non-APF authorized utility, EKCRGCAT, is supplied to perform the same function, but at a considerably slower speed.

4) the software shall be year 2000 compliant, and shall function correctly in the next century according to published specifications as long as regular software maintenance is applied.

Version 2, Release 1 February 13, 2005 (Revised for: LE00450)

Printed in USA

Contents

Name	Contents
Installation Guide	E-SRF installation including: installation and maintenance steps, startup and shutdown considerations, and backup and recovery procedures.
Change Summary Guide	Contains all new features and system function changes.
General Overview	An overview of E-SRF and its components.
Resource Grouping Facility Guide	Brief overview of the Resource Grouping Facility, its relationship to E-SRF, language command syntax, TSO commands and JCL.
Access Analysis Reports Guide for ACF2 *Access Analysis Reports Guide for RACF*	Brief overview of Access Analysis reports, explanation of the DataOwner and Userid/LogonidOwner reports, command syntax, utilities necessary for creating input to reports, and JCL.
Event Reporting User Guide	A "How To" guide for users of E-SRF Event Reporting.
Event Reporting Facility - Command Reference	Explains the Event Reporting Facility command processor, command syntax, and JCL.
Event Reporting Facility - Masterfile and Data Dictionary Reference	Explains the structure of the E-SRF Masterfile and describes all Masterfile fields.
Event Reporting Facility - Messages and Codes	Lists Event Reporting Facility messages and codes.
Event Reporting Facility - Report Overlays Guide	An overview of the report overlays provided with the Event Reporting Facility.

This page intentionally left blank

Chapter 1: Introduction

The E-SRF Event Reporting System maintains its own database to maintain information about the people, places, and things in your computing environment.

The information stored on this database is used to create reports.

This repository of information is referred to as the E-SRF "Masterfile". The Masterfile is the heart of Event Reporting.

Using the E-SRF UPDATE FUNCTION, security related information such as violations, loggings, and administrative data from your Resident Security System (RSS) are applied to the Masterfile. This information is gathered from the security system's journal files (*usually the operating system's SMF data*), normalized, and distributed to the appropriate Masterfile Segments and Objects (*discussed later*).

For example, a security logging contains several different types of information such as a *userid* (the person who attempted access), the *source* of the terminal the user was sitting at, and the *dataset* or *resource name* the user was attempting to access.

On the E-SRF Masterfile, each type of information is stored in its own object so you can maintain and report on the information from any desired perspective.

Information on the functionality of the UPDATE process is described further in this publication. For more information about the usage of the UPDATE command, refer to the: *E-SRF Event Reporting Command Reference*.

In addition to storing security journal information, E-SRF also keeps statistics, such as how many violations were recorded against a particular resource, or how many signon errors a user has had. This statistical information takes up very little space on the Masterfile, and can be stored for a longer period than security journal information. This is especially useful when producing long-term trend analysis reports.

This page intentionally left blank

Chapter 2: Masterfile Characteristics

Basic Information

Why a "*Masterfile*" and not a commercial database?

The design of the E-SRF Event Reporting Masterfile (*normally referred to as "the Masterfile"*) was to be a multi- dimensional database capable of storing and retrieving massive amounts of data as quickly and as efficiently as possible. The structures engineered for the Masterfile were designed for, intended for, and optimized for this particular application. No attempt was made to make the Masterfile architecture a universal data handling facility (such a commercial database). Doing so would make processing the Masterfile much less efficient for this application. This was the reason the Masterfile architecture was designed and developed instead of using a commercially available database manager.

The E-SRF Masterfile is a VSAM KSDS Cluster

The E-SRF Masterfile is stored on disk using the VSAM (Virtual Storage Access Method) access method as a KSDS (Keyed Sequential Data Set) VSASM Cluster. VSAM was chosen for this application simply because VSAM was (and still may be) the recommended base access method of choice for MVS systems using DASD (Direct Access Storage Devices) as a storage medium at the time the system was initially designed and developed.

In E-SRF Event Reporting, VSAM is simply a place to store your data when it is not being used. Access to the actual data is controlled by the E-SRF Event Reporting system. VSAM alone can locate particular records, allow you to read and write to these records, but on its own could not be used to process the Masterfile. This is also true with other database products that use VSAM to physically store its data on DASD.

Masterfile OBJECTS

The basic unit of storage on the Masterfile is the "*object*". An object is similar to a *record* and contains a set of related data fields. E-SRF Event Reporting Masterfile Objects may contain up to sixteen million characters of data relating to the object's key. Masterfile Objects are normally referred to as OBJECTS

Masterfile SEGMENTS

A Masterfile SEGMENT (normally referred to as SEGMENT) is a logical grouping of like objects. For example, all Objects that relate to "*resources*" are collectively referred to as the RESOURCE SEGMENT. You will learn more about this later.

VSAM and physical record segments

The Masterfile data is stored on disk using the VSAM access method. To accommodate VSAM's maximum record limitations, objects are divided into "*VSAM Physical Record segments*". Additionally, objects may be compressed to conserve both processor storage and disk storage.

What this really means is we may have a very large amount of data that makes up a Masterfile Object. Possibly much more than can fit on a single VSAM logical record. E-SRF will divide-up the object into individual "pieces" that will fit on VSAM records and store the object's data onto these VSAM records. These individual VSAM records are also called "segments". Yes, this may be confusing, but these segments are have no meaning to you, other than you knowing some trivia on how the Masterfile is constructed.

A VSAM "physical record segment" is really just a logical record to VSAM and any program that may attempt to process the Masterfile's VSAM Cluster.

VSAM record segments and their internal format are transparent to you. When an object is needed, the entire object is assembled in storage. It's data fields are normally referenced by the datanames contained in the data dictionary.

The E-SRF Masterfile file structure is very complex and as of this release may only be accessed by the E-SRF system. Attempting to access the Masterfile through other methods will yield unpredictable results.

If after reading the information contained in this and other E-SRF Event Reporting publications, there may be something you need to do that you feel may not be in this product, please feel free to contact EKC Technical Support for assistance. Many reports were developed this way and this trend will continue as long as customers have requirements.

There are segments and there are "segments"

This can be very confusing, but is does not have to be.

As mentioned before, normally the discussion of "*segments*" will not refer to VSAM record segments. VSAM segments were discussed here only for your information. They relate to how the data is physically stored on VSAM. They have no meaning to the usage of this product on a normal day-to-day basis. "Masterfile SEGMENTS", as you will learn, are very important to the day-to-day use of this product. You will learn about Masterfile Segments very quickly. Please do not confuse these terms.

How data is stored on the Masterfile

There will be plenty of discussion about this in subsequent topics in this publication. In a nutshell, the Masterfile contains all the data that is required to produce Security Event Reports. The information is acquired from input data, such as the loggings that are produced by your security system.

Data items are all normalized, classified, named and stored on the Masterfile. Each item contains a name. A data dictionary is used to identify and locate data when required for processing. Individual data items are in a wide range of data formats. Field sizes may be one bit (*yes, bit, not byte*). Bit fields are usually YES or NO indicators. Character fields, such as resource names may be as large as one thousand characters in length.

Tokenization

Event Reporting releases prior to release 2.1 imposed limits on how long character fields could be. This was due to the amount of disk and memory required to contain them, as well as certain VSAM limitations, such as the maximum length of a VSAM record's key.

In latter releases of MVS, support for longer resource names was provided which created a need for Event Reporting to maintain very large data fields.

Concurrently with these new requirements, Masterfiles were becoming quite large and consuming large amounts or resource and central processor time required to update. Event Reporting needed to have its performance improved at the same time additional data needed to be stored on the Masterfile.

The solution in release 2.1 was to tokenize particular data items into "tokens" (*three byte representations of specific data*). The token is stored on the Masterfile along with the full contents of the data item in the Masterfile's token dictionary. When a particular character string (such as 1,000 character resource names, along with their class and volume) is presented, a token is associated to the name using the token dictionary. Now, all that is stored is the token representing the actual data. The token is resolved on demand when the data item is needed for processing.

Tokenization alone greatly reduced the size of the Masterfile and the resources necessary to process the data contained on it. It did however increase the instruction path required to access the tokenized data. This increase was completely offset by the savings gained by not processing the long data. The net result was a reduction of overhead required to manage the Masterfile.

The Update Function now performs much more efficiently than in releases prior to 2.1.

During the production of reports, there may be a slight increase, *if at all*, of processor requirement with reports that use tokenized data. Event Reporting was always optimized for report production. Any change in resource usage targeted at facilities that are already optimized will normally be noticed.

Masterfile Format Compatibility across Releases

As in any data processing application, the file contents and organizational structure can change from one release to another.

The E-SRF Event System maintains certain control information that identifies the Masterfile's current level. E-S-SRF knows which level the Masterfile has to be in order to process the data contained within it. If there is a discrepancy, an attempt is be made to convert the Masterfile *UPWARD* to the current level supported by the E-SRF version being executed.

This means the product's Masterfile is *UPWARD COMPATIBLE*, and you are able to automatically convert a Masterfile from *ANY* previous release to the current product release.

When a discrepancy is detected, the Conversion Upgrade Function is invoked.

Conversion is automatic… but you have to allow it.

Yes, the product will automatically detect the need for conversion and it will try to run the conversion for you.

Because of the impact of a conversion, (*you cannot go back unless you restore your Masterfile VSAM Cluster*), the product needs confirmation from you that the conversion should be performed.

Before any conversion is performed, the Upgrade Function tests the Command Processor's execution entry parameter (normally specified via the "PARM" parameter in your JCL). The constant "UPGRADE" (PARM=UPGRADE) must be present for the conversion to occur. This is to prevent converting a Masterfile when you really do not want it converted.

Please note that this is an *UPWARD* conversion. It is not possible to convert a converted Masterfile back to a lower product level.

Failure to provide the UPGRADE parameter when a conversion upgrade is required will cause the current execution to be *abended* via ABEND964 following the message E490-MCV.

If you run the Command Processor with the UPGRADE parameter and a conversion upgrade is not required, the message E491-MCP will be posted, as a warning and normal processing will continue. It is not advisable to run with the UPGRADE parameter unless a conversion is possible. The purpose of the UPGRADE parameter is to prevent converting a Masterfile to a particular product release and expecting to run a lower release level against the same Masterfile.

Consider testing a new version of E-SRF using a copy of your production Masterfile.

What really happens when you convert?

During conversion, the Masterfile is normally loaded into the Level 2 cache. (You will learn about the Level 2 cache later). The data is reformatted by the conversion programs (in the cache) as per the requirements of the conversion being run. All subsequent processing will be executed as if the Masterfile was at the required release. When the cache is written back to the Masterfile's VSAM Cluster, the Cluster will be at the new release.

How far back can you convert?

Each release that alters the format of the Masterfile has its own conversion facility. The Conversion Function will step up the Masterfile data from the Masterfile's current release to the Masterfile release currently required for the version of the product being executed.

Older Masterfile release converters have been retained on the system. This means you can skip over releases and still use an older Masterfile.

More important, you can process an old Masterfile that you may have saved years ago with the latest release of the software.

This page intentionally left blank

Chapter 3: Masterfile Organization

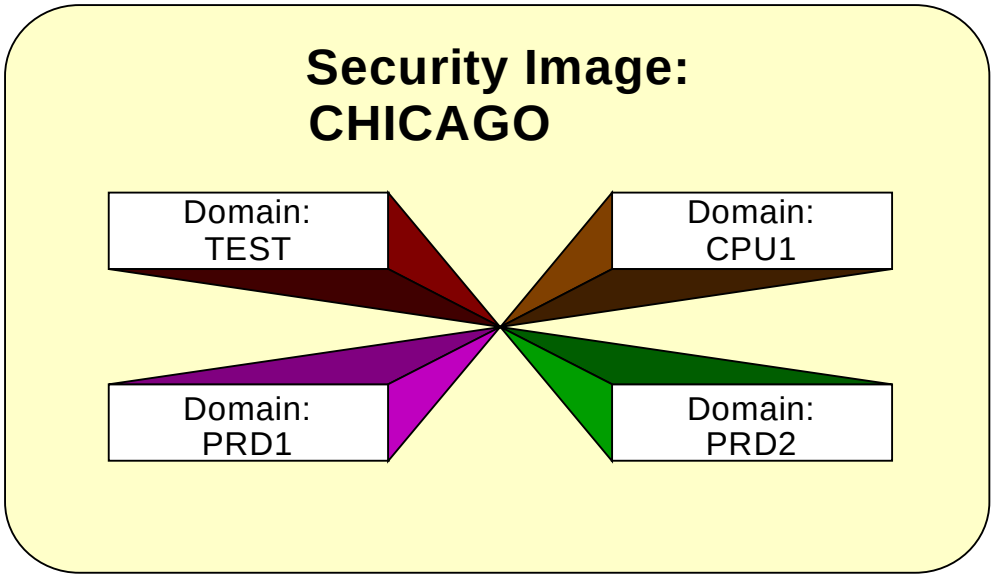
Any information in the Masterfile can be used to create reports. You simply tell E-SRF which parts you want to see by giving E-SRF the *name* of the place where that information is stored. E-SRF divides the information into individual items of data and stores them in groupings with other related information. Those groupings are called IMAGES, DOMAINS, SEGMENTS and OBJECTS.

This section details the structure of the Masterfile and how you use that structure to identify which information, you want in a report.

IMAGES and DOMAINS

A security IMAGE is the largest *grouping* in Event Reporting. IMAGE refers to the users and resources that share a common set of security databases.

For example, ABC Company has three MVS machines in Chicago: TEST, CPU1, and PRODUCTION. Production is divided into two LPARs (Logical PARtitions) - PRD1 and PRD2. There are four different MVS systems referred to as SYSIDs, LPARs, or Domains. All four domains use the same set of security databases. Therefore, in E-SRF terms, there is one CHICAGO Security Image.



Each security image in E-SRF includes information about the people, places, and things on the domains assigned to that image. All events that take place on those domains are captured by the RSS and applied to the Masterfile for the CHICAGO security IMAGE.

This means IMAGES are further divided into DOMAINS. The Chicago IMAGE will contain for DOMAINS: TEST, CPU1, PRD1, and PRD2.

Images

IMAGES are “CONFIGURED” (*and SYNCHRONIZED*) on the Masterfile, and represent a specific set of Resident Security System (RSS) databases used to administer security.

An IMAGE is always associated with a specific RSS. You may have multiple IMAGES on the same Masterfile that deal with different implementations of the same or different RSS.

IMAGE is the highest level grouping of Event Reporting data.

The Resource and Source Masterfile segments do not physically separate their data by IMAGE, but IMAGE may be used to refer to data related to a specific IMAGE.

The User Masterfile segment does separate its data based on IMAGE. The IMAGE ID is part of the User Segment object key. This is because the same USERID may not be the same user across IMAGES.

If the USERID “JOHN” is in two IMAGES, the ID may be representing John Smith in one IMAGE, and John Jones in another. For this reason, the User Segment data must be separated.

The DOMAIN (*explained in the next section*) is used to separate events from one system to another. DOMAINS are assigned to IMAGES. This tells the system which DOMAINS belong to which IMAGES.

For example, CICS transaction TR01 can be executed on the *TEST* and *PROD* MVS systems. The Resource segment for TR01 will contain any event recorded for TR01 on *TEST* and *PROD*. On the User Segment, the situation is the same, except if the two systems (DOMAINS) are *assigned* to different IMAGES; they will reside on objects belonging to the proper IMAGE.

Even though User information is the only data dependent on the IMAGE, IMAGE ID may be used to reference resource *or* user events. If IMAGE is used with a RESOURCE Segment object, it is determined by the assignment of the DOMAIN on which the resource event occurred. If IMAGE is used with a USER Segment object, it is processed at the object key level.

Domains

As discussed earlier, an E-SRF "Domain" refers to a specific MVS system that is commonly referred to as an LPAR or SYSID.

In the world of MVS, the operating system executes on a mainframe either all by itself, or in a portion of the mainframe, referred to as an LPAR (L*ogical* PAR*tition*). Other hardware vendors can have other names for this, but LPAR appears to be the universal term.

When MVS runs in an LPAR, it executes as if it was the only operating system on the physical mainframe computer. The mainframe computer may host one or more LPARs and some of these may be running operating systems other than MVS.

MVS has a term that represents an individual execution of the MVS operating system. The usual term is the four character MVS "SYSID". Sometimes, it is referred to as the SMF ID, or even a JES ID. No matter what it may be referred to in your installation, it means the same thing to E-SRF.

Do not confuse the MVS term SYSID with the *CICS "SYSID"*. They have completely different meanings. The CICS SYSID has no relevance in E-SRF.

Other operating systems may refer to their existence on a specific piece of hardware as a "DOMAIN".

Because of the confusion with this terminology, E-SRF has adopted the term "**DOMAIN**" to describe a specific copy of an operating system (such as MVS) running on a specific computer system complex. In the Data Dictionary, the name "SYSTEM" may be used interchangeably with "DOMAIN". However, DOMAIN is the preferred term.

Event Reporting supports an eight character DOMAIN ID associated with each event. Events are stored on a Masterfile object in chronological order from oldest to most recent. Events from all Domains are stored together. The DOMAIN ID is treated as another data field captured along with an event. This is how E-SRF determines which system the event occurred on. It also tells the Update Function how to store the event on the Masterfile.

IMAGES consist of one or more specific *DOMAINS* that share a common set of security databases.

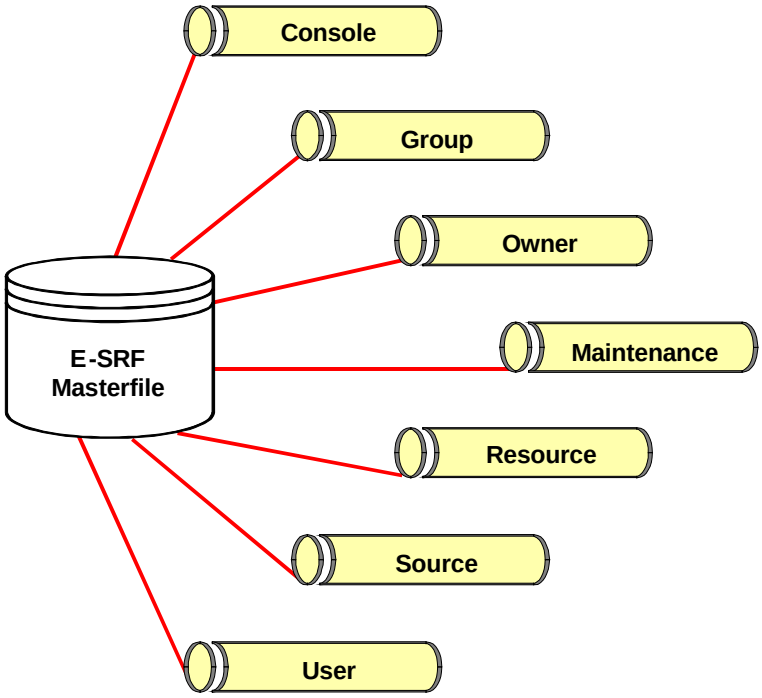
Domains are NOT actually defined to the Masterfile. You *ASSIGN* DOMAINS to existing IMAGES using the *ASSIGN* command. The only Masterfile objects directly affected by a domain assignment are those contained in the USER Segment. The IMAGE is part of the object's key. The key is USERID and IMAGE. These two datanames, however, can be treated separately in reports. Refer to the *Event Reporting Command Reference* and the *User Guide* for more information on how to *assign* DOMAINS to IMAGES.

In Event Reporting, you cannot have two systems with the same DOMAIN name. Event Reporting will attempt to treat both as the same DOMAIN. If this situation exists, you must pre-process your journalized event data by altering the supplied *domain ID* to something unique, and separate out unlike Resident Security System (RSS) data to individual journal datasets. An individual Update Function execution can only processes a single RSS.

In summary, security events originate in DOMAINS. DOMAINS "live" in IMAGES. A DOMAIN is an eight character Masterfile field that is captured and used by Event Reporting to track on which system a specific event occurred. In the case of MVS, a DOMAIN is the four-character SYSID, left justified, padded with blanks.

Segments

The Masterfile is divided into SEGMENTS. A "segment" *is a grouping of information that is related in some way*. For example, the USER segment contains information about the users on your computer system; the RESOURCE segment contains information about the datasets and resources in your computing environment. A segment may also be viewed as a "file within a file".



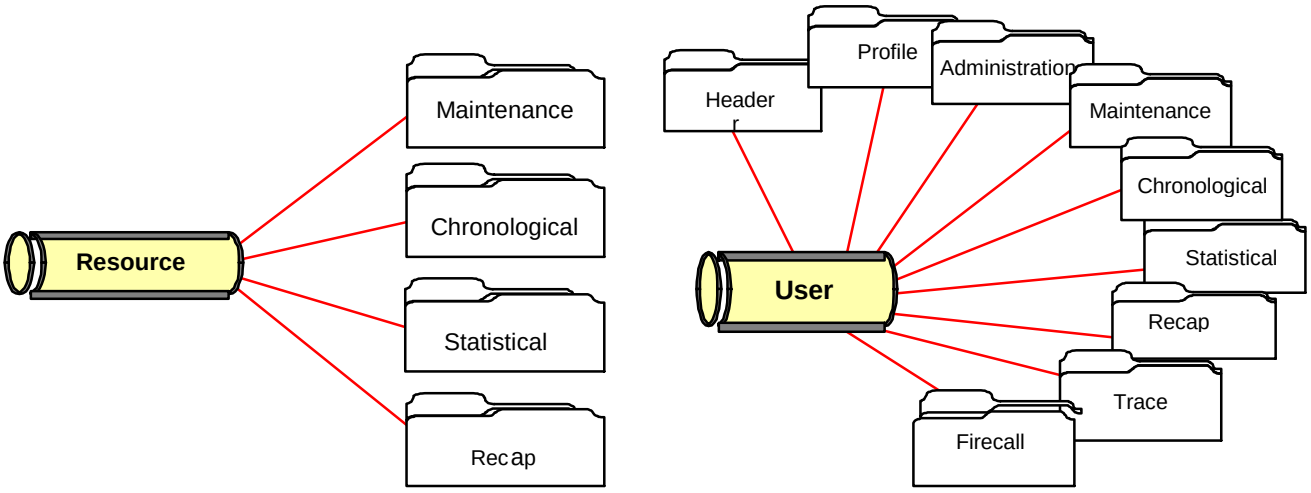
There are seven segments in the current release of E-SRF: Console, Group, Owner, Maintenance, Resource, Source, and User. In addition, E-SRF includes segments that store E-SRF control information, the data dictionary, and user-related items of data from your Resident Security System. For example, the fields of the ACF2 Logonid record or user information fields from the RACF database are stored in the E-SRF Masterfile in separate objects within the User segment.

Data from one segment may interact with data from another. Consider a resource violation report that includes the user's name and department code in the report output. The user information comes from the User Header (UA) object contained in the User Segment. The violation information comes from the Resource Statistical (RS) object contained in the Resource Segment of the Masterfile.

Objects

Segments are further divided into smaller groupings called OBJECTS. The *Object Types* represent similar types of data about that segment. For example, the USER segment includes a HEADER object, which describes a particular user. Another object, the User Chronological object, provides complete detailed information relative to a specific security event that has occurred.

An "object" *is a collection of data fields that relate to a common entity*, such a resource, source, or a user. As previously defined, a "record" *is a collection of related fields stored on a file*. In E-SRF, a record is an OBJECT. When processing the E-SRF Masterfile, processing is done at the object level. An *object* is what is requested, updated, or printed. How E-SRF internally deals with objects and processing is transparent to you.



As shown below, the RESOURCE and USER segments have multiple object types to further separate the data collected.

The following list briefly describes the information contained within these object types:

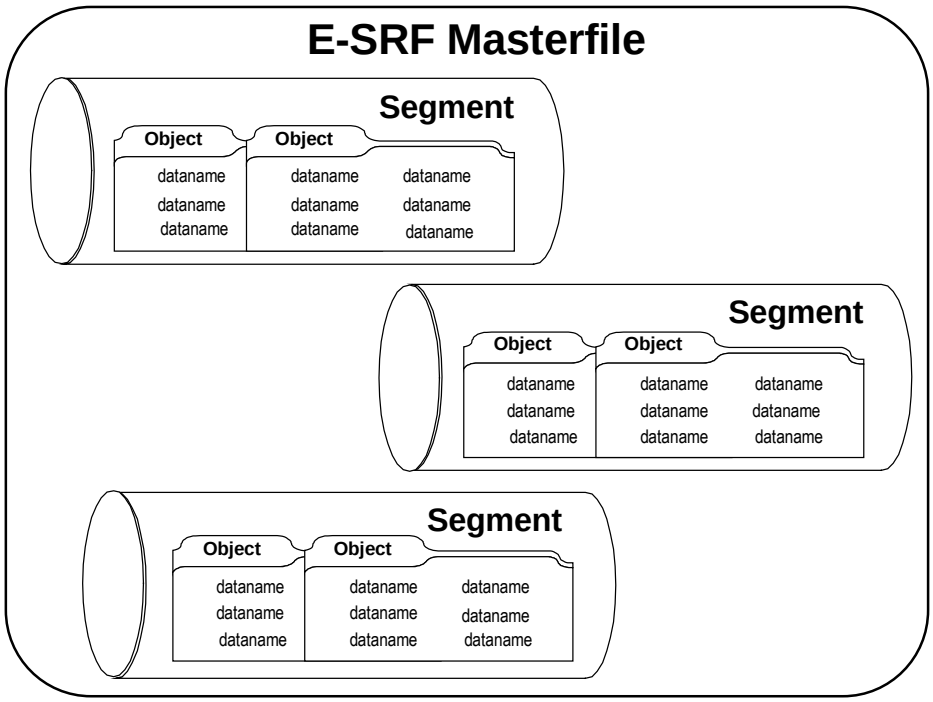
Administration object:	a summary of changes made by a user with security privileges
Chronological object:	a detailed listing of events that have occurred
Firecall object:	events that have occurred because of the ETF/* Firecall Facility
Header object:	information about a user, group, or owner
Maintenance object:	information about changes made to a user or resource
Profile object	RACF connected groups and their attributes.
Statistical Object	daily summary of journalized security events
Recap object:	single element per day statistical summary of all journals
Trace object:	a detailed listing of events that are "traced" by the RSS

Keys

Each object type within a segment is identified to E-SRF by a key. The key includes unique pieces of information that E-SRF can use to look for a particular item in the Masterfile. To you, the key represents a shorthand way of identifying information you are asking E-SRF to include in a report. We will discuss using this information to create reports later in this section.

As you read through this guide, the key symbol, , can be used to quickly identify the key for each object on the E-SRF Masterfile.

Datanames



Each individual piece of information stored within the Segment and Object groupings is referred to as a *Dataname*. For example, the dataname for the eight-character userid is USERID. The dataname for the twenty-character string that identifies a user is: “NAME”.

The E-SRF Masterfile is organized this way so all information is readily available. Reports are produced by identifying *Datanames* contained on the Masterfile’s *Segments* and *Objects*.

This flexibility enables a view of security events from many different perspectives. For example, if you want to see a summary of a particular security event for resources, you request the dataname RESOURCE from the STATISTICAL object contained on the RESOURCE segment of the Masterfile, or, to display the number of security violations, which have occurred for users within resources, you would specify RS.VIOS (violation count) for RS.USERID (userid creating the violations). The information will be presented in the sequence dictated by the segment, in this case class and resource name. *If desired, you may sort the data in other sequences* Dataname Naming Conventions

.

Data Item Types

In addition to knowing what the dataname of an object is, it is helpful to know how the information is stored on the Masterfile. The actual information can be stored as a single data item or array data item.

SINGLE Data Items

Data fields stored in a "linear representation" on a particular E-SRF object are referred to as single data items.

An example of this is a data processing record consisting of several fields (*field1 field2 field3... field20*). These twenty fields make up a single logical record, each field having its own name and purpose.

Record key and other fields	*fieldxxxx1, fieldxxxx2, fieldxxxx3, fieldxxxx4, ….fieldxxxx20*
Object key and other fields	*dataname1, dataname2, dataname3, dataname4, …dataname20*

ARRAY Data Items

A ***LIST*** of several data fields stored in a "linear representation" is called an array. Each occurrence on the list is considered an array element. Individual fields within the array element are called array items. The sequence of fields referred to as an *ARRAY ELEMENT* may be repeated over and over again. Array data items "live" in array elements, array elements "live" in objects.

E-SRF relates the individual array items with their respective datanames within the array element structure.

An example of this concept is a single group of several fields (*field1 field2 field3...)* repeated one or more times in a single data processing data record. E-SRF utilizes this concept when maintaining array element objects on its Masterfile, as shown below:

***Object* key and other fields**

Array Element	Array Items
Event 1	*dataname1, dataname2, dataname3, dataname4, …dataname20*
Event 2	*dataname1, dataname2, dataname3, dataname4, …dataname20*
Event 3	*dataname1, dataname2, dataname3, dataname4, …dataname20*
	. . .
Event *n*	*dataname1, dataname2, dataname3, dataname4, …dataname20*

This page intentionally left blank

Chapter 4: Using Datanames to Identify Information

As previously discussed, the Masterfile is divided up into segments that contain objects that are used to maintain your data.

The Masterfile is a random access VSAM file. It contains a KEY and data associated with the key.

The overall Masterfile key has datanames that are associated to it. These names may be used to reference any object within any segment.

The mapping of the Masterfile key is altered based on the type of segment being processed. The RESOURCE dataname contained on the key may be considered a superset of other datanames when relating to a different segment being processed. In the case of the User Segment, the RESOURCE dataname really contains the USERID and IMAGE of the user.

The naming conventions employed in the data dictionary should help you understand the usage of the dataname by the actual name given to the data field.

KEY FIELDS

These fields stand by themselves, meaning the entire dataname is the name of the data field. These names may be referenced anytime for any object within any segment. The names of these data fields are: SEGMENT, CLASS, RESOURCE, VOLUME and OBJTYPE.

The following fields are not actually stored on the Masterfile, but are *related* to the key.

COMMENT	provides grouping "comment" text for the object.
GROUP	provides the group ID of the group associated with this object.
OWNER	provides the owner ID of the owner associated with this object.

OBJECT KEY FIELDS

These fields redefine the RESOURCE KEY field mentioned above, and stand by themselves. These datanames may be referenced for any object within a specific segment. The names of these data fields are:

Source Segment:

SOURCE	The name of the source the objects in the segment represent.

User Segment:

LOGONID	The userid of the user the objects in the segment represent.
IMAGE	The IMAGE a particular userid belongs to.
USER	The userid of the user the objects in the segment represent.
USERID	The userid of the user the objects in the segment represent.

OBJECT DATA FIELDS

These fields contain your data. The datanames allow you the ability to access them in reporting. The dataname naming conventions allow you to know at a glance the segment and the object the data represents, and the dictionary name of the data itself.

The name is divided up into two character strings separated by a period, similar to the convention used in naming datasets.

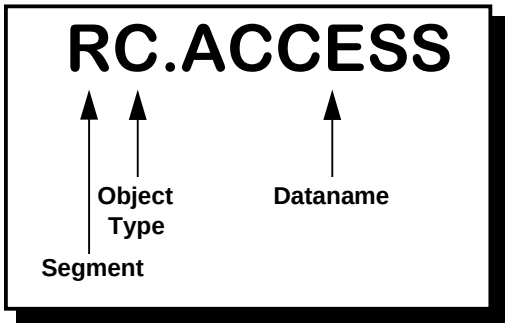
The first string consists of two characters. The first character is the SEGMENT, followed by the second character, which identifies the OBJECT within the segment.

The second string consists of the dictionary name for a specific piece of data.

Using Datanames to Identify Information

To tell E-SRF which particular item of data you want, you identify the Segment and Object type, followed by the Dataname.

For example, to include the type of access that was attempted in a security violation for a resource, the following would be used:



R = Resource Segment, **C** = Chronological Object type, and **ACCESS** = Dataname. If you want to include a piece of information that is part of the *key* for that object, you do not need to specify the Segment and Object identifier to E-SRF. For example, to include the Resource name in a report of violations by resource, you do not specify RC.RESOURCE. You would specify RESOURCE, since E-SRF uses that data item as part of the key for that segment object.

This Reference Manual is organized by the Masterfile Segment and Object Type(s). Included with each dataname is a description of the data and its type (Single or Array). At the beginning of each new Object Type or Segment, the key to the data is identified with the symbol, .

GA	Group	Header	Describes a group
MC	Maintenance	Chronological	Describes maintenance events (Virtual object; made up of RM and UM object data)
FC	Console	Chronological	Describes console events
OA	Owner	Header	Describes an owner
RC	Resource	Chronological	Detail list of events
RM		Maintenance	Changes made to a resource
RR		Recap	Daily recap information
RS		Statistical	Daily Resource statistical by user
SR	Source	Recap	Detail list of events
SU		Chronological	Invalid userid signon attempts
UA	User	Header	Describes a user (RSS)
UB		Sec Admin	Describes administration activity
UC		Chronological	Detail list of events
UF		Firecall	List of events because of Firecall
UM		Maintenance	Changes made to a user
UP		Profile	RACF connect groups.
UR		Recap	Daily recap information
US		Statistical	Daily User statistical by resource
UT		Trace	List of events from RSS “Trace”

Chapter 5: Masterfile Dataname Formats

Introduction

The E-SRF Masterfile is provided to maintain security event data. This data is maintained in Segments and Objects as discussed in the previous chapter. An object typically consists of data relating to a particular security event or defines something involved in security events such a user, a group or owner.

The data itself is considered fields within an object. Fields have characteristics dictating how they are identified and how their format will appear.

This chapter describes the various formats that are contained on the Masterfile's various objects. You can view how each field is classified by running the ESRFDICT report overlay.

Normally, this information is not required to produce good reports. It is documented in this publication due to user request. Having a good understanding of the data formats will enrich your ability to produce better reports.

Data Formats

The following is a list of data formats that are used by Masterfile fields that you may use in your reporting:

ACCESS	Current event's access request

Description: This descriptor identifies what type of access was required for the current event.

ACTION	Action taken by RSS for current event

Description: This descriptor identifies what action was taken based on the current request and the security definitions that relate to the event's final disposition.

BINARY	A number stored in binary format

Description: Binary numbers may be one; two, three or four bytes long and consist of a number stored in the computer's native base two notation.

An example of binary data would be a violation count.

CHARACTER	Free form data text

Description: Character text is just what the name implies, any type of text character, (normally consisting of letters, numbers and punctuation).

An example of character data would be someone's address (1212 WEST 22ND STREET). Notice the data consists of numbers, spaces and letters. The "numbers" do not represent anything other than part of someone's address. The entire text string is considered character data.

COMMENT	External Grouping Comment text

Description: This descriptor identifies the optional comment data that is available for either selection of reporting.

Comment data comes from the E-SRF External Grouping facility. For data to be present, External Grouping must be active, and the proper comment information must be present.

Comment information is determined dynamically based on the External grouping facility and is NOT stored on the Masterfile. This means the comment data may differ based on the External Grouping definitions associated with the current execution.

Comments may be very useful for labeling objects. Consider the hypothetical CICS transaction POC7. This transaction runs a program in CICS that opens a purchase order without management approval. On a report, it may not get the attention is deserves, but if you supply a grouping rule that "labels" this resource as "CREATE P.O. WITHOUT APPROVAL", it may bet some deserved attention.

You can also use the comment data for selection. To extend our example, you could do a SCAN on "WITHOUT APPROVAL' and select all data with this text anywhere in the comment field. The comment field may or may not be displayed on the report.

DATE(A) — Date stored in Absolute Format

Description: Date stored in absolute format. Absolute dates are computed by calculating the number of days that have elapsed from January 1, 1900 until the current date. The result is stored as a two byte unsigned integer. This type of date representation is very useful in selecting dates as well as providing the ability to format the date in whatever format desired.

DATE(P) — Julian Date stored in packed format

Description: Date stored in Julian YYDDD date formatted "packed" into four bytes.

Packed data is a numeric representation where the data is converted from "zoned format" (one number per character) into "packed" format, where two numbers are stored in a single byte with the sign placed at the end of the field.

If you have a Julian date of 99365 (December 31, 1999), which is stored in zoned format as: x'F9F9F3F6F5' (in six bytes), and wish to represent it in packed Julian format, the result would be x'0099365F' in four bytes. The left byte is zeros because only three bytes were needed to pack the field. The "F" at the right of the field is the sign. This particular sign represents "unsigned" values (which are always treated as positive).

An additional Julian date format (referred to as "Lillian") has been developed by IBM to help address some year-2000 issues. The Lillian format makes use of the "unused" high order byte (which normally contains zeros) as a means to identify the century. E-SRF does not use this format because all date information E-SRF maintains is in ABSOLUTE form, (making this unnecessary). It is possible that data maintained in the User Header (provided by Resident Security Systems) use the Lillian convention. E-SRF determines the current century in Julian dates by examining the year. If the year is between zero and forty-nine, it is assumed to be the twenty first century, otherwise the date is assumed to be a twentieth century date.

ENCRYPTED — Encrypted data field

Description: Data stored in this format is encrypted. It was encrypted for a reason and therefore will stay that way.

To encrypt data is to place it in a format that is either impossible (or at the very least very difficult) to determine the useful contents of the original data.

There are two basic types of encryption.

The first is general encryption used to code data in such a format that its contents are useless to anyone else except the intended party. To view the data, you must know the data's encryption "key", which when applied to the data using specialized logic will render the original data. This is used when data must be stored and one cannot be sure the data is completely secure.

The second type is "one-way" encryption. This type of encryption will never allow anyone to be able to obtain the original data. There is no "key" involved. The data is encrypted and that is the end of it. This type of encryption is useful for password authentication. You supply a password to some specialized logic and the result is stored on a password file. When you sign on, what ever you supply as a password is processed the same was and matched to what was stored on the password file.

If you select an encrypted field for display, E-SRF will not attempt to display the data. Instead, the text "ENCRYPTED" will appear. The fields are normally contained on the user header and identified as ENCRYPTED.

ETF-F/CALL — EKC Tool Facility FIRECALL current status

Description: This descriptor identifies the current status (if any) of the ETF FIRECALL involvement in the current event's final disposition.

ETF-STATUS — EKC Tool Facility current status

Description: This descriptor identifies the current status (if any) of the ETF involvement in the current event's final disposition.

EXT-GROUP — External Group Name

Description: This descriptor identifies the field as an EXTERNAL GROUP NAME. This refers to the E-SRF External Grouping Facility assigned group name.

The group name is not stored on the Masterfile. Instead, the group name is derived from presenting the argument the group is representing to the E-SRF External Grouping facility on demand. What this means is the group name is dynamic, based on the External grouping facility's grouping rules associated with the current execution

Group names are from one to six-teen characters in length.

FLAG: *xx* — On-Off Bit-Byte flag

Description: Flags represent a state (YES/NO, UP/DOWN, 0/1, etc.). Flags are used extensively in E-SRF. They can be represented as a single character, or contained in a byte shared with seven other flags.

This descriptor relates to an eight-bit byte containing a possible eight discrete flags (one for each bit within the byte).

The *xx* portion of this descriptor represents which bit within the byte the field is referring to. The bit is shown in hexadecimal notation. 80, 40, 20, 10, 08, 04, 02, 01 (representing the eight bits from left to right).

To someone looking at a report, it makes no difference how the state is stored in the computer, it is either YES or NO. When selecting based on flags, you simply specify the name of the field and whether it is YES or NO. When data is reported, each individual flag specified is a separate column and will be represented by either YES or NO (depending on the setting of the bit).

IMAGE-ID — Security Image ID

Description: This descriptor relates to the eight character IMAGE Id that is assigned to the object. This is an E-SRF maintained field.

For events, the IMAGE is related based on the DOMAIN (or sysid) that hosted the event. Using your parameters, you define what DOMAIN(S) are related to what IMAGE. IMAGE is simply a grouping used to identify and separate based on a set of security databases.

For User Header objects, the IMAGE was established when the security databases were used to SYNCHRONIZE the particular IMAGE.

In either case, all updating is based on the DOMAIN the event occurred in, relating to the IMAGE that DOMAIN belongs to.

INTERNAL — A binary number internally computed

Description: E-SRF provides internally computer fields for user convenience.

An example would be Total Allow Accesses found on the User Statistical Object. This is the sum of all individual allow statistics for the user. These fields are not contained on the Masterfile, and are computed on demand.

Treat these fields as you would any other numeric field.

MAINT — Maintenance type information

Description: This descriptor identifies what was maintained by the maintenance event.

The following list is what will appear:

DATASET	dataset access change
PROFILE	profile control change
RESOURCE	non-dataset access change
SYSTEM	system control option change
USERID	Userid change

MULTIV(*nn*) — Multi-Valued fields

Description: Multi-valued fields were born in ACF2. It was an attempt to provide a more flexible UID string to deal with today's broad access responsibility.

What they are is a named field that contains one or more occurrences of data. For example, you may have a three-character JOB field. This field however can contain up to sixteen discrete three-character job codes. The "array" (or list) of JOB codes is identified by the name, which points to the first entry in the list.

E-SRF makes no use of this facility. However, ACF2 does. It may be used as part of a UID string, or it may simply exist on their Logonid database for reference.

If you select on a multi-valued field, all occurrences are examined.

If you report on a multi-valued field, report services will create a sub-column for the maximum number of entries that may be contained in the list. Caution should be exercised because depending on how wide this field is, it could consume a lot of report real estate. One sub-column for each possible occurrence. Blank occurrences are identified with a + sign in the first byte of the sub-column for ease of reading.

The *nn* represents the maximum number of occurrences that can occur for this field.

NIBBLES — Hexadecimal nibblized data

Description: The data is shown in its hexadecimal nibble format.

What this means is each "nibble" (one half byte) is displayed individually. If your original field were four characters long, the result would be eight characters in length.

Hexadecimal nibbles are the actual hexadecimal value associated to the four bits. A single byte contains two nibbles, the left side of the byte is called the ZONE nibble, and the right side of the byte is called the NUMERIC nibble. This terminology dates back to the "punch card" days. The top three rows were called the ZONE and the bottom 10 rows were called the numeric portion. Other than numbers, it normally required a punch in both the ZONE and NUMERIC rows to represent a character. The letter "A" was represented by a punch in the "12" row (a ZONE row) and another punch in the "1" row (a NUMERIC row) both in the same column.

This way of thinking was carried forward, the same letter "A" is represented by a 1100 0001 (which is a x'C1' in nibblized hexadecimal format. The "C" is in the ZONE nibble, the "1" is in the numeric nibble.

Hexadecimal is base sixteen, this means the numbering scheme is 0 1 2 3 4 5 6 7 8 9 A B C D E F (and after F comes 10). Base 10 contains ten digits, while base sixteen contains sixteen digits. Alphabetical characters were used because mankind only came up with ten digits. Hexadecimal is simply shorthand for binary.

E-SRF itself does not maintain nibblized fields, but they can and do exist on the User Header that is supplied by the Resident Security System

Nibble fields are treated just like they appear. Before they are processed (for selection or display), they are formatted. This allows you to select at the nibble level, and view actual nibblized data. In most cases, the nibblized data would be impossible to select or display because the combination of the ZONE and NUMERIC nibble (which constitutes a byte) would not be represented by any type of graphical character. Consider the byte 0001 1000 (by the way, this is twenty-four in binary). This byte has no graphical character to represent it. It would be possible to figure out its decimal equivalent, but it is much easier to simply consider it as x'18'. Additionally, nibblized data treats each nibble as separate data, allowing you to select based on the particular nibbles that may be desired.

OWNER — Data Owner ID

Description: This descriptor identifies the OWNER that is related to the E-SRF External Grouping Facility group name.

Groups must be defined to the E-SRF Masterfile (in Group Headers). Owners must also be defined in Owner Headers. Owners are "groupings of groups" and become "report targets" in report distribution.

The definition of groups is required mainly to identity the group's OWNER and any "Interested Parties" (other OWNERS who want to see the events). If a group is not defined to the Masterfile, grouping continues, but the data is routed to the DEFAULT OWNER.

The definition of OWNERS is required mainly to provide reports to data owners.

The owner information, like the group name, is not stored on the Masterfile. It is dynamically provided based on the External Grouping facility being active to cause the grouping, and the proper definition of GROUP and OWNER Masterfile Header objects.

PACKED — Packed Decimal Data representation

Description: Numeric data stored in packed decimal format.

Packed data is a numeric representation where the data is converted from "zoned format" (one number per character) into "packed" format, where two numbers are stored in a single byte with the sign placed at the end of the field.

For example, the zoned number 1234 appears in hexadecimal as x'F1F2F3F4'. It takes four bytes to contain the data; each number is represented in its EBCDIC graphical representation. This is referred to as UNKACKED numeric data, and to the computer, it is not a number at all, just characters. To do any computations with this data, the data must be at least packed. Packing the four bytes into another four bytes will result in x'0001234F'. You should notice the difference. The sign was placed in the end nibble, the actual numbers had their ZONE nibble stripped off and were placed right justified padded with necessary zeros. Now the data is considered to be a numeric value and can be considered for numeric operations.

Most data on the E-SRF Masterfile is BINARY, which is the next step (you notice the packed data is decimal (the reason it is called Packed Decimal)? Converting it to binary would render x'000004D2', yet another numeric format.

There are some Packed Decimal fields carried on the Masterfile. Treat them as you would any numeric data. On selection and formatted on reports, you will not be able to tell the difference as to whether the data was PACKED or Binary in its original form.

REASON	**Current event's disposition reason code**

Description: This descriptor identifies what the reason was for the event's final disposition.

RECTYPE	**Event "RECTYPE" as it relates to External Grouping Facility**

Description: This descriptor shows how the event was classified for presentation to the E-SRF External grouping facility. The use of RECTYPE allows more granular grouping, such grouping Violations to one group and Access Logs to another.

RESOURCE	**Resource Masterfile key field**

Description: This descriptor identifies the use of the RESOURCE Masterfile key field.

The formatting is dependent of the Masterfile segment being processed and is provided to make reports easier to read.

For example, the RESOURCE key field for a User Segment object consists of USERID and IMAGE. These two fields occupy the first sixteen characters of the field, with the remaining data blank. E-SRF will format the field with the userid, followed by the IMAGE (in parentheses).

SYSTEM	**Resident Security System type**

Description: This descriptor identifies the particular Resident Security System, which the event is related to,

TOD-STCK	**Time of day stored in hardware Store Clock format**

Description: Date and time information stored in the format the hardware uses to relate time. This is the most exact means of representing the date and time of an event.

This value comes from the hardware STCK (store clock) instruction. Its value is maintained on the hardware and presented as a *doubleword* (eight bytes of storage) on demand.

Data in this format is not very useful. The value must be converted into date and time information that can be used for selection and reporting. It will be reported as DATE and TIME together in one column.

Treat this as any date and time field.

TOD-.01	**Time of day stored in hardware Store Clock format**

Description: Time information stored in the format the hardware uses to relate time. This is the most exact means of representing the time of an event.

This value comes from the hardware STCK (store clock) instruction. Its value is maintained on the hardware and presented as a *doubleword* (eight bytes of storage) on demand. This particular format looses the high order four bytes, thus only leaving time in .01 seconds.

Data in this format is not very useful. The value must be converted into time information that can be used for selection and reporting.

Treat this as any time field.

TOKEN	**Masterfile data TOKEN**

Description: Certain data on the Masterfile has been "tokenized" to afford us very large data field processing and conserves space on the Masterfile.

A token is a very short binary value assigned to represent the actual data. The data itself is stored in the Resource Object Token Dictionary, which is currently maintained in the Control Segment of the Masterfile. Individual events reference this data using it's token.

Normally you will never see tokens unless you specify the actual dataname that represents the token instead of the real dataname for the field.

Data in this format is not very useful and would normally not be shown on reports

UID-CHAR	**Universal IDentifier character string**

Description: E-SRF maintains a Universal IDentifier string for each user. This field is currently twenty-four characters and is that way because that is what was required to store the largest character string.

In ACF2, the E-SRF Universal IDentifier is a direct mapping of the ACF2 UID string.

In RACF, the E-SRF Universal Identifier is the concatenation of the RACF OWNER, Default GROUP and the USERID.

It is ok to refer to this field as the UID. It is used much the same way as ACF2 used its UID string and really does not cause too much confusion.

This field is formatted like any normal character field except all spaces are substituted with *periods* (.). This is because UID information is normally a composite of other fields, and these fields may be omitted or partially filled in. To eliminate confusion and to make the reports more readable, the periods are included. This is for reporting only. When making selections, do not use the periods.

Special Note *Please review the following information about UID strings:*

UID data at the event level was introduced in product release 1.6. This information is now provided in RC.UID and UC.UID, (as well as UA.UID) Masterfile objects.

UID data in event data objects represent the UID string at the time of the logging. If you have the ability to alter the UID string and revalidate requests, you will end up with the UID allowing access for event loggings and the last UID tested for violations.

If you have UID selections and reporting for event data, you may want to alter them from UA.UID to whatever event object you are reporting on. This will provide more accurate selection and reporting.

UID data in the User Header represent what the current UID is at the time of the last KNOWN update. This may or may not be the same as what is shown in event data.

If Multi-Valued fields are used in the UID string, the UID shown will be constructed using the first occurrence of the data in the field list. This constructed UID string is what is used for selection and reporting.

If you want to make selections against the full range of Multi-valued fields, you will have to apply your search against the actual Multi-Valued field itself.

If you want to report on all possibilities, you will have to select the Multi_valued field for reporting.

Please refer to the Dataname MULTIV(*nn*) described above for more information about Multi-Valued

Chapter 6: Data Dictionary Structure

System Dictionary

E-SRF contains a *SYSTEM DICTIONARY, which* is placed on the Masterfile when the system is initialized. This dictionary contains datanames common to all Security Images on the Masterfile.

Image Dictionaries

When a particular Security Image is configured to E-SRF, a separate *IMAGE DICTIONARY* is placed on the Masterfile. This dictionary contains datanames that relate to the *IMAGE* assigned to it. The datanames normally correspond to the USER HEADER object because this data is entirely related to a particular RSS database being presented to E-SRF. Note that a single image is normally a set of all domains (sysids) being serviced by a common set of Resident Security System databases.

More than one image may be contained on a single Masterfile. Each has its own image dictionary and it may or may not contain the same datanames as they relate to the associated RSS databases. This structure allows you to define multiple security images to a single Masterfile, even across Resident Security Systems. A single userid may be represented across multiple security images but does not have to be the same individual.

E-SRF's reference to a particular dataname in an IMAGE dictionary is transparent to you. You specify the dataname and E-SRF relates to the proper IMAGE and location of the data.

NOTE: In this release, all data referenced by the same dataname across images must be in the same format and location. This will change in a future release.

This Reference manual describes the information related to security events. It does not attempt to describe the information contained in the Resident Security System datanames, which are located in the USER segment of the Masterfile. The ESRFDICT report overlay prints a listing of all datanames in the current E-SRF Masterfile, including those from the Resident Security System. See the *Report Overlays Guide* for instructions to produce that listing.

Putting It Together

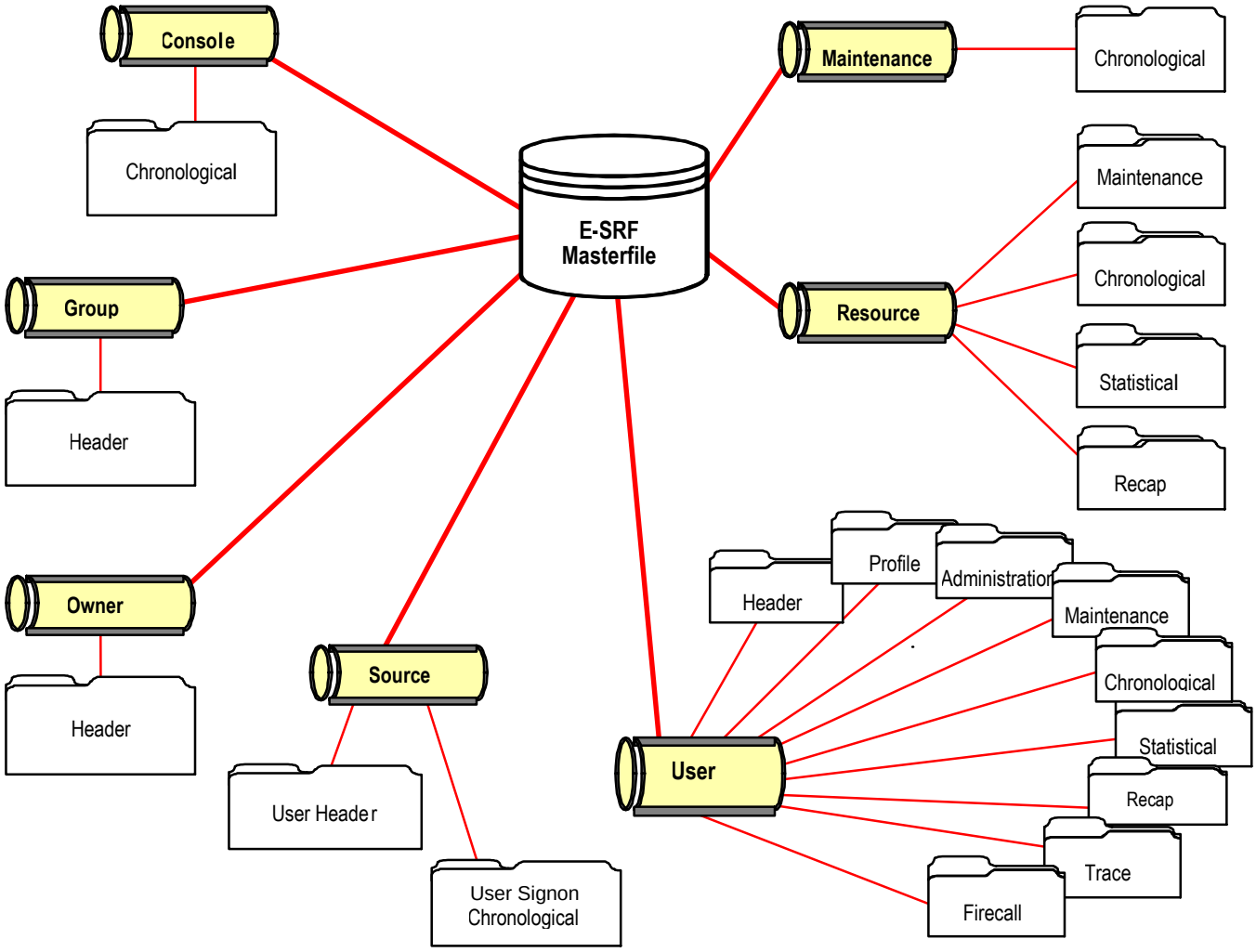
When you initialize an E-SRF Masterfile, E-SRF creates a System Dictionary. When you configure one or more security images, E-SRF creates an Image Dictionary for each image that you configure. The System Dictionary and Image Dictionaries make up the *E-SRF Data Dictionary.*

Run the ESRFDICT Report Overlay to see a listing of the System Dictionary and all associated Image Dictionaries. For more information about running the ESRFDICT Report Overlay, see the *Report Overlays Guide.*

This page intentionally left blank

Chapter 7: Masterfile Structure

The E-SRF Masterfile is divided into Segments that are further divided into Objects.



Masterfile Segments and Object Descriptions

Below are a listing of each segment, a brief description, and a list of the object types available.

Control Segment

The Control Segment is used to internally maintain control over the Masterfile and all E-SRF processing options. This segment is not available to the user through the data dictionary. No security event data exists in the Control Segment that would be directly referenced by the user.

OBJECT TYPES: Control

Console Segment

All logged console activity captured by the Resident Security System is stored in this segment.

OBJECT TYPES: Chronological

Group Segment

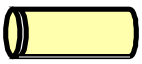
This segment contains a single object type that identifies GROUPS, as viewed by E-SRF. To use E-SRF automatic report distribution, the target groups must be defined in this segment.

OBJECT TYPES: Header

Owner Segment

This segment contains a single object type that identifies OWNERS, as viewed by E-SRF. E-SRF automatic report distribution routes the reports to the identified owners and interested parties. The target owners must be defined in this segment.

OBJECT TYPES: Header



Maintenance Segment

The Maintenance Segment is a Virtual Segment.

There is no physical appearance of this segment on the Masterfile. The segment consists of a single object MC (Maintenance Chronological). This object may be referenced as if it exists on the Masterfile. The data is assembled for you from the RM (Resource Maintenance) and the UM (User Maintenance). You use the normal MC datanames. Processing this object provides a seamless report showing maintenance to both Resource and Userids.

OBJECT TYPES: Chronological

Resource Segment

This segment contains objects that describe “things”, that is the various resources the Resident Security System is attempting to protect. Datasets are considered a type of resource

OBJECT TYPES: Chronological
Maintenance
Recap
Statistical Summary

Source Segment

This segment contains objects that describe “places” where access is being attempted.

OBJECT TYPES: Recap
Invalid userid chronological

User Segment

This segment contains objects that describe “people” who are making access requests.

OBJECT TYPES: Header
Administration
Chronological
ETF/* Firecall
Maintenance
Profile
Recap
Statistical Summary
Trace Chronological

This page intentionally left blank

Chapter 8: Masterfile Physical Characteristics and Size

Purpose of this chapter

This topic describes the E-SRF Event Reporting Masterfile and how to determine its physical VSAM characteristics. The information contained in this chapter is related to how this product interacts with VSAM (Virtual Storage Access Method), which is the MVS systems support required to store your Masterfile on the physical disk drives, and the Event Reporting System.

This chapter is designed to help you understand how to maintain your Masterfile on your computer system's disk storage.

What the Masterfile is

The Masterfile contains all the data collected about security events that take place in your system. E-SRF uses the data in the Masterfile to create reports. This chapter will give you information about what is carried on the Masterfile, the types of records stored in the Masterfile, how long much space these records will take, and a formula for determining how big an E-SRF Masterfile should be.

E-SRF collects information about security events that take place in your system.

The details about security-related events, such as who attempted access, what data they were accessing, the date and time of the access attempt, etc., is collected, "normalized" into a relational format, and stored in a database called the E-SRF Masterfile. Additional information that includes Resident Security System definitions of your users and other E-SRF control information are also contained on your Masterfile.

Once the information has been collected and applied on the Masterfile, the E-SRF reporting components can use this data to produce reports.

File characteristics

The Event Reporting System maintains its "database" in a VSAM cluster referred to as the "*Masterfile*". The Masterfile maintains its own file system structure used throughout the Event Reporting system. This file system is a logical system, fully compatible with VSAM, and uses VSAM as its external Input/Output interface. It follows all the rules of a normal VSAM cluster, as they would be followed by any classic VSAM application.

The Masterfile is a VSAM KSDS Cluster. It is processed both sequentially and randomly. Within E-SRF, the entire contents of the Masterfile may be CACHED to provide rapid data retrieval, or may be processed directly by VSAM. The Masterfile is defined as a VSAM cluster using the IDCAMS utility. This process was initially completed during your installation of E-SRF. Please refer to information in this charter for more information on this process.

Masterfile Objects

A logical record (or set of information in E-SRF) is referred to as an "*object*". An object has a key and associated data. It is "read" by E-SRF components as a single entity. E-SRF file management functions will read whatever VSAM record(s) may be required to acquire the E-SRF object. Object may be viewed as logical records.

Masterfile Segments

A logical group of objects containing data about a single category of information is referred to as a "*segment*". For example, all USERID data is contained in the USER segment. Segments may be viewed as "little files within a larger file".

Object Characteristics

In E-SRF, an object may be as long as sixteen million characters. Because VSAM will not let us store a sixteen million-character "record", the actual data contained on the cluster consists of 8,294 character "*record segments*". One or more of these "*record segments*" comprises a single E-SRF object in compressed form. The term "*record segment*" here refers to a physical record on VSAM, which is not the same as the grouping of objects, also referred to as a "segment". VASM record segments are internal structures and with rare exception are never discussed in E-SRF publications.

A single E-SRF object may span one or more VSAM record segments. Additionally, before the object is written out to VSAM, the object may be is compressed. Compression may render variable results depending on the data being compressed. We will stay conservative and assume a two to one compression factor in our discussions. Other than attempting to figure out how many actual VSAM records make up your cluster for space determination, there is no need to concern yourself with VSAM record segments.

Defining the VSAM Cluster

Defining a VSAM Cluster is a straightforward procedure except for one thing ... determining how big to make it? With this product, this is a bigger problem solving opportunity than in most other systems. It is difficult to determine how big to make the cluster without understanding how the data appears, and how much data will be stored on the cluster.

The following IDCAMS commands are supplied in the User's Guide for defining the Masterfile VSAM cluster, and may be used as a sample for your review:

Please read the remainder of this chapter before defining your Masterfile VSAM Cluster.

```
DEFINE CLUSTER -
       (  NAME(ESRF.MASTER) -
          OWNER(ESRF) -
          VOLUME(XXXXXX) -
          SHAREOPTIONS(2 3) -
          SPEED -
          REUSE ) -
       DATA -
       (  NAME(ESRF.MASTER.DATA) -
          KEYS(62 0) -
          SPANNED -
          RECORDS(250000 10000) -
          RECORDSIZE(2048,8192) -
          FREESPACE(0,0) ) -
       INDEX -
       (  NAME(ESRF.MASTER.INDEX) )
```

VSAM "FREESPACE" considerations

VSAM provides a means to include FREESPACE at the Control Interval and Control Area level. FREESPACE is unused space that you engineer into your cluster in anticipation of adding and lengthening the data contained in your cluster. Normally this parameter must be carefully tuned to provide performance when adding and updating data on your cluster. Improper settings could cause adverse disk I/O bottlenecks.

The E-SRF Masterfile *does not require this parameter* as long as you use the CACHE when updating the Masterfile. Yes, you can set it to zero and be happy about it. In this product, as long as the CACHE is used during the Update Function, setting FREESPACE to zero will improve performance and conserve disk space.

More about this later in this chapter

Size of the E-SRF Masterfile

Determining what size to make your Masterfile may be difficult because E-SRF stores data in compressed form, and because you do not yet know the type of information that will be stored.

The best-known approach may not be the most scientific or intellectual method available.

Using the example below, you may be able to "*guess*" how big to make it and adjust after your activity stabilizes. Before discounting this approach, read the remaining material and you may agree.

The following is a brief discussion of each object and how much data may be contained on each. Please note, there are two types of objects stored on the Masterfile: *Header* objects that simply contain data and *Array Element* objects, which contain "lists" of data.

A header object is the simplest form of data, just a bunch of fixed fields placed on the object one after the other. This is a common approach and is used in most file data structures.

An array element is more complex. You have some fixed fields, followed by a "list" of fields. The items in the "list" are typically single events that were logged by your system. Each element in the list consists of fields that describe the event. You have a set amount of fixed fields. You also have a second set of fields on the same object, (*which is another group of fixed fields*), but are repeated from 1 to 32,767 times. The amount of space required for array element objects is dictated by the number of items in the list. The depth of the list is normally constrained by the ELEMENTS system parameter. Additionally, the RETAIN value associated for an object sets a limit on how old events may be retained on the Masterfile. Additionally, the object may be smaller simply because the particular object may not be filled to its capacity with events. Perhaps it is only referenced several times a day.

If you want to be very safe in your estimates, assume that each array element object will be full, that is the object will contain the number of elements specified by the ELEMENTS system parameter. You will probably end up with a very large cluster, but you can make it smaller later.

"Out of the box", this product ships its sample parameters with ELEMENTS(6144) set. This number can be as high as ELEMENTS(32767). Retain values are set for each object individually using the RETAIN parameter. These settings are conservative, but they can be as high as DAYS(4095).

You can eliminate objects from the Masterfile that you do not use in reporting by setting DAYS(0) for the unwanted objects. It is recommended that you do not do this initially, but go back and review these settings later and make the necessary adjustments.

Once you gain experience with this product, you will better understand your needs, how long to save particular data and how this product uses VSAM, allowing you to make better choices. In the beginning, you simply want to get this up and running.

Every object may contain one or more actual VSAM records. A VSAM record is 8192 bytes long. You must assume there will be at least one VSAM record for each object. If the object is longer than 8192 bytes, additional VSAM records will be present to store the object. The overhead in each VSAM record includes the VSAM record key (62 bytes). The remainder of the record is considered payload data. If the payload and overhead is less than 8192 bytes long, the shorter length will be used. This means if we only need 300 bytes to store an object, that is all that will be used.

You are estimating VSAM space. This means how many 8192 byte records you have to provide space for to contain the Masterfile objects you will create as you apply the log data from your security systems.

We will briefly describe what is in each object (payload data) and an opinion on how to estimate space for it.

Masterfile Impact from Security Events

E-SRF applies update to the Masterfile by reading in Resident Security System journal data, normalizing the data into a level universal with the entire E-SRF system, and updates the appropriate Masterfile objects.

Updating is NOT one to one. Consider the events that take place for the following two events:

If a userid is added or changed, the following occurs:

The target User Header Object is updated (added if not present).

The target user's Maintenance Object is updated (added if not present) reflecting the user's old and new information.

The changer's Administration Object is updated (added if not present) reflecting the administration activity to the target user.

If a security event logging occurs:

The target resource CHRONOLOGICAL, STATISTICAL, and RECAP objects are added if necessary and upgraded reflecting the event.

The target resource VIOLATION object is added if necessary and upgraded if the logging was the result of a violation.

The requesting user's CHRONOLOGICAL, STATISTICAL and RECAP objects are added if necessary and upgraded reflecting the event

Control Segment space estimates

The control segment contains information that is required to run the E-SRF Event Reporting system. Information, such as all of your installation options, the System Data Dictionary, a Data Dictionary for each Defined IMAGE, your DOMAIN assignments, and the entire Object Token Resolution Dictionary are stored in this segment.

Master Control object:

Objects contained in this segment are not mapped in your data Dictionary. The data on this segment is used to control the processing of this system. This is a critical segment.

The Master Control object is used to store all of your settings and contains information needed to run E-SRF in your organization. The length of this object is 1024 bytes and requires one short VSAM record.

Update Control Object:

The UPDATE control object maintains information about your use of the Update Function and helps prevent you from updating the system more than once with the same journal dataset. The object is 8124 bytes long and requires one VSAM record.

Image objects:

There will be an IMAGE control object for the SYSTEM image and one for each "IMAGE" you define to E-SRF. This array element object contains 600 bytes of fixed fields plus a 64-byte element for each field contained in its data Dictionary.

The SYSTEM image contains data dictionary elements that describe every accessible field on the Masterfile that is common to all images and is established when you initialize your Masterfile.

The individual IMAGES you define describe particular images in your organization and contain dictionaries that you established that relate to particular images. These objects are built during the CONFIGURATIOIN process. This is how E-SRF "learns" about what security systems you have and how they are configured. Currently, the only data dictionary structures that are image dependent are the Userid fields found on the User Header (UA) object. This object provides the User header userid fields found in your data dictionary.

The SYSTEM IMAGE currently requires four VSAM records.

A typical user specified IMAGE requires two VSAM records.

Domain objects:

This single object is used to map all of your DOMAINS into the objects you have defined on the Masterfile. This tells E-SRF what to do with data from a particular DOMAIN (a domain is the same thing as a SYSID). This object is build and maintained by the ASSIGN statements you provide. For example, you have a production system called NY01, it belongs to the NEWYORK image. This is where thin information is stored.

Typically, unless you have a huge installation with many images and domains, this object should fit nicely on one VSAM record.

Token Resolution Dictionary objects:

Starting in Release 2.1, many Masterfile "names", such as resource and profile names have been tokenized. A token is a very short (*three bytes in release 2.1*) representation of a piece of data that may be quite larger. In E-SRF, a tokenized name may be from 1 to 1000 characters long'. This allows E-SRF Event Reporting to support very long resource names. On the particular objects that reference the resource name, only the token is stored. The actual resource name text is stored in the Token Dictionary contained on this object.

There will be a single VSAM record for each tokenized names. This means if you have 15,000 tokenized names, there will be 15,000 tokenized objects stored in the control segment, each occupying a single object.

Although the maximum name length is 1000 characters long most, if not all of yours will be quite smaller. If you want to really be safe, assume each VSAM record will be 256-characters long.

What are tokenized as of this writing are: Resource names, Profile names, UID data and your grouping structure (if you have one).

Console Segment space estimates

This segment only has a single object type; the Console Chronological

The Console Chronological (FC) Object is an array element object that contains an element for each event that was considered a "console" event by your security system. Sometimes these events are logged on the Operator's Console. Each element is 88 characters long.

There will be a single object for EACH DOMAIN that console data has been detected for. If you have five domains, you can probably expect five Console Chronological objects over the course of processing.

If you totally fill this object with data, the object payload would be just shy of 3 million bytes (uncompressed). This object is normally compressed, so a full object would be around 1.3 million. Depending on compression, this object could require several hundred VSAM records to contain it.

You can account for this, but this is only if you intend to keep (*or even ever process*) 32,767 security console events in each domain.

Group Segment space estimates

The Group Segment contains a single object; the Group Header (GA). A GA object will exist for each group you define to E-SRF Event Reporting.

The size of a GA object is 512 characters (*uncompressed*) and each will require a single short VSAM record.

Maintenance Segment space estimates

The maintenance Segment contains a single object; the Maintenance Chronological (MC) object. A MC object will logically exist for each maintenance event that occurred for any resource or userid.

This segment is VIRTUAL. You can access all fields contained on the segment using the Data Dictionary, but the segment itself does not physically exist on the Masterfile. The data requests are resolved from other objects (specifically the RM and UM objects).

This means there are no space requirements for this segment.

Owner Segment space estimates

The Owner Segment contains a single object; the Owner Header (OA). An OA object will exist for each owner you define to E-SRF Event Reporting.

The size of an OA object is around 350 characters (*uncompressed*) and each will require a single short VSAM record.

Resource Segment space estimates

The Resource Segment is normally the second largest segment on the Masterfile. It contains a series of objects for each resource that was journalized by your security system. Each object is an array element and some may become quite large.

Resource Chronological object:

This object contains a list of all logged activity against a resource for the retention windows specified by the RETAIN parameter. Each element is 61 bytes long. As of Release 2.1, most of the large data areas have been tokenized. Because of this, it is not recommended to compress this object. Doing so would not yield much savings and introduce additional processing overhead.

If you totally fill this object, with ELEMENTS(32767), the object payload would be around 2.4 million bytes. Most of the time, retention on this object is low, meaning several days. How much data piles up on it depends on how much logging you do. Typically, you will have some resources that may fill an object, but most of the time, you will not. If you happen to fill one, it will take around 250 VSAM records to contain it.

Resource Maintenance object:

This object contains a list of all logged maintenance activity against resource definitions made to your security systems for the retention window specified by the RETAIN parameter. Each element is 89 characters long.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around 3 million bytes (*uncompressed*). This object is normally compressed, so a full object would be around 1.3 million. Depending on compression, this object could require several hundred VSAM records to contain it.

You can account for this, but this is only if you intend to keep (*or even ever process*) 32,767 maintenance items for a single resource within the retention windows established. More typically, one VSAM record per resource is a good base to start with.

Resource Recap object:

This object contains a list of logged activity summaries. One for each calendar day that loggings took place. This means the most elements you would accumulate in a week, assuming you had logged activity for the resource every day would be seven. Each element is 44 characters long.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around 3 million bytes (*uncompressed*). This object is normally compressed, so a full object would be around 1.3 million. Depending on compression, this object could require several hundred VSAM records to contain it.

It would take you over 80 years to fill this object to its capacity. You can account for this, but typically, one VSAM record per resource should cover this object.

Resource Statistical object:

This object contains a list of logged activity summaries just as the RR object does, except it contains an element for each USERID that initiated the event. This makes this object harder to estimate. Each element is 44 characters long.

It can get quite large if your security system logs a resource that is used by a large compliment of users. If 500 users accessed a single resource, thousands of times each, you would wind up with 500 elements for that particular day.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around 1.5 million bytes. This object is normally NOT compressed, so a full object would be around 1.5 million bytes long, making a full object span almost 200 VSAM records.

Source Segment space estimates

The Source Segment is provided to maintain event data relating to SOURCES. Sources are names of "places" and are presented to us on the security system event log record. The actual source names are determined by the security system.

Most of the time people do not run with these objects active. If you do not, then you have RETAIN settings set as such and you do not have to account for these objects.

Source Recap object:

This object contains a list of logged activity summaries. One for each calendar day that loggings took place. This means the most elements you would accumulate in a week, assuming you had logged activity for the resource every day would be seven. Each element is 104 characters long.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around 3.5 million bytes (*uncompressed*). This object is normally NOT compressed. It could require over 400 VSAM records to contain it.

It would take you over 80 years to fill this object to its capacity. You can account for this, but typically, one (or two at the most) VSAM record per resource should cover this object.

Source Userid object:

This object is used to track signon activity at the source level, which includes signon failures due to invalid userid. This is the only object that tracks invalid userid signon failures.

The only time anything is posted to this object is when it is a signon and it fails. Each element is 22 characters long.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around .75 million bytes. This object is normally NOT compressed. It could require over 90 VSAM records to contain it.

To fill one of these objects you would have to have a considerable number of signon failures. A good starting point would be one VSAM record for each source where signon activity originates

User Segment space estimates

The Resource Segment is normally the largest segment on the Masterfile. It contains a series of objects for each user that is defined to each security system IMAGE. Every object except the User header is an array element object.

Its characteristics are very similar to the Resource Segment except there is potential to maintain more data. Logged events may be carried on the Resource Chronological object, but given the same window specifications on User Segment objects, you may end up with more data in the User Segment simply because users are individually maintained at the IMAGE level, where Resource Objects are not. The userid may actually be viewed as the USERID + the IMAGE. For example, if the userid: JOE is in two images, Joe will have a set of objects in both images, quite capable of maintaining data from DOMAINS assigned to those images.

User Header object:

The User header (UA) object is the foundation object for a userid in the E-SRF Event Reporting System. In normal processing, a UA object will exist for every userid that exists in each security IMAGE that Event Reporting is maintaining data and reporting on. The normal process is to run a SYNCHRONIZE to initially build the data as it exists on a particular IMAGE's security system. After that has been completed, let E-SRF maintain this object through its Update Function.

The bottom line is you will need an object for each user defined in each IMAGE that is being processed by Event Reporting.

These records are normally (and should be) compressed. Each Resident Security System (RSS) has its own format of the data carried in this object. Typically, after compression, the object should easily be contained on a single VSAM record.

Consider this to be a one to one relationship between the number of UA objects contained on the Masterfile and the number of VSAM records needed to contain them.

User Security Administration (maintenance) object:

The User Security Administration (UB) object contains a list of all logged maintenance activity by the user who made the changes. This object can be viewed as a list of what each security administrator did to the security system that was logged during the retention window.

These objects will only be present if the user associated with it issued maintenance commands in the security system. Normally, this means you will only have UB objects for security administrators. This object is an array element and each element is 56 bytes long.

If you totally fill this object with data, the object payload would be somewhere around 2 million bytes. This object is normally NOT compressed, so a full object would be around 2 million bytes long, making a full object span almost 250 VSAM records.

A filled object implies you have your ELEMENTS(32767) set and actually have that many events stored. More realistically, if you have very busy security administrators, and are retaining this data for a long time period (*which is recommended*), and you have ELEMENTS set to the maximum setting, accounting for 100 VSAM records per security administrator per IMAGE would be a good starting point. It may take time for this object to fill up.

User Chronological object:

The User Chronological (UC) object contains a list of all logged activity against the userid (within a particular IMAGE) for the retention windows specified by the RETAIN parameter. Each element is 64 bytes long. As of Release 2.1, most of the large data areas have been tokenized. Because of this, it is not recommended to compress this object. Doing so would not yield much savings and introduce additional processing overhead.

If you totally fill this object, with ELEMENTS(32767), the object payload would be around 2.6 million bytes. Most of the time, retention on this object is low, meaning several days. How much data piles up on it depends on how much logging you do. Typically, you will have some resources that may fill an object, but most of the time, you will not. If you happen to fill one, it will take over 250 VSAM records to contain it.

User Firecall object:

The User Firecall (UF) object contains a list of all logged FIRECALL (EKC ETF/x Firecall Facility) activity against the userid (within a particular IMAGE) for the retention windows specified by the RETAIN parameter. Each element is 158 bytes long.

If you totally fill this object, with ELEMENTS(32767), the object payload would be over 5 million bytes. This is the largest object within the E-=SDRF Masterfile, but due to its purpose, one of the least populated objects. The presence and the amount of data stored on this object depends on how much the FIRECALL facility is used. If you happen to fill one, it will take over 600 VSAM records to contain it.

Normally, only a small subset of users are involved with Firecall.

User Maintenance object:

This object contains a list of all logged maintenance activity against userid definitions made to your security systems for the retention window specified by the RETAIN parameter. Each element is 89 characters long.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around 3 million bytes (*uncompressed*). This object is normally compressed, so a full object would be around 1.3 million. Depending on compression, this object could require several hundred VSAM records to contain it.

You can account for this, but this is only if you intend to keep (*or even ever process*) 32,767 maintenance items for a single resource within the retention windows established. More typically, one VSAM record per resource is a good base to start with.

User Profile object:

Currently, the User Profile (UP) object will only exist for users who belong to IMAGES that are secured by RACF. Consideration for this object is not required for IMAGES not secured with RACF.

This object contains a list of all RACF groups that are connected to the user. Each element is 136 characters

This object will likely never be filled up. You can store 60 elements on a single VSAM record.

You can account for this object by assuming only one VSAM record would be required for each user in each RACF image that is defined to the Masterfile.

User Recap object:

The User Recap (UR) object contains a list of logged activity summaries. One for each calendar day that loggings took place. This means the most elements you would accumulate in a week, assuming you had logged activity for the resource every day would be seven. Each element is 106 bytes long.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around 3.5 million bytes (*uncompressed*). This object is normally compressed, so a full object would be around 1.8 million. Depending on compression, this object could require over 400 VSAM records to contain it.

It would take you over 80 years to fill this object to its capacity. You can account for this, but typically, one or two VSAM record per user should cover the requirements for this object.

User Statistical object:

The User Statistical (US) object contains a list of logged activity summaries just as the RR object does, except it contains an element for each RESOURCE that was related to the event. This makes this object harder to estimate. Each element is 39 characters long.

It can get quite large if your security system logs many resources that a particular user has access to. For example, if a single user had access to 500 resource thousands of times each, you would wind up with 500 elements for that particular day.

If you totally fill this object, with ELEMENTS(32767), the object payload would be somewhere around 1.4 million bytes. This object is normally NOT compressed, so a full object would be around 1.4 million bytes long, making a full object span almost 200 VSAM records.

User Trace object:

The User Trace (UT) object is identical to the User Chronological object.

Its data source is trace records from ACF2 Images. When user trace is on, the logging information is stored in this object. Normally this object is run with a RETAIN DAYS(0) set. If this is the case, no UT objects will be maintained on the Masterfile. If you want this information the Masterfile, use the same approach as you would for the UC object, except understand the source of the data will be only from the users whom you have trace turned on for.

ESTIMATING the number of VSAM records in your Masterfile

The Masterfile is stored on VSAM. Some of the objects are in a COMPRESSED format

The records stored in VSAM are used to build Masterfile OBJECTS.

Records on VSAM are variable length and are 8,192 bytes in size, (if the full VSAM record is required to contain the required data). If less than the maximum VSAM record length is needed to store data, the VSAM records will be trimmed to the required length.

If an object is 12,000 bytes long, it will take two VSAM records to contain it. One full 8,192-byte record, with the residual 4000 bytes contained on a second, smaller record. As the object grows, the second VSAM record grows until it fills up, resulting in the possibility of an additional third record to be associated with the current object.

It is almost impossible to mathematically compute this file size with any true accuracy without extensive knowledge of your data. Your data changes every day. You are in complete control of your data from this product's point of view.

An example to get you started:

The following is an example of how you may attempt to estimate the number of records and size required to contain a particular E-SRF Masterfile on VSAM. This is only an example; **yours will vary**.

Using ACF2 as an example, we will assume the following configuration:

5,000 userids
3,000 ACF2 rule sets
2 LPARS (DOMAINS), both residing on a single Security IMAGE
3,000 sources (point of entry)
200 sources that had invalid userid attempts.
8 resource owners and thirty resource groups
50,000 resource names that may be logged, with violations spread over 5,000 resources
6 security administrators
1000 "targets" of security administration.

Objects	Purpose
50,004	E-SRF Control Segment (SINGLE)
1	Security IMAGE definition
8	Resource OWNERS defined to E-SRF (SINGLE)
30	Resource GROUPS defined to E-SRF (SINGLE)
2	CONSOLE Segment (ARRAY), one for each DOMAIN
150,000	RESOURCE Segment (Logged events for 50,000 resources) (ARRAY)
5,000	RESOURCE Segment (Resources with one or more violations) (ARRAY)
3,000	SOURCES that hosted logged security events (ARRAY)
200	Sources which hosted invalid userid attempts
5,000	Unique USERS contained in Resident Security System (SINGLE)
15,000	Unique USERS with loggings and/or violations (ARRAY)
8	Unique USERS administered the Resident Security System (ARRAY)
2000	Administration targets. 1000 in the MC, 1000 split between users and resources.
230,253	**TOTAL NUMBER OF OBJECTS REQUIRED TO STORE EXAMPLE**

This tells us the maximum number of objects required. If each object takes a single VSAM 8,192 character record, you would specify enough space to hold 230,253 individual VSAM records. This would be a good place to start

Impact of the Resource Object Token Dictionary

- Long character strings, such as resource names are tokenized when stored on the Masterfile. To provide this data to you on reports, the tokens used must be resolved. When a new resource is added to the Masterfile, a small binary token is assigned to represent the resource name text. The token and the full text it represents are stored in the Resource Object Token Dictionary. This dictionary is currently maintained in the Masterfile's Control Segment.

- As you may notice, there is an object contained in the control segment for EACH resource residing on the Masterfile, which are the Resource Object Token Dictionary entries for the resources contained in the resource segment. Their length is 62 (object VSAM key) bytes followed by the length of the data being tokenized. Resource names are short (*when considering a 8,192 byte record*). If you consider a 44-character resource name, you could think of each of these records being around 106 bytes long. If you have several 1000 character resource names, their dictionary records would be 1062 bytes long.

- Other items are tokenized. For example, your entire grouping structure and profile names are also tokenized. Because these records are so small, considering them in an estimate would not be worth the effort.

- Although the Control Segment now contains many more objects than in previous releases, please consider a token may be referenced many thousands of times by events stored on the Masterfile. The amount of space conserved by this process is orders of magnitude over what would be required to maintain this data un-tokenized, (*even in a compressed form*).

Looking at the above example

- The numbers above are overstated. You probably will not have loggings for every userid or resource on your system. You will probably not have security changes performed on every userid and resource security definition. Token dictionary objects are very small.

- How you have your SET ELEMENT(*nnn*) specified will dictate the MAXIMUM length each object *could* be. If you retain data for a long time, or your installation "logs everything" you may require more space. For more information about the SET ELEMENT(*nnn*) command, see the *Command Reference Guide*.

- Some of the VSAM data may be is compressed which means the total number of additional VSAM records required to contain the full object will probably be less than the computed amount, or the actual length of residual records will be less.

- If you truly needed all the objects in the above example, you would need the number of VSAM records indicated PLUS additional VSAM records (some objects will probably be longer than 8192 bytes) to contain the entire object.

- The maximum size of an *array element* is currently 256 bytes (although no array element is currently that large as of this release). If you specified 6,144 as the maximum number of array elements you want to store on an object, the biggest object would be approximately 1.5 million bytes. The average is less than half that size or less, requiring on the average of 80 VSAM records to contain to payload data within a full object.

- Considering the VSAM data may be compressed (based on your E-SRF RETAIN options), and if the compression ratio is two to one, you can figure an array element object will have a maximum VSAM record requirement of 35 to 50 VSAM records (*if the object is full, and very few are*).

Determining how to set up your VSAM cluster

VSAM allocation in records:

Because most people do not know and probably do not care about the geometry of the physical DASD storage hardware that their Masterfile will be stored on, the only real thing you can be "sometimes" sure of is the ability to allocate space based on number of records being stored in the cluster.

The record length specification is based on the maximum record length (in this case, it is 8,192 bytes) and the *average* record length (in this case, it is 2,048 bytes).

```
RECORDS(250000 10000) -
RECORDSIZE(2048,8192) -
```

As you can see in our example, we are requesting enough disk space to contain 250,000 records with a maximum length of 8,192 bytes, but with an average length of 2,048.

The 10,000 specification allows VSAM to allocate "*secondary extents*" in the event the cluster becomes full.

VSAM does not simply multiply the number of records by the maximum length. It also considers the *average record length*, as it does the *Control Area* (CA) and the *Control Interval* (CA) sizes, as well as certain DASD device geometry.

As mentioned before, your cluster will probably be overstated in our example for the volume of data stated, but it will be safe for you to start with.

VSAM FREESPACE general information:

FREESPACE has been a classic problem in VSAM from its inception.

This specification is used to "sprinkle" free space (areas on disk that initially do not contain any useful data) throughout your cluster.

FREESPACE may be established on each (and every) Control Interval.

FREESPACE may be established on each (and every) Control AREA.

As you populate a VSAM cluster with new records, or make existing records longer the physical location of the existing record, or the one after it will no longer fit where it is located. VSAM moves data around in such a manner that the FREESPACE you set aside begins to get used. Remember, data is stored one record behind the other. If you want to make one bigger, or add one, it the data has to go somewhere.

If the FREESPACE for a particular Control Interval is exceeded, the Control Interval is split into two, giving the original Control Interval and the new Control Interval (*created in the Control Area's FREESPACE*) fifty percent free space (initially). It is quite possible based on the keys contained on your data that the new free space on one of both of these Control Intervals may never be used again.

CI splits are not a real good thing to have happen. If you split enough Control Intervals, you will start filling up the Control Area FREESPACE.

If the FREESPACE for a particular Control Area is exceeded, the Control Interval is split into two, giving each CAI fifty percent free space (initially).

Control Area splitting is a terrible thing to have happen, and should be avoided any way possible. Control Area splits will consume enormous resources and tend to be the root of most VSAM problems. When you run out of Control Area space, it is considered to be "out of space" and no more Control Area splits will occur.

Determining how to allocate FREESPACE and how much has always been at best "guesswork".

VSAM FREESPACE for the ESRF Masterfile:

VSAM FREESPACE for the E-SRF Event Reporting Masterfile can be a very easy task:

IF YOU RUN THE UPDATE FUNCTION WITH THE "CACHE" ON, SET YOUR FREESPACE TO ZERO.

With the cache on, E-SRF deals with the issues concerning adds, deletes, mass inserts, and constant record length extensions and reductions. By the time VSAM sees the data, it is loaded as contiguous sequential stream of data. A by-product of the use of the cache is you never have to reorganize your cluster.

If you do not use the cache, no matter what you specify as FREESPACE, you will have a very high volume of Control Interval and Control Area splits. Additionally, you will have to reorganize your cluster frequently.

Space Allocation recommendations:

Always let VSAM allocate your INDEX Component.

Unless you are an expert in VSAM and know the geometry of the DASD device, which is going to contain your cluster, make your allocation in RECORDS, not tracks or cylinders.

Specify SPANNED

Let VSAM decide how big to make your Control Interval (CI)

Determine how many records you need as if there were a one to one relationship between the number of desired objects and the number of VSAM records it would take to contain them. This may be a "guess" but has proven effective to get you started.

If you have no idea of how many objects (you are not alone), then start out by allocating 80,000 records. Do not worry about how much space this actually allocates as this time. You will adjust this later.

INITIALIZE, CONFIGURE, and SYNCHRONIZE your cluster (as discussed in previous sections of this publication) and then back up your cluster. If you get VSAM related errors, make the necessary adjustments to your cluster definitions and re attempt the process.

Run your Update Functions. Again, if you get VSAM related errors, make the necessary adjustments to your cluster definitions, restore your cluster from the previous step and re attempt the process.

In the beginning, the Update Function has to create all of the objects that normally just get updated. This means the update may take longer (per transaction relative to the size of the cluster) than normal.

Do an IDCAMS "*LISTCAT*" function and review the HIGH ALLOCATED RBA and the HIGH USED RBA (do not be misled, *look at all extents*). There will be a set of extents for the INDEX and DATA component. One or both oh these components could be in multiple extents.

The High Used RBA (Relative Byte Address) is how much space is actually being used. The High Allocated RBA is how much space is set aside to contain the data. If the two numbers are very far apart, you could make your cluster smaller. If they are close together, you may have to make it bigger. In the beginning, this has to be looked at. After your file "settles down" (you have been running your updates past your largest RETENTION period), it will generally use the same amount of space (barring unusual security changes or activity relating to journalizing).

Make sure you are looking at the last extent for each component.

To Re-Allocate your Masterfile VSAM Cluster:

This may have to be done in case you want to change the VSAM Cluster characteristics of your cluster

"*REPRO*" the cluster to a sequential flat file using the IDCAMS utility.

After checking the results of the "*REPRO*", delete and re-define your cluster the way you want it.

"*REPRO*" your cluster from the flat file back into your new cluster.

Run another "*LISTCAT*" to make sure the resulting cluster is the way you want it.

Masterfile Backup requirements:

Remember, like any other data processing application, you should always back up your VSAM cluster

This page intentionally left blank

Chapter 9: Masterfile Cache

Introduction:

E-SRF provides the ability to "cache" data in virtual storage. Under certain circumstances, this provision will greatly improve run time performance.

This topic is discussed throughout this publication and other related publications.

The reasoning behind caching is that access to main memory is much more efficient and less time consuming than accessing the same data stored on disk storage.

Two main issues must be considered and addressed. The first is the vast amount of disk I/O operations required to maintain the type of data contained on the Masterfile in such a manner that is useful for reporting. The second issue is the amount of processor resource that is needed to compress, decompress and prepare the same data to conserve both main storage and disk space.

Both of these issues are addressed by the cache facility incorporated into this product.

Please note that if you are only running a single or several reports undistributed reports, the cache may not yield as much for you as it would if you were running the UPDATE function, running many reports, or running one or more reports under the Report Distribution facility.

Level-Two cache:

To address the disk I/O issue, a "caching" option may be set which will load the entire contents of the Masterfile into virtual storage. A high speed optimized indexing structure is built that will manage sequential, random and direct insertions with relatively low processor usage. This particular cache may consume a large amount of virtual storage.

The Level-Two cache is maintained in the E-SRF's "Pooled Storage" Facility. In latter releases, Pooled Storage has been placed in one or more data-only address spaces.

As of release 1.6, the storage required for Pooled Storage, (*by default*), is maintained in a separate DATA-ONLY address space. Additional data-only address spaces were added in Release 2.1. The ability to revert to using the primary address space is available using the OPTION POOL execution parameter (as explained in the *E-SRF Event Reporting Command Guide*).

Placing the Level-Two cache in its own address space helps address Virtual Storage constraints that may exist in some installations with very large E-SRF Masterfiles. It also extends the maximum size of the Masterfile to four gigabytes. Masterfiles larger than that cannot be processed using the cache facility in its existing form. If you have a Masterfile that is larger than four gigabytes, it is recommended that you contact EKC technical support to determine why the file is that large, and, if it is unavoidable, what type of enhancements would be required to maintain larger caching structures in the product.

Running the level-two cache in the primary address space should only be done if you have no other option available to you due to your installations standards local to your environment.

Cache INITIALIZATION statistics:

The level-Two cache is normally initialized under your command although it may be initiated internally by special functions (such as the inter release Masterfile conversion function).

The following is posted on the E-SRF Event Reporting Control Report describing the initialization of the Level-Two cache.

This example was taken from a run where the Masterfile that was just initialized, configured and synchronized in a previous execution. The cache was activated in the current execution because the next command was the Masterfile's first Update Function. Notice how few objects exist on the Masterfile? Yours will probably be much greater.

```
------------------------------------------------>            CACHE ON
E400-$MCP INITIATING MASTERFILE CACHING OPTION
E610-$SCP CREATING POOLED STORAGE ADDRESS SPACE FOR POOL: 1
E600-$SCP ALLOCATING POOL STORAGE SEGMENT: 1, FOR POOL: 1
E610-$SCP CREATING POOLED STORAGE ADDRESS SPACE FOR POOL: 2
E600-$SCP ALLOCATING POOL STORAGE SEGMENT: 1, FOR POOL: 2
E610-$SCP CREATING POOLED STORAGE ADDRESS SPACE FOR POOL: 3
E600-$SCP ALLOCATING POOL STORAGE SEGMENT: 1, FOR POOL: 3
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 463,304
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 869,704

  1)  ..........2   NUMBER OF LEVEL 2 LOOKUP VECTORS
  2)  ......16,384   PRIMARY EXTENT OF EACH LOOKUP VECTOR
  3)  ......11,470   INPUT MASTER OBJECTS READ
  4)  ......11,470   MASTERFILE OBJECTS INITIALLY PLACED IN LEVEL 2 CACHE

E401-$MCP CACHE BUILD COMPLETE
```

The message E400 indicates the Masterfile Control Program (MCP) is about to initialize the Level-Two cache. Note that the Level-One cache will be dynamically initialized ON DEMAND and with only the number and size of buffers that are required.

The message E610 is posted by the E-SRF Storage management component and will only appear when the POOLED Storage facility is being established, and is being done in a data-Only address space. Once the Pooled Storage facility is established, it stays active until the E-SRF session terminates.

Message E610 will be posted every time it is determines that there is no room on any Pooled Storage segments to contain the current requested storage. Currently, pooled storage is allocated in eight megabyte "segments". Pooled storage is very efficiently managed. All attempts are made satisfy requests in the existing segment(s) before this action is taken. This includes relocating storage (and adjusting any pointers that make reference to it) so the storage is contiguous (without "gas bubbles" between requested areas.

Statistic line 1 shows how many lookup vectors are being employed by the cache manager.

Statistic line 2 shows the primary capacity of each lookup vector in this particular execution.

Statistic line 3 shows how many objects were read from the E-SRF Masterfile.

Statistic line 4 shows how many objects were placed in the Level-Two cache.

Cache UPGRADE statistics:

The level-Two cache is normally upgraded under your command or automatically when shutdown is detected.

The following is posted on the E-SRF Event Reporting Control Report describing the UPGRADE of the Level-Two cache.

This example was taken from a run where the Masterfile that was just initialized, configured and synchronized in a previous execution. The cache was activated in the current execution and an Update Function was executed. The cache UPGRADE was automatically invoked because shutdown was detected. This is the normal operational mode (let E-SRF upgrade the cache when the system is shutting down).

Please note there are two types of Cache Upgrades, one that updated the Masterfile's VSAM cluster (in place) and another that rebuilds the cluster just as if it was being initially loaded. The choice is made by the Masterfile Control program and is based on the number of changes required the cluster verses the total number of records contained on the cluster. Due to the high volume of update to the Masterfile, the REBUILD is the normal choice.

```
CACHE UPGRADE TO MASTERFILE IN PROGRESS:

E404-$MCP MASTERFILE REBUILD FROM CACHE IN PROGRESS

E460-$IOP OPEN MASTERFILE FOR: LOAD PROCESSING
E460-$IOP OPEN MASTERFILE FOR: UPDATE PROCESSING

      ++CACHED OBJECTS PROCESSED ........: 2,876
      ++DELETED OBJECTS NOT LOADED ......: 206
      ++ERRORS WRITING VSAM OBJECTS .....: 0
      ++TOTAL OBJECTS LOADED ON VSAM ....: 2,670

E418-$MCP MASTERFILE CONTROLS HAVE BEEN UPGRADED
E405-$MCP REBUILD HAS BEEN COMPLETED

RELEASE MASTERFILE CACHE STORAGE:

      ++OBJECTS WITH RESIDUAL EXPANSION..: 0
      ++AGGREGETE RESIDUAL EXPANSION.....: 0
      ++AVERAGE RESUDUAL BYTES PER OBJECT: 0
```

The title line indicates the Cache UPGRADE is being executed.

The message E404 indicates the Masterfile Control Program (MCP) decided to use the REBUILD method of applying the cache to the VSAM Cluster.

The first message E460 informs you that the current OPEN status of the Masterfile has been set to LOAD. This allows the cluster to be "loaded" with Masterfile data from the cache.

The second message E460 informs you that the current OPEN status of the Masterfile has been set to UPDATE. When you get this message, this means the Masterfile has been upgraded with the information contained in the Level-Two cache.

There may be other lines between these messages, such as progress scaling lines.

The next four statistics inform you how many cached objects were processed, how name DELETED (true deletes) objects were NOT written back to the cluster, how many errors occurred during the operation and the total number of OBJECTS written back to the Cluster.

Please note the number of objects is a number usually less than the number of actual VSAM records on the cluster. More than one VSAM record may be required to contain a single E-SRF object.

The message E418 informs you that the Masterfile's VSAM cluster is now current.

The message E405 informs you that the Level-Two cache rebuild function has been completed.

If this request is terminating the use of the Level-Two cache, a message indication this is posted (as can be seen in the example). Some statistics showing how efficient (or not-efficient) the cache was during its usage are shown.

Level-One cache:

To address the processor storage consumption required to manage the accessing, addressing, lengthening, compression and decompression of objects stored on the Masterfile, a “Level One” cache is automatically provided when the Level-Two cache is requested.

The level-One cache resides in the primary address space.

The Level-One cache contains fifteen storage buffers what will maintain the last fifteen objects processed (*when random Masterfile object processing is occurring*). Sequential processing will use only one buffer.

The size of these buffers is adjusted to be large enough to contain the object AFTER decompression. Size is determined by how large your objects are. The largest possible requirement would be the number of ELEMENTS times the size of the largest element on the Masterfile. This is a situation where running with an exceptionally high ELEMENTS specification can cause excessive storage consumption.

Issues relating to caching:

The Level-Two cache index structures, processing programs and all other storage areas are maintained in the Primary address space. Assuming the job is conditioned to execute with no REGION restrictions, this gives you two gigabytes (minus the common areas) for everything that is required.

The Level-Two cache data areas are maintained in Pooled Storage, which normally reside in data-only address spaces.

Placing Pooled Storage in data-only address spaces will allow the bulk of the Level-Two cache to reside outside the primary address space. If the Pooled Storage is set to reside in the primary address space, please realize that the entire Level-Two cache (not just the index structures) will reside in the primary address space.

You may discover E-SRF may abnormally terminate for error(s) relating to the use (or size) of data-only address spaces. If this occurs, please pay your systems programmer a visit to find out what you have to do in your installation to condition your job to avoid these problems.

Storage Limitations and APF-Authorization:

The use of virtual storage for the reasons mentioned above is a common practice with application systems that provide the type of processing capability found in E-SRF Event Reporting.

It should be noted that E-SRF Event Reporting runs as a normal "*application program*". This means the E-SRF Event Reporting System is not APF-authorized.

APF authorization allows application programs to issue specific requests to the MVS operation that can make the application able to do most of what MVS can do in its privileged state. This is something that auditors, security and system administrators have been addressing for many years.

Consider the Sort/Merge utility found on your system that is used throughout your daily production processing. Program products such as DB2, IMS and CICS also make use of multiple address spaces to address issued relating to performance and virtual Storage Constraint Relief. The main difference is these products are APF-authorized, thus making them "*immune*" to most installation limits placed on the use of facilities such as Data-Only address spaces.

This type of issue has been addressed before. The product could be set up to be APF-authorized, but other than getting around installation restrictions, there is no reason to open up the product to the exceptional processing ability provided by running APF authorized.

At the current release level, being APF-authorized would not provide any performance improvement. Placing an APF authorization requirement may create installation procedural issues. The potential for programming errors that could cause severe errors in your installation could also be opened up.

A future release of E-SRF Event Reporting may require APF-authorization. When (*and if*) that becomes a requirement, it will be published as such.

If your installation desires to make the product APF-Authorized, (*to get around installation storage requirements*), please contact EKC Technical Support for assistance.

This page intentionally left blank

Chapter 10: Masterfile Shutdown Statistics

The E-SRF Masterfile Control Program (MCP) maintains statistics that are posted on the E-SRF Event Reporting System Control Report.

This information may be reviewed to determine how much activity is being processed against your Masterfile and how well certain facilities (such as the cache) are providing services to you.

The following example is from a RACF Update run. Several reports were run after the update. The ELEMENTS specification was set to the maximum to illustrate how big a Level-One cache can get.

Please note that the statistics are numbered as a convenience. Your report may have more or less statistics based on what was executed. If this occurs, the numbering will not match the samples shown here.

```
MASTERFILE CONTROL STATISTICS:
```

The title line indicates that the Masterfile Control program (MCP) has entered its shutdown procedure and has posted statistics.

```
 1)  .........573   PHYSICAL VSAM STATS.: RECORD SEGMENT BROWSES PEFORMED
 2)  ..........15   . . . . . . . . . . . RECORD SEGMENT READS PERFORMED
 3)  .......3,498   . . . . . . . . . . . RECORD SEGMENT ADDS PERFORMED
 4)  ...........1   . . . . . . . . . . . RECORD SEGMENT UPDATES PERFORMED
 5)  ...........0   . . . . . . . . . . . RECORD SEGMENT DELETES PERFORMED
 6)  .........235   . . . . . . . . . . . MAXIMUM RECORD SEGMENTS FOR AN OBJECT
 7)  ...........0   . . . . . . . . . . . INTERNAL RRA EXCEPTIONS DETECTED

 8)  .......1,705   . . . . . . . . . . . USER HEADER RSS DATA RIGHT PADDING APPLIED
```

PHYSICAL VSAM STATISTICS show how much I/O activity was processed against the Masterfile's actual VSAM cluster. Notice we are showing "record segments" (*also referred to as VSAM records*). Record segment are how E-SRF objects are actually stored on the cluster. One or more record segments are required for each E-SRF OBJECT contained on the cluster. "Record Segments" only exist on the VSAM cluster and are rarely referred to in E-SRF data discussions (which deal with (OBJECTS).

Statistic 1 is a count of VSAM records read via sequential browse operations.

Statistic 2 is a count of VSAM records read via random read operations.

Statistic 3 is a count of VSAM records added to the cluster.

Statistic 4 is a count of VSAM records actually updated in place on the cluster

Statistic 5 is a count of VSAM records deleted from the cluster.

Statistic 6 is the maximum number of VSAM records required to contain a single E-SRF object.

Statistic 7 is a count of "exceptions" (errors) detected while assembling E-SRF objects from the data contained on the object's VSAM records. This statistic should always be zero.

Statistic 8 indicates how many times the User header object required right padding to be added because the original data was trimmed less than the mapped fields found in the data dictionary for the specific image.

```
 9)  ..........1   VSAM OBJECT COMMITTS: OBJECT COMMITTS DEFERRED
10)  ..........1   . . . . . . . . . . . OBJECT COMMITTS PERFORMED
11)  ..........0   . . . . . . . . . . . EXISTING OBJECT REFERENCE
```

VSAM OBJECT COMMITS show how many times changes to the VSAM cluster actually took place during normal Masterfile processing. Notice the low numbers. This is because the cache was activated immediately so all processing requests were services from the cache. Had the cache been inactive, these numbers would reflect the workload presented against the Masterfile during the life of the E-SRF session.

Statistic 9 shows how many objects were deferred from being applied to the Cluster. Work is deferred for many reasons within E-SRF, normally for performance purposes

Statistic 10 shows the actual number of objects had their record(s) applied to the cluster.

Statistic 11 shows how many times requests were made to acquire records from the cluster that were already contained in local I/O buffers within the Masterfile Control program. NOTE: this does not include look-asides performed by VSAM processing routines contained in MVS.

```
12)  .......2,670   SHORT READS . . . . . NUMBER OF SHORT READS PERFORMED
13)  ..........0   . . . . . . . . . . . : NUMBER OF SHORT READS EXPANDED
```

SHORT READS are Masterfile reads that only present a very small portion of the object for processing. These are normally executed when you are searching for a particular object by examining the object's *KEY*. Usually, if the key is what was being searched, a follow read is issued to "expand" the object. Other times, all that is needed is the data contained in the "shore" area and it is used for processing. Expanding the object means retrieving the entire object's contents.

Statistic 12 shows how many "short read" requests were made.

Statistic 13 shows how many "short read" requests were expanded to obtain the object's full content.

```
14)  ..........0   RELATIONAL READS....: NUMBER OF RELATIONAL READS REQUESTED
15)  ..........0   . . . . . . . . . . . : OBJECTS REQUESTED
16)  ..........0   . . . . . . . . . . . : REQUESTS SATISFIED BY EXISTING OBJECT
17)  ..........0   . . . . . . . . . . . : REQUESTS SATISFIED BY OBJECT READS
18)  ..........0   . . . . . . . . . . . : UNABLE TO SATISFY RELATIONAL REQUEST
```

RELATIONAL READS are Masterfile reads that are issued to "relate" data from one object to another.

An example of what would cause a relational read:

You are requesting a report showing all resource violations using the Resource Chronological object. You want the User's name and possibly some other user information to show on the report. This additional information is not contained on the Resource Chronological object. It is however stored on the User Header object (which is contained on the User Segment of the Masterfile). To provide the proper User Header information, the proper User Header (for the current userid) is "related" to the current Resource Chronological Object (the event) and retrieved.

Statistic 14 tells you how relational read requests were required.

Statistic 15 tells you how many objects were required to satisfy the above requests.

Statistic 16 indicates many requests were satisfied by the current request being the same as the previous request.

Statistic 17 indicates how many required actual Masterfile references were required to satisfy the above requests.

Statistic 18 tells you how many requests could not be satisfied because the related target object did not exist on the Masterfile.

```
19  ......32,767   RECORD LIMITS.......: MAXIMUM ELEMENTS ALLOWED FOR AN OBJECT
20)  ...2,032,008   . . . . . . . . . . . MAXIMUM INTERNAL OBJECT LENGTH
21)  ...5,832,782   . . . . . . . . . . . MAXIMUM EXTERNAL OBJECT LENGTH
```

RECORD LIMITS are computed lengths of storage areas based on Masterfile processing requirements during the current run.

Statistic 19 displays what you have currently set as the maximum ELEMENTS that will be stored on a single Masterfile object. This is a value you choose and can be altered.

Statistic 20 shows you what the maximum "internal" size of the largest Masterfile object was. The "internal" area is where Masterfile objects are stored in the form exists physically on the file.

Statistic 21 shows you what the maximum "external" size of the largest Masterfile object was. The "external" area is where Masterfile objects are placed when programs, such as an Update Overlay, or a Report Overlay request Masterfile data. Data contained in this area would be decompress and in a format documented for actual processing use.

The External area should be larger than the internal area because of data compression.

Level-One cache buffers are normally the size if the external area.

```
22)  .......3,235   LEVEL 2 CACHE STATS.: BROWSES PERFORMED
23)  .....304,462   . . . . . . . . . . . READS PERFORMED
24)  .......2,361   . . . . . . . . . . . ADDS PERFORMED
25)  .....299,753   . . . . . . . . . . . UPDATES PERFORMED
26)  .........287   . . . . . . . . . . . DELETES PERFORMED
```

LEVEL 2 CACHE STATS show you how your level-Two cache was utilized for this execution. Counts shown here are Masterfile OBJECTS. It takes one or more actual VSAM records to contain a Masterfile Object.

Statistic 22 displays how many BROWSE operations satisfied by the Level-Two cache.

Statistic 23 displays how many READ operations satisfied by the Level-Two cache.

Statistic 24 displays how many ADD operations satisfied by the Level-Two cache.

Statistic 25 displays how many UPDATE operations satisfied by the Level-Two cache.

Statistic 26 displays how many DELETE operations satisfied by the Level-Two cache.

```
27)  ......31,632   LEVEL 2 CACHE SLOTS:. LARGE ENOUGH TO ACCOMODATE REQUEST
28)  ......27,837   . . . . . . . . . . . RESIZED TO ACCOMODATE REQUEST
29)  ...........0   . . . . . . . . . . . DROPPED FROM POOL DUE TO DELETED OBJECT
```

LEVEL 2 CACHE SLOTS show you how your level-Two cache storage SLOTS were utilized for this execution.

Statistic 27 displays how many times the current cache SLOT was large enough to contain a Masterfile object when that Masterfile object was altered.

Statistic 28 displays how many times the current cache SLOT was not large enough to contain a Masterfile object when that Masterfile object was altered. In this case, SLOTS are resized in place (if possible), larger areas are located in the Pooled Storage area and relocated.

Statistic 29 displays how many times a SLOT was dropped from the Level-Two cache because the owning object was deleted from the Masterfile.

```
30)  .........15   LEVEL 1 CACHE STATS.: NUMBER OF BUFFERS USED
31)        87.5 M   . . . . . . . . . . . STORAGE FOR ALL BUFFERS

32)  .....304,364   . . . . . . . . . . . TOTAL DIRECT READ REQUESTS MADE
33)  .....244,310   . . . . . . . . . . . READS PROVIDED BY LOOK-ASIDE BUFFERS

34)  .....299,753   . . . . . . . . . . . UPDATES TO BUFFERS PERFORMED
35)  ......59,419   . . . . . . . . . . . TIMES BUFFERS WERE UPDATED TO CACHE
```

LEVEL 1 CACHE STATS show you how your Level-One cache was utilized for this execution. A Level-One cache will only exist if the Level-Two cache is present.

Statistic 30 displays how many buffers were used to contain the level 1 cache.

Statistic 31 shows you how much storage was required for all Level 1 cache buffers. Please note that buffers are allocated on demand and only resized if the current object to be contained in the buffer is larger than the target buffer. This means the individual buffers may be of different lengths.

Statistic 32 shows you how many times a READ request was made by looking into the level-One cache first. Normally all object READS issued against the Masterfile while this cache is active.

Statistic 33 shows you how many of the above reads were satisfied by buffers contained in the Level 1 cache. Your cache is providing great benefit if this number is high (*in relation to the request count*). Level 1 buffers contain objects exactly as they are used in processing.

Statistic 34 shows you how many times an UPDATE request was made by looking into the level 1 cache first. Normally all object UPDATES issued against the Masterfile while this cache is active.

Statistic 35 shows you how many of the above updates were satisfied by buffers contained in the Level 1 cache. Your cache is providing great benefit if this number is high (*in relation to the request count*). Object preparation can be a processor intense function. By referenced data in the Level 1 cache, we save the majority of the overhead needed to prepare an object for processing.

Chapter 11: Masterfile Update

Introduction

The E-SRF Event System uses the UPDATE function to maintain data the E-SRF Masterfile.

Journal data from specific Resident Security Systems (RSS) is posted as it occurs to a "sequential file", such as the operating system's SMF journaling facility.

At a scheduled time, (*usually off-hours when the RSS journals are available for processing*), the E-SRF Update function is executed.

The Update Control Program examines the UPDATE command and its sub-parameters, loads the applicable Update Overlay, and processes the request.

An update overlay is written for a specific RSS. For example, the overlay for RACF is called ESRFRACU.

An Update function is related to the RSS Update Overlay. This means each Update function processed within a particular E-SRF update execution must process data from the same RSS type. For example if you have two images of ACF2 and three images of RACF, you would need to run at least *TWO* Update functions, one for ACF2 and one for RACF. You may, but do not have to run all journals for a particular RSS in the same Update function.

The RSS Update Overlay directed by the Update Control Program reads the journals, normalizes the security system's logged event data, and posts transactions to the E-SRF Masterfile. A single captured event may cause anywhere from one to eight individual update transactions. Each update transaction involves the processing of a particular Masterfile object.

Performance

An object in E-SRF may be up to sixteen million characters long. This means a single object may "span" multiple VSAM physical records stored on your DISK drives.

It is possible that a single RSS journal record could represent five or more Update Function transactions to be posted against the Masterfile. Consider this example: Ten or more physical VSAM records could exist for each of the five objects to be updated. This would mean fifty or more VSAM records could be involved and updated for the single journal.

Remember, the list of array elements is in sequence relative to date and time. The update coming in may not be and probably will not be if you run multiple update functions.

Consider the following extreme example:

Assume there are 500,000 actual RSS journal records, each with an average of five transactions to post for each record. Let us also assume each object has ten physical VSAM records to contain the object's data. It could take as many as *twenty five million VSAM updates to complete the process*. This is why we always run the Update Function with the CACHE option ON. The objects are read into main storage and updated there. At the end of the job, the cache routines will rebuild the VSAM Masterfile cluster with the updated data.

Data Retention and Purging

Most E-SRF objects have specific number of days data will be retained before it is "purged" off the Masterfile.

Two types of purges are performed: Expired data purging and Unexpired ROLLOFFs.

Expired Data Purging

The first time the Update Function is executed in a given day, the automatic purge function is invoked. Any data that is older than the current date minus the retain days specified for a given object, will be removed from the Masterfile. This is how the Masterfile "*self cleans*" itself.

During the update, any individual transaction (at the object level) with data that is older than the event date minus the object's retention days will automatically be dropped from the update. All transactions for a particular RSS journal are processed independently for this reason.

Unexpired ROLLOFF Purging

The depth of the list of array elements is controlled by the SET ELEMENTS(*n*) option. It will not matter how large you set this, you may still encounter a situation where you have more elements than capacity to contain them. This occurs mostly with data centers that initially install this product. It normally comes from logging schemes where "*everything is logged*". In addition, extensive usage of a user ID attribute (*such as ACF2's NON-CNCL or RACF's OPERATIONS*) may contribute to this.

When the number of array elements is exceeded, the oldest ones are "rolled off" in groups equal to ten percent of the maximum list capacity. This means if you SET ELEMENTS(5000) and the list fills up, five hundred of the oldest elements will be removed from the list to make room for five hundred more future elements.

The current product supports 32,767 array elements per object. Future releases may increase this substantially. The solution to this problem may not be to increase this value.

The ELEMENTS limit is imposed to prevent you from using up your entire DASD farm, and all of the CPU cycles available to process and store loggings that possibly no one will ever look at. The real solution may be the proper analysis and altering of the RSS definitions (rules and profiles), and possibly the use of *E-SRF EXCLUDE Grouping Rules* to reduce loggings that are not required.

You picked a value for ELEMENTS. Maybe it is too small. Maybe it is not. If you really believe it is too small, by all means increase it, but do not do this because of this situation, unless, you really need the unexpired data that is being rolled off the Masterfile.

Is your Masterfile Too Big?

If you feel that your Masterfile is too large, or it takes too long to run your updates, you may be doing too much logging. This product makes it very easy for you to correct this situation. You may find that over eighty percent of all of your loggings come from less than ten percent of the users and resources. This is very common for new users of this product.

Set the retain values for all the objects except the RR, UA, and UR to zero. Set RR and UR to a low value, such as five. Leave the UA retain value alone. Doing this will tell E-SRF to only process the "recap" transactions. Your Masterfile will become much smaller, and the update will run quicker. Run the "top 40" logging reports and see which USERS and which RESOURCES are causing the problem and modify the RSS definitions or write grouping rules to exclude them from E-SRF.

Do this for several weeks, or until you feel that your logging is under control, then restore your desired retain values. You will now have meaningful data to produce meaningful reports, make your DASD manager happy with a smaller E-SRF Masterfile, and greatly reduce the size of the SMF journal files.

Some RSS logging capabilities do not allow you to maintain the depth of control desired. E-SRF provides a means that will allow you to *EXCLUDE* loggings from the Masterfile during the Update Function. This involves turning on the "EXCLUDE" capability and setting up grouping rules to specify "what" and "for whom". Refer to the *Command Guide* and the *User's Guide* for more information.

Make sure your *ELEMENTS* and *RETAIN* values are realistic with your reporting needs.

Transaction Processing Functions Available

Each RSS creates event journals in their own way. E-SRF is designed to contain all RSS required journalized data, in a single system to produce reports to end users, without requiring any knowledge of the security system being used to protect their data.

Some systems provide more data than others do.

It is the design direction of this product to, *if possible,* narrow this "information gap" to make reviewing security events as consistent as possible. The update overlays, in conjunction with future journalizing enhancement provisions, will continue to address these issues and enhance this ability in future releases of this product.

In the list below are the UPDATE internal functions currently available to the RSS Update Overlays.

UPFC	**Update Console Chronological Object**

Description: Provide data and manage the Console Chronological Object.

This object contains data that is RSS dependent. It is not necessarily data presented on the operator's console.

UPRC	**Update Resource Chronological Object**

Description: Provide data and manage the Resource Chronological Object.

This is a detail of each logged access the resource sustained. There will be a single array element for each access.

UPRM	**Update Resource Maintenance Object**

Description: Provide data and manage the Resource Maintenance Object.

Information about the maintenance of RSS data for RESOURCES is stored on this object. This data is RSS dependent.

UPRR	**Update Resource Recap Object**

Description: Provide data and manage the Resource Recap Object.

Data is summarized as to the number of logged events that were sustained for a particular RESOURCE for a particular day. This means there will be one array element for each day that access was sustained.

UPRS	**Update Resource Journal Statistical Daily Summary Object**

Description: Provide data and manage the Resource Statistical daily Summary Object.

Data is summarized as to the number of journalized security events that were sustained for a particular RESOURCE for a particular USERID during a particular day. Journalizing in this case include both journalized granted accesses, granted access due to special reasons and access denials (violations). This means if one hundred users accessed the resource in a single day and had their access journalized by the resident security system, there will be one hundred array elements for the particular day for the particular resource.

UPSR — Update Source Activity Recap Object

Description: Provide data and manage the Source Recap Object.

Recap data is stored pertaining to the SOURCE (terminal ID) which hosted the event. Note this is the only place where the number of invalid userids is tracked. If you need to how many invalid userids were specified during signon, the update control report displays a summary of each ID and the number of tiles this has occurred. This is enough for most situations. However, this object stores an aggregate count of how many occurred on a particular SOURCE, and if this is what you need, this is the object to use to obtain the information.

UPSU — Update Source User Chronological Object

Description: Provide data and manage the Source User Chronological Object.

When a user enters an invalid userid, the Resident Security System rejects the attempt and follows up with a journal indicating the actual character string presented as well as which "source" the incident occurred on.

E-SRF does not maintain Invalid Userids on its Masterfile's user Segment. Doing so would clutter up the User Segment with what would amount to typographical errors and would have little or no meaning.

The Update Function publishes a summary of each invalid userid's character pattern and how many times the pattern was presented relative to the set of update journals being processed.

To make post update reporting of invalid userids possible, these events are also recorded within the SU (Source Userid) object (*if this object is being retained*). The event is recorded treating the character string as a "resource". By activating the SU object, and preparing ESRFLIST reports, you can produce reports showing some (or all) invalid userid signon attempts.

UPUA — Replace/Update User Header Object

Description: Provide data and manage the User Header Object. This includes replacing the RSS dependent data if it is more recent than what is currently stored in the object.

Deleted user management is also carried out. If the RSS requests the user to be deleted, the date of the delete is stored in the object. The user is retained on file until the number of days lapsed between the delete date and today's date is equal to go greater than the largest "retain" setting for the entire object.

UPUB — Update User Administration Chronological Object

Description: Provide data and manage the User Administration Chronological Object.

This routine is called if a particular user applies maintenance against any RSS definitions. The only users that will be associated with this object will be security administrators. This object provides you with data on how who is doing what to what.

UPUC — Update User Chronological Object

Description: Provide data and manage the User Chronological Object.

This is a detail of each logged access the user initiated. There will be a single array element for each access.

UPUF Update User FIRECALL Chronological Object

Description: Provide data and manage the User Firecall Resource Log Object. This object will only exist if the particular user engages in a session with the EKC FIRECALL facility.

UPUI Update User Header (UA) with field list

Description: Update UA object. Provide a list of individual field datanames (as per IMAGE dictionary) and their respective data values to be used to upgrade individual field dataname values. Pass entire RSS dependent field area, or a *"dummy"* to be used to insert a new user if required.

UPUM Update User Maintenance Object

Description: Provide data and manage the User Maintenance Object.

Information about the maintenance of RSS data for USERS is stored on this object. This data is RSS dependent.

UPUP Update User Profile Object

Description: Provide data and manage the User Profile Object.

This object maintains RACF CONNECT groups and their attributes. If you are not applying update from a RACF image, this function is not executed. Each time a CONNECT group is established, updated or removed; this function is called to maintain this information of the Masterfile.

UPUR Update User Recap Object

Description: Provide data and manage the User Recap Object.

Data is summarized as to the number of logged events that were requested for a particular USER for a particular day. This means there will be one array element for each day that any logged access was requested.

UPUS Update User Resource Journal Statistical Summary Object

Description: Provide data and manage the User Statistical Summary Object.

Data is summarized as to the number of journalized security events that were sustained for a particular USER for a particular RESOURCE during a particular day. Journalizing in this case include both Journalized granted accesses, granted access due to special reasons and access denials (violations). This means if a user accessed one hundred resources in a single day that had their access journalized by the resident security system, there will be one hundred array elements for the particular day for the particular userid.

UPUT Update User Trace Chronological Object

Description: Provide data and manage the User Trace Chronological Object.

This is a detail of each TRACED access the user initiated. There will be a single array element for each access. This data is similar to the UC object. If the user is not being traced, this object will not exist.

ACF2 Update Processing: Facilities and Limitations

ACF2 uses the SMF system journalizing facility to write its journal records. A user SMF record id is selected from the journal. It is usually type 230, but does not have to be. The default E-SRF update processing record id is type 230. If the ACF2 record id is not type 230, you must override the SMF record type in the UPDATE command. Failure to do this will cause only type 230 journal records to be processed by the ACF2 update overlay program.

ACF2 provides a robust array of journal data, which enables E-SRF to do full function processing against this data.

The ACF2 SMF record contains a common section, followed by specific data relating to a "sub-type".

The E-SRF Update Overlay (ESRFACFU) is used to read the ACF2 journal records and "normalize" this data into a format, which the E-SRF Update facility may use to process transactions against the Masterfile's objects.

Limitations:

The current implementation of ACF2 provides full support except where the SMF journal record formats are not disclosed for licensed end user access. Currently, these journals are SECTRACE, MVS Open Edition, and the capability to display before and after rule administration. If this information becomes available, E-SRF will be enhanced to process this data in the current release under development, or a maintenance release, which ever is earlier.

Type: A	**ACF2 Console Journal**

Function: Extracts data relative to console operator interaction with the ACF2 started task.

Transactions: UPFC

Type: C	**ACF2 VM Command Services**

Function: This journal is not processed by E-SRF at this time.

Transactions: NONE

Type: D	**ACF2 Dataset Log/Violation Journal**

Function: Extracts data relative to dataset rule logging, both allowed accesses and violations.

Transactions: ETF/A test rule: UPUC and UPRC

Trace record: UPUT

All others: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS and UPSR

Type: E	**ACF2 Information Storage Update Event Journal**

Function: Extracts data relative to the updating of the ACF2 Information Storage VSAM cluster (INFO STORAGE DATABASE).

Please note that the entire type code which includes the single character "drawer ID" and its associated three character "type" code is processed as a single data field. The entire four characters are inserted into the class field, left justified padded with blanks.

Transactions: UPRM UPUB

Type: G — ACF2 Invalid GSO Record

Function: This journal is not processed by E-SRF at this time.

Transactions: NONE

Type: I — ACF2 Intercept Journal

Function: This journal is not processed by E-SRF at this time.

Transactions: NONE

Type: J — ACF2 "Restricted" Logon Journal

Function: This journal occurs when a restricted logon is being processed.

A restricted logon will occur when a logon without a password is processed by ACF2 and the signon is valid. For example, when a started task is initiated or a batch job is executed.

A logon is performed on behalf of the started task with a Logonid that is the same as the task name. This Logonid cannot have a password, and contains the RESTRICT privilege. These LIDS may be very powerful, and are associated with the task for the duration of its execution.

Transactions: Processing is deferred.

The information is normalized and held. When the follow-up signon, a System Entry Validation (SEV) call is made, the association to the signon is performed and the SEV then becomes a "SIGNON-R" instead of a "SIGNON".

Type: L — ACF2 Logonid Maintenance Journal

Function: Extracts data relative to any Logonid update and System Entry Validation (signon) and normalizes the event. There are two formats to this journal: a total LIDREC replacement, and an Alter Request Element (ARE) field list processor. In either format, a LIDREC replacement image is provided, with a time-stamp associated with the LIDREC update.

Transactions:

Lid maintenance:	UPUB (against the requester's Logonid)
	UPUA (against the target Logonid)
	UPUM (against the target Logonid for each changed field)
SEV (Signon):	UPUA, UPUC, UPUR and UPSR
SEV Invalid Userid:	UPSU

Type: O — ACF2 Open Edition Journal

Function: This journal is not processed by E-SRF at this time. The data structures of this journal record are not available for customer usage at this time.

Transactions: NONE

Type: P	**ACF2 Invalid Password / Authority Journal**

Function: Extracts data relative to the unsuccessful signing on of a user (System Entry validation failure).

Transactions: Invalid Logonid: UPSR and post to invalid Logonid list

Other failures: UPUC, UPUR and UPSR

Type: R	**ACF2 dataset Rule Insert/Update/Delete Journal**

Function: Extracts data relative to the maintenance of dataset Rules.

Transactions: UPRM and UPUB

Type: S	**ACF2 SECTRACE Journal**

Function: This journal is not processed by E-SRF at this time. The data structures of this journal record are not available for customer usage at this time.

Transactions: NONE

Type: T	**ACF2 TSO Transaction Journal**

Function: This journal is not processed by E-SRF at this time

Transactions: NONE

Type: V	**ACF2 generalized Resource Log/Violation Journal**

Function: Extracts data relative to resources validated using Generalized Resource Rules, both allowed logged accesses and violations.

Please note that the three character "TYPE" code is inserted into the eight character CLASS normalized field, left justified, padded with blanks.

Transactions: ETF/A test rule: UPUC and UPRC

Trace record: UPUT

All others: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS and UPSR

Type: Z	**ACF2 Distributed Database Function Journal**

Function: This journal is not processed by E-SRF at this time.

Transactions: NONE

Type: x'85'	**ETF/A "Firecall" Journal**

Function: Extracts data relative to the use of the EKC's ETF/A "Firecall" function.

Transactions: UPUC, UPUR and UPUF

RACF Update Processing: Facilities and Limitations

RACF uses the SMF system journalizing facility to write its journal records. SMF *system* record type 80 is used, along with three others that E-SRF is not processing at this time.

RACF provides a very respectable array of journal data. E-SRF is able to do most functions with the current RACF journals provided. All access control functionality is present. What is lacking at this time is the full capability of dynamic user information update. Because of the way RACF maintains user information, it is more difficult than with other security systems for you to use data associated with users when providing reporting criteria. User grouping is also affected. If grouping your users is a requirement, it is more difficult to group your users in other structures rather than the way they are grouped for RACF processing.

If you have a need for reports based on information contained on the User Header object, you may have to run the SYNCHRONIZE command to keep the user data that exists in E-SRF current with the user data that exists on your RACF system. Please refer to Appendix H for a summary of the various user fields maintained for RACF, and how they are upgraded on the Masterfile. Most of the important fields are dynamically upgraded on the Masterfile, but some are not. Additionally, because it is possible for you to skip some SMF Journalized RACF update processing, E-SRF has no way to guarantee the integrity of the data contained on the Masterfile. More development effort is being expended to assist with these shortcomings. If you perform a SYNCHRONIZE, your Masterfile will be at the same level as the RACF database was when the RACF database unload utility was executed.

If the above-mentioned issues are understood and provided for, you will find the RACF implementation of this product to be equal or better than other security systems.

The RACF SMF record is a standard IBM formatted record with complete disclosure of the formats of every piece of information contained on it. This product will be enhanced to continue taking advantage of this situation. This will result in even more capability, such as the full support of MVS Open Edition.

The E-SRF Update Overlay (ESRFRACU) is used to read the RACF journal records and "normalize" the data into a format that the E-SRF Update facility may use to process transactions against the various Masterfile objects.

Limitations:

All RACF journal data is disclosed. The limitations exist only in the ability to maintain user data.

E-SRF will be enhanced to provide support for user data when this information becomes a component of RACF or, an optional enhancement will be developed to provide this data to you. MVS Open Edition will be supported in a future release.

Certain events (*as indicated*), specifically events 3, 4, 5, 6, and 7 are skipped. Programming logic exists within the RACF Update Overlay to process them (as described below), but the data is redundant with other data contained on within other journals.

If you feel this data is required and wish to have it included, please contact EKC Technical Support for information on how to activate these event types.

Event 01:	**Job initiation, LOGON/LOGOFF from RACINIT**
Function:	Extract all LOGON/LOGOFF data.
Transactions:	Invalid Userid: UPSR and post to invalid Logonid list
	Other actions: UPUC, UPUR UPSR, and UPUAI

Event 02: Resource Access (RACHECK and DIRAUTH)

Function: Extract all information related to RESOURCE validation.

Transactions: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS UPSR, and UPUAI

Event 03: Add volume (RACDEF type ADDVOL and CHGVOL)

EVENT CURRENTLY SKIPPED DUE TO PRESENCE ON OTHER JOURNALS.

Function: Extract all information related to the SAF calls issued when defining and altering VOLUMES.

Transactions: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS, UPSR and UPUAI

Event 04: Rename dataset (RACDEF type DEFINE or NEWNAME)

EVENT CURRENTLY SKIPPED DUE TO PRESENCE ON OTHER JOURNALS.

Function: Extract all information related to the SAF call issued while defining DATASET resources.

Transactions: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS, UPSR and UPUAI

Event 05: Delete resource (RACDEF type DELETE)

EVENT CURRENTLY SKIPPED DUE TO PRESENCE ON OTHER JOURNALS.

Function: Extract all information related to the SAF call issued while deleting a DATASET resource.

Transactions: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS, UPSR and UPUAI

Event 06: Delete one volume of a multi-volume resource (RACDEF type DELETE)

EVENT CURRENTLY SKIPPED DUE TO PRESENCE ON OTHER JOURNALS.

Function: Extract all information related to the SAF call issued while deleting a single volume of a multiple volume resource.

Transactions: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS and UPSR

Event 07: Define resource (RACDEF type DEFINE)

EVENT CURRENTLY SKIPPED DUE TO PRESENCE ON OTHER JOURNALS.

Function: Extract all information related to the SAF call issued while defining a resource.

Transactions: UPUC, UPUR, UPUS, UPRC, UPRR, UPRS, UPSR and UPUAI

Event 08: ADDSD Add DATASET security definition to RACF

Function: Extract all information related the addition of a DATASET access control definition to RACF.

Transactions: UPUB and UPRM

Event 09: ADDGROUP Add GROUP to RACF

Function: Extract all information related the addition of a RACF GROUP definition.

Transactions: UPUB and UPRM

Event 10: ADDUSER Add USER security definition to RACF

Function: Extract all information related the addition of a USER access control definition to RACF.

Transactions: UPUB, UPUM, UPUP and UPUAI

Event 11: ALTDSD Alter existing RACF DATASET security definition.

Function: Extract all information related the alteration of an existing RACF DATASET definition.

Transactions: UPUB, UPRM and UPUAI

Event 12: ALTGROUP Alter existing RACF GROUP definition

Function: Extract all information related the alteration of an existing RACF GROUP definition.

Transactions: UPUB and UPRM

Event 13: ALTUSER Alter existing RACF USER definition

Function: Extract all information related the alteration of an existing RACF USER definition.

Transactions: UPUB, UPUM, UPUP and UPUAI

Event 14: CONNECT Add or alter an existing RACF CONNECT specification

Function: Extract all information related the use of the CONNECT specification.

Transactions: UPUB, UPUM and UPUP

Event 15: DELDSD Delete existing RACF DATASET security definition.

Function: Extract all information related the deletion of an existing RACF DATASET definition.

Transactions: UPUB, UPRM and UPUAI

Event 16: DELGROUP Delete existing RACF GROUP definition

Function: Extract all information related the deletion of an existing RACF GROUP definition.

Transactions: UPUB and UPRM

Event 17: ALTUSER Delete existing RACF USER definition

Function: Extract all information related the deletion of an existing RACF GROUP definition.

Transactions: UPUB and UPUM

Event 18: PASSWORD RACF user password administrative maintenance

Function: Extract all information related the administration of USER passwords.

Transactions: UPUB, UPUM and UPUAI

Event 19:	PERMIT	RACF resource PERMIT administrative maintenance

Function: Extract all information related the administration of the PERMIT access control specification.

Transactions: UPUB and UPRM

Event 20:	RALTER	Alter non-dataset RESOURCE security definition in RACF

Function: Extract all information related the alteration of non-dataset RESOURCE access controls in RACF.

Transactions: UPUB and UPRM

Event 21:	RDEFINE	Add non-dataset RESOURCE security definition to RACF

Function: Extract all information related the addition of a non-dataset RESOURCE access controls to RACF.

Transactions: UPUB and UPRM

Event 22:	RDELETE	Delete non-dataset RESOURCE security definition in RACF

Function: Extract all information related the deletion of non-dataset RESOURCE access controls in RACF.

Transactions: UPUB and UPRM

Event 23:	REMOVE	Remove a USER from an existing RACF GROUP

Function: Extract all information related the removal from a specific USER from an existing RACF GROUP.

Transactions: UPUB, UPUM and UPUP

Event 24:	SETROPTS	Define and maintain RACF system options

Function: Log the fact that SETROPTS were changed.

Transactions: UPUB

Event 25:	RVARY	Alter current RACF system control in real time

Function: Determine the nature of the request and log the action of the CONSOLE Chronological Object.

Transactions: UPUB and UPFC

Event 26 - 58:	Open Edition extensions

Function: These journals are not processed in this release. They will however be processed in a near future release.

Transactions: NONE

This page intentionally left blank

Chapter 12: Update Control Report

Introduction

The E-SRF Update Function maintains a control report that may be of interest. This report is written to SYSPRINT along with all other E-SRF control information. It is not intended to be a "security" type report. Various events occurring during the Update Function, and processing statistics are posted on this report. Each individual execution of the Update Function will produce this report.

This report may change as new releases of the product are provided with additional functionality added to the Update Function. Items are normally added to this report. With rare exception, items are normally never removed from it.

The Update Function control report is the same for all supported RSS with the exception of RSS specific information.

The following example is from an ACF2 Update Function execution with narration of what this information means:

Turning on the cache before the UPDATE command

```
-------------------------------------------------->     CACHE   ON
E400-$MCP INITIATING MASTERFILE CACHING OPTION
E610-$SCP CREATING POOLED STORAGE ADDRESS SPACE FOR POOL: 1
E600-$SCP ALLOCATING POOL STORAGE SEGMENT:  1, FOR POOL: 1
E610-$SCP CREATING POOLED STORAGE ADDRESS SPACE FOR POOL: 3
E600-$SCP ALLOCATING POOL STORAGE SEGMENT:  1, FOR POOL: 3

  1)  ..........2   NUMBER OF VEVEL 2 LOOKUP VECTORS
  2)  ......16,384   CACHE PRIMARY EXTENT
  3)  .........386   INPUT MASTER OBJECTS READ
  4)  .........386   MASTERFILE OBJECTS INITIALLY PLACED IN CACHE

E401-$MCP CACHE BUILD COMPLETE
```

The user wisely decided to use the E-SRF caching facility for the execution of the Update Function. Normally, this specification only needs to be none once in single command processor execution. All references to the Masterfile will be made from the cache until the cache is rebuilt.

Notice the E610 messages indicating the cache is contained in separate address spaces. In this execution, there are two separate level -2 cache structures providing cache support for the Masterfile data. Normally three structures are required. Because this was the first update after a COLD job, the Masterfile only contains control information and userids. The first resource that is applied will trigger the addition of POOL 2. You will see this in the Update Function sample output.

The E600 message is posted each time a new cache storage segment is required. Currently, each cache storage segment is eight million bytes long.

Additional segments are usually required, but are not allocated until there is storage request is unable to fit in any storage segment allocated to a specific pool.

Issuing the UPDATE command

```
------------------------------------------------>     UPDATE  RSS(ACF2) DDNAME(SMFCHGO)
 E180-CMD  MASTERFILE UPDATE IN PROGRESS

 E460-$IOP OPEN MASTERFILE FOR: UPDATE PROCESSING
```

The user requested the Update Function. The Resident Security System (RSS) was identified as ACF2. Has the journals been created by a RACF system, RACF would have been specified instead of ACF2. This tells E-SRF which Update Overlay to use to process the data. You may have multiple Images and domains contained in the specified DDNAME, provided all data came from the same RSS type.

This means, if you have both ACF2 and RACF, it would take at least two Update Functions to apply all of your journals to the E-SRF Masterfile.

The DDNAME specification indicates which DD in the JCL will be used for the SMF input data for the update.

The message E180 indicates the Update Function has been cleared to run.

You may notice the E460 message. Normally, the Masterfile is opened for INPUT processing. As soon as E-SRF needs up write something to the Masterfile, it is closed and re-opened for UPDATE. If you do not have UPDATE access to the Masterfile, a standard security violation will occur and the job will terminate as such.

Once the Masterfile has been opened for Update, it stays that way for the duration of the command processor's execution.

```
--> BEGIN MASTERFILE UPDATE FUNCTION
 E181-$UCP STEPDOWNS WILL BE ACCEPTED
```

This posting indicates the Update Function has started. Any particular information regarding the way the update will run will be shown here.

Notice the E181 message telling you that "STEPDOWNS: will be accepted. This is the default mode of operation and should always be run this way. Stepdowns are update transaction activity that is out of date and time sequence. With the rare exception of a single update file being processed each day, it is almost impossible to run any other way. Please note USERID header information Stepdowns are always rejected to keep the Masterfile from being regressed due to updating it out of order.

Masterfile Cleanup – Expired Data Purge

```
E771-$UCP MASTERFILE EXPIRED DATA PURGE IN PROGRESS
E790-$UCP DELETED USER COMPUTED RETENTION DAYS: 1,024, FROM OBJECT: A, PURGE DATE: 07/12/1994

  1)  ......14,320   OBJECTS ANALYZED
  2)  ...........0   NUMBER OF EXPIRED ELEMENTS DETECTED
  3)  ...........0   OBJECTS WHICH HAD EXPIRED EVENTS REMOVED
  4)  ...........0   OBJECTS WITH NO REMAINING ELEMENTS DELETED

  5)  ...........0   DELETED USERID OBJECTS DETECTED
  6)  ...........0   DELETED USERID OBJECTS EXPIRED AND DROPPED

  7)  ...........0   ... (CONSOLE CHRONOLOGICAL OBJECTS)

  8)  ...........0   ... (RESOURCE CHRONOLOGICAL OBJECTS)
 .9)  ...........0   ... (RESOURCE MAINTENANCE LOG OBJECTS)
 10)  ...........0   ... (RESOURCE RECAP OBJECTS)
 11)  ...........0   ... (RESOURCE STATISTICAL OBJECTS)

 12)  ...........0   ... (SOURCE RECAP OBJECTS)
 13)  ...........0   ... (SOURCE INVALID USERID OBJECTS)

 14)  ...........0   ... (USER ADMINISTRATIVE MAINTENANCE OBJECTS)
 15)  ...........0   ... (USER CHRONOLOGICAL OBJECTS)
 16)  ...........0   ... (USER FIRECALL LOG OBJECTS)
 17)  ...........0   ... (USER MAINTENANCE LOG OBJECTS)
 18)  ...........0   ... (USER RPROFILE OBJECTS
 19)  ...........0   ... (USER RECAP OBJECTS)
 20)  ...........0   ... (USER STATISTICAL OBJECTS)
 21)  ...........0   ... (USER RSS EVENT TRACE OBJECTS)
```

The message E771 indicates the Masterfile EXPIRED data purge is taking place. This will run automatically the first time you run the Update Function on a particular calendar day. It will run only once. If you skip a day, it still only runs once, (just does more work).

This function will examine all objects on the Masterfile and automatically remove all data that has been determined as EXPIRED. The criteria is controlled by the RETAIN specifications that you have established for each object type contained on the Masterfile. In the case of this example the RETAIN values are set to 4095 days resulting in *NO EXPIRED DATA*. Normally, much lower days are set and therefore you will see activity.

When you first start using this product, there will be no expired data until data on the Masterfile has exceeded the retain days specified for specific objects.

The E790 message posts the calculated adjusted retention days of the User Header object. This is to make sure the User header is still present for reporting data in the event that the Userid is deleted. The idea is to make sure it stays on the file for the same number of days that the longest retention for any object type on the USER SEGMENT of the Masterfile.

Initiating a specific update

```
E275-$CRT LOADING MODULE: ESRFACFU
E750-$UCP ENTER UPDATE: ESRFACFU
E704-ACFU STARTING ACF2 DAILY EVENT UPDATE
```

The E275 message indicates the RSS Update Overlay is being loaded into storage and will be executed for the duration of the update. Because we specified ACF2 as the RSS, the program name is EARFACFU. Had this been a RACF update, the program name would have been ESRFRACU.

The E750 message indicates control is being transferred to the Update Overlay.

The E704 message indicates the official start of update transaction processing.

Reading Journals and applying the data to the Masterfile

```
E714-ACFU SMF SECURITY RECORD ID: 230
E754-$UCP READING SMF INPUT FROM DDNAME: SMFCHGO
E327-$IOP FILE AVAILABLE FOR DD: SMFCHGO,  (RECORD LENGTH: 12,284)
E610-$SCP CREATING POOLED STORAGE ADDRESS SPACE FOR POOL: 2
E600-$SCP ALLOCATING POOL STORAGE SEGMENT: 1, FOR POOL: 2
E716-ACFU SMF SECURITY RECORDS PROCESSED: 10,000
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 609,608
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 812,808
E716-ACFU SMF SECURITY RECORDS PROCESSED: 20,000
E600-$SCP ALLOCATING POOL STORAGE SEGMENT: 2, FOR POOL: 3
E716-ACFU SMF SECURITY RECORDS PROCESSED: 30,000
E716-ACFU SMF SECURITY RECORDS PROCESSED: 40,000
E716-ACFU SMF SECURITY RECORDS PROCESSED: 50,000
E716-ACFU SMF SECURITY RECORDS PROCESSED: 60,000
E716-ACFU SMF SECURITY RECORDS PROCESSED: 70,000
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 1,016,008
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 1,219,208
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 1,422,408
E416-$MCC INTERNAL OBJECT AREA EXPANDED TO: 1,625,608
```

This is the bulk of the Update Function's execution.

This section is the same for any RSS, this particular example an ACF2 update was run. It could have been a RACF update and the control reporting would have been the same.

The E714 message posts the Security Journal SMF record type and the E754 confirms the DDNAME of where the SMF data will come from.

The E327 posts the status of the actual opening of the SMF input dataset.

The Update Function keeps track of how many SMF records it processed for update. Each time 10,000 records are processed; the E716 message is posted on the control report for informational purposes. Please note that 10,000 records generate many times that many actual Update Transactions posted to the Masterfile.

The E429 message is posted to indicate the compression of the newly updated Masterfile object cannot be completed because the target area is too small. E-SRF really does not know how much data you are going to present for updating so it tries to select a low conservative size for its data areas. When errors like this occur, they are recovered by re-establishing larger areas as can be seen by the recovery E416 message. A larger area was provided and processing continues until the problem occurs again.

Other events and exceptions may be logged here. Possibly the E716 and the E600 messages, most of these messages will not mean anything to you as a user of this product unless you have a problem. If a problem does occur, these messages may aid in solving the problem.

End of journal data detected

```
E755-$UCP END OF UPDATE INPUT DETECTED
E716-ACFU SMF SECURITY RECORDS PROCESSED: 76,632
E711-ACFU END OF SMF INPUT DATA DETECTED
```

The E755 message indicates "end of data' was detected on the update input SMF file.

The E716 message posts the total number of security journal records that were used to perform the update.

The E711 message indicates “end of data” detected by the Update Overlay (in this case the ACF2 Update Overlay program ESRFACFU) and Masterfile updating has been completed.

Journal File Recap

```
E781-$UCP UPDATE DATE RANGE WAS: 04/25/1997 TO: 04/27/1997

E756-$UCP SMF INPUT STATISTICS:

   RECORDS MAX.LEN  DESCRIPTION:.
---------- -------  ------------------------------------------------
.........1 .....18  TYPE 002 (02) - DUMP HEADER
.........1 .....18  TYPE 003 (03) - DUMP TRAILER
.......512 ....348  TYPE 224 (E0) - USER SMF RECORD TYPE 224
....76,632 ..1,184  TYPE 230 (E6) - ACF2 SECURITY LOG RECORD (DEFAULT)

E328-$IOP FILE NO LONGER AVAILABLE FOR DD: SMFCHGO
```

The above information is posted simply for reference. It recaps the physical characteristics of your input update file.

The E781 message tells you the date range of security data found of the update file.

Please note: The file used for this update contains some very old SMF records. You may see this throughout this product’s documentation. This data is used to verify every release to help insure product performance across releases. The actual processing was run and verified using the current product release and included in the product’s documentation

The E756 message indicates the update data was SMF, and the statistics below indicate which SMF record types were found on the file. The Update Function only requires certain record types. In the case of ACF2, this is normally user type 230. In the case of RACF, system type 80 is the bulk of the data, with other system record types required to fill in what RACF does not journal.

This display shows all types detected, how many and the maximum length of each type detected.

Normally, the input data provided to this product was ‘pre stripped’, meaning only data required for processing will be present. This may not be the case. You may .get the entire SMF file. This is OK because E-SRF will only process the records that are required and skip all the non-essential records.

The E328 message indicates the update input file has been closed and is no longer available to E-SRF unless it is re-opened again.

Processing Exception List

```
E739-ACFU UPDATE PROCESSING EXCEPTION LIST

T  REASON      KEY OR DESCRIPTION                              OCCURRENCES
-  ----------  ----------------------------------------------  -----------
END OF EXCEPTION TABLE
```

The E739 message posts processing exceptions. The reason for the exception, the KEY (or description) of the data and how many times it occurred will be shown.

This should be an empty section. If something is displayed here, it is normally due to errors in journaling the event.

End of Update Recap (*for ACF2*)

The following reporting came from an ACF2 Update.

The basic reporting is the same for all RSS processing, except journal data itself is different. This means data is tracked and accounted for in a manner suitable for the particular RSS.

Information for RACF is in the next section of this chapter.

ACF2 Update Recap Statistics

```
ACF2 UPDATE RECAP STATISTICS

  1)  ..........0   ACF2 SMF RECORDS WITH INVALID DOMAINS
  2)  ..........0   ACF2 SMF SECURITY RECORDS NOT PROCESSED
  3)  ......76,632  ACF2 SMF SECURITY RECORDS SELECTED FOR PROCESSING
```

This is the start of the Update Recap and is RSS dependent. This example is for ACF2. RACF has similar statistics and understanding what is being reported normally applies to any RSS.

The first statistic reports the number of events that had DOMAINS not defined on the particular E-SRF Masterfile being updated. This means you presented data that contained DOMAINS (sysids) what were not established via the ASSIGN specification to particular IMAGES contained on the Masterfile.

The second statistic (in ACF2) means a particular sub-type was found on the update file that is unknown to E-SRF. A hexadecimal printout of the first sixteen of these will occur on the control report. This simply means an error exists on either the update file or the RSS developer created an additional journal subtype that E-SRF does not know about. Please report this to EKC Technical Support so we can provide support for the new sub-type.

The third statistic identifies the number of records the update overlay selected for processing.

ACF2 SMF Sub-Type Record Statistics

```
ACF2 SMF SUB-TYPE RECORD STATISTICS

  4)  ..........0   UNKNOWN SMF SUB-FUNCTION
  5)  ..........0   NON APPLICABLE SMF SUB-FUNCTION

  6)  .........263   TYPE A: ACF2 CONSOLE JOURNAL
  7)  .........263           RECORDS PROCESSED

  8)  ..........0   TYPE C: ACF2 VM COMMAND SERVICES
  9)  ..........0           RECORDS PROCESSED

 10)  ......59,029   TYPE D: DATASET LOG/VIOLATION JOURNAL
 11)  ......20,108           ACF2 TRACE RECORDS PROCESSED
 12)  ......38,921           RECORDS PROCESSED
 13)  ..........0            ETFA TEST RECORDS PROCESSED

 14)  .......1,542   TYPE E: INFO STORAGE DATABASE UPDATES
 15)  ..........0            SKIPPED: NO LOGONID SUPPLIED
 16)  ..........0            SKIPPED: NO INFO-STORAGE RECORD CLASS
 17)  ..........0            SKIPPED: UNKNOWN ACTION TYPE
 18)  ..........0            SKIPPED: DUE TO PROGRAM REQUEST
 19)  .......1,542           RECORDS PROCESSED

 20)  ..........0   TYPE G: INVALID GSO RECORD
 21)  ..........0           RECORDS PROCESSED

 22)  ..........0   TYPE I: INTERCEPT JOURNAL
 23)  ..........0           RECORDS PROCESSED

 24)  .......3,581   TYPE J: RESTRICTED LOGONID
 25)  .......3,581           RECORDS PROCESSED
 26)  .........328           UNIQUE LOGONIDS DETECTED

 27)  .......7,283   TYPE L: LOGONID MAINTENANCE JOURNAL
 28)  ..........0            SKIPPED: NO ACTION
 29)  ..........0            SKIPPED: INVALID RECORD DETECTED
 30)  ..........0            NO 'LIDREC' IMAGE PRESENTED
 31)  ..........24           LOGONID MAINTANANCE
 32)  ..........0            ... TRANSACTIONS SKIPPED DUE TO ZERO CHG/NOCHG COUNTS
 33)  ..........24           ... TRANSACTIONS CONTAINING 'ITEMS NOT CHANGED'
 34)  ..........24           ... AGGREGATE NUMBER OF 'ITEMS NOT CHANGED'
 35)  .......7,259           SIGNON (SEV) REQUESTS: (VALID LOGONID)
 36)  .......3,581           SEV REQUESTS CONVERTED TO RESTRICTED

 37)  ..........0   TYPE O: MVS OPEN EDITION JOURNALS
 38)  ..........0           SKIPPED: DATA FORMATS UNAVAILABLE
 39)  ..........0           JOURNALS PROCESSED

 40)  .......4,403   TYPE P: INVALID PASSWORD/AUTHORITY
 41)  ..........0            SKIPPED: NO LOGONID
 42)  ..........0            SKIPPED: DUE TO PROGRAM REQUEST
 43)  .......1,965           LOGONID ERRORS PROCESSED
 44)  .......2,438           OTHER ERRORS PROCESSED

 45)  ..........0   TYPE R: RULE INSERT/UPDATE/DELETE
 46)  ..........0           SKIPPED: NO LOGONID
 47)  ..........0           SKIPPED: NO ACTION
 48)  ..........0           RECORDS PROCESSED

 49)  ..........0   TYPE S: ACF2 'SECTRACE' JOURNALS
 50)  ..........0           SKIPPED: DATA FORMATS UNAVAILABLE
```

```
  51)  ..........0           JOURNALS PROCESSED

  52)  ..........0   TYPE T: TSO TRANSACTION JOURNAL
  53)  ..........0           RECORDS PROCESSED

  54)  .........531   TYPE V: RESOURCE LOG/VIOLATION JOURNAL
  55)  .........111           ACF2 TRACE RECORDS PROCESSED
  56)  .........420           RECORDS PROCESSED
  57)  ..........0           'SHORT RESOURCE NAME AREA USED
  58)  .........531           'LONG' RESOURCE NAME AREA USED
  59)  ..........0           ETFA TEST RECORDS PROCESSED

  60)  ..........0   TYPE Z: DISTRIBUTED DATABASE FUNCTION

  61)  ..........0   TYPE X'85': ETF/A JOURNALLED EVENTS
  62)  ..........0           UNKNOWN EVENT TYPES SKIPPED
  63)  ..........0           FIRECALL EVENTS SKIPPED
  64)  ..........0           FIRECALL RECORDS PROCESSED

  65)  ..........0   SMF/80:   NUMBER OF SMF TYPE 80 RECORDS DETECTED
  66)  ..........0             RECORD TYPES NOT SUPPORTED

  67)  .........277   EVENT 02: RESOURCE ACCESS
  68)  .........277             RECORDS PROCESSED
  69)  .........167             DATASET ACCESS
  70)  .........110             NON-DATASET RESOURCE ACCESS
  71)  ..........0             ETF/A FIRECALL INVOLVEMENT
  72)  .........277             E-PAL CONTROLLED ACCESS
  73)  ..........0             → UNKNOWN PROCESSING REQUEST

  74)  .......2,455   RESULTING TRANSACTIONS FLUSHED BY THE ESRF MASTERFILE UPDATE PROCESSOR

E705-ACFU DAILY EVENT UPDATE HAS BEEN COMPLETED
```

This information is purely related to the RSS you have just updated. This is an ACF2 example.

Statistic line 4 shows how many unknown SMF types were detected.

Statistic line 5 shows how many unknown SMF sub-types were detected.

Statistic lines 6 through 58 report on how many times the ACF2 Update Overlay detected and processed the various ACF2 journal record types. Typically, you have the number of records detected, special considerations that normally causes records to be skipped, followed by how many were actually applied to the Masterfile. Normally, it is DETECTED, PROCESSED or SKIPPED. Certain types have various reasons for skipping activity.

If a record is redundant, incorrectly formatted or simply not needed for the type of reporting provided by E-SRF, the Update Overlay normally skips the record.

Statistic 65 shows how many standard security journal records (SMF type 80 records) were processed. Normally RACF creates these records and ACF2 does not. They may also be created by security product extensions that may be in place in your installation which run with ACF2. If you are not running any of these products, you probably will never process any.

The ACF2 Update Overlay processes a very small subset of these records. Statistic 64 shows how many were presented. Statistic 65 shows how many the Update Overlay seemed to find useful.

Current support only provides for event type 02 processing. Statistics 65 through 70 provide information showing what the Update Overlay did with the event 02 records.

Certain activity may be "flushed". Userid events with no userid, Grouping rules may identify certain resources that are not to be maintained on the Masterfile, or residual update activity that cannot be applied due to errors may end up. The number of transactions flushed by the Update Overlay is reported in statistic 71.

End of Update Recap for RACF

The following reporting came from a RACF Update.

The basic reporting is the same for all RSS processing, except journal data itself is different. This means data is tracked and accounted for in a manner suitable for the particular RSS.

Information for ACF2 is in the previous section of this chapter.

RACF Update Recap Statistics

```
RACF UPDATE RECAP STATISTICS

   1)  ..........0   RACF SMF RECORDS WITH INVALID DOMAINS
   2)  ..........0   RACF SMF RECORDS WITH SETUP ERRORS
   3)  ..........0   RACF SMF SECURITY RECORDS NOT PROCESSED
   4)  ......13,593  RACF SMF SECURITY RECORDS SELECTED FOR PROCESSING

   5)  .........24   UNIQUE USERIDS PROCESSED (USERID + DOMAIN)
```

This is the start of the Update Recap and is RSS dependent. This example is for RACF. ACF2 has similar statistics and understanding what is being reported normally applies to any RSS.

The first statistic reports the number of events that had DOMAINS not defined on the particular E-SRF Masterfile being updated. This means you presented data that contained DOMAINS (sysids) what were not established via the ASSIGN specification to particular IMAGES contained on the Masterfile.

The second statistic (in RACF) means a particular event type was found on the update file that is unknown to E-SRF or was in a format that the journal setup processor could not properly process. A hexadecimal printout of the first sixteen of these will occur on the control report. This simply means either an error exists on the update file or the RSS developer created an additional journal event type that E-SRF does not know about. Please report this to EKC Technical Support so we can provide support for the new sub-type.

The third statistic identifies the number of records the update overlay selected for processing.

The fourth statistic shows how many journal records the RACF Update Overlay selected for processing.

The fifth statistic shows how many unique uses were detected during the update process.

RACF SMF Event Type Record Statistics

```
RACF SMF SUB-TYPE (EVENT) RECORD STATISTICS

   6)  ..........0   UNKNOWN RACF EVENTS DETECTED
   7)  .........129  EVENTS SKIPPED DUE TO PROGRAM REQUEST
   8)  ..........0   NON APPLICABLE SMF SUB-FUNCTION

   9)  .....100,439  EVENT 01: LOGON/LOGOFF
  10)  .....100,439            RECORDS PROCESSED
  11)  ......49,901            SYSTEM SIGNON VALIDATIONS
  12)  ......43,804            SYSTEM SIGNOFF REQUESTS
  13)  .......6,734            INVALID USERIDS DETECTED

  14)  .......6,171  EVENT 02: RESOURCE ACCESS
  15)  .......6,171            RECORDS PROCESSED
  16)  .......3,336            DATASET ACCESS
  17)  .......2,835            NON-DATASET RESOURCE ACCESS
  18)  ..........0             ETF/R FIRECALL INVOLVEMENT
  19)  ..........0             E-PAL CONTROLLED ACCESS

  20)  ..........0  EVENT 03: ADDVOL/CHGVOL REQUESTS
  21)  ..........0            RECORDS PROCESSED
  22)  ..........0            DATASET REQUEST
  23)  ..........0            NON-DATASET RESOURCE REQUEST

  24)  ..........1  EVENT 04: DATASET RENAME REQUESTS
  25)  ..........0            RECORDS PROCESSED
  26)  ..........0            DATASET RENAME
  27)  ..........0            NON-DATASET RESOURCE RENAME

  28)  .........98  EVENT 05: RESOURCE DELETE REQUESTS
  29)  ..........0            RECORDS PROCESSED
  30)  ..........0            DATASET DELETE
  31)  ..........0            NON-DATASET RESOURCE DELETE

  32)  ..........0  EVENT 06: DELETE SINGLE OR MULTI-VOLUME RESOURCE
  33)  ..........0            RECORDS PROCESSED
  34)  ..........0            DATASET DELETE
  35)  ..........0            NON-DATASET RESOURCE DELETE

  36)  .........31  EVENT 07: DEFINE RESOURCE REQUESTS
  37)  ..........0            RECORDS PROCESSED
  38)  ..........0            DATASET ACCESS
  39)  ..........0            NON-DATASET RESOURCE ACCESS

  40)  ..........7  EVENT 08: COMMAND: ADDSD
  41)  ..........7            RECORDS PROCESSED

  42)  ..........2  EVENT 09: COMMAND: ADDGROUP
  43)  ..........2            RECORDS PROCESSED

  44)  .........48  EVENT 10: COMMAND: ADDUSER
  45)  .........48            RECORDS PROCESSED

  46)  ..........0  EVENT 11: COMMAND: ALTDSD
  47)  ..........0            RECORDS PROCESSED

  48)  ..........0  EVENT 12: COMMAND: ALTGROUP
  49)  ..........0            RECORDS PROCESSED

  50)  .........265  EVENT 13: COMMAND: ALTUSER
```

```
 51)  .........265            RECORDS PROCESSED

 52)  ..........38   EVENT 14: COMMAND: CONNECT
 53)  ..........38             RECORDS PROCESSED

 54)  ...........4   EVENT 15: COMMAND: DELDSD
 55)  ...........4             RECORDS PROCESSED

 56)  ...........0   EVENT 16: COMMAND: DELGROUP
 57)  ...........0             RECORDS PROCESSED

 58)  .........200   EVENT 17: COMMAND: DELUSER
 59)  .........200             RECORDS PROCESSED

 60)  ...........0   EVENT 18: COMMAND: PASSWORD
 61)  ...........0             RECORDS PROCESSED

 62)  .........106   EVENT 19: COMMAND: PERMIT
 63)  .........106             RECORDS PROCESSED

 64)  ...........0   EVENT 20: COMMAND: RALTER
 65)  ...........0             RECORDS PROCESSED

 66)  ...........3   EVENT 21: COMMAND: RDEFINE
 67)  ...........3             RECORDS PROCESSED

 68)  ...........0   EVENT 22: COMMAND: RDELETE
 69)  ...........0             RECORDS PROCESSED

 70)  ...........0   EVENT 23: COMMAND: REMOVE
 71)  ...........0             RECORDS PROCESSED

 72)  ...........9   EVENT 24: COMMAND: SETROPTS
 73)  ...........9             RECORDS PROCESSED

 74)  ...........0   EVENT 25: COMMAND: RVARY
 75)  ...........0             RECORDS PROCESSED

 76)  ...........0   SMF TYPE 20: JOB INIT RECORDS
 77)  ...........0            SKIPPED: NO RACF RELOCATE AREA
 78)  ...........0            SKIPPED: EMPTY RACF RELOCATE AREA
 79)  ...........0            SKIPPED: USERID NOT AVAILABLE
 80)  ...........0            WARNING: USERID '*' DETECTED
 81)  ...........0            JOB INIT RECORDS PROCESSED

 82)  ...........0   SMF TYPE 30: COMMON ADDRESS SPACE WORK
 83)  ...........0            SKIPPED: NOT SUBTYPE 4 (STEP TOTAL)
 84)  ...........0            SKIPPED: ZERO IDENTIFICATION OFFSET
 85)  ...........0            SKIPPED: ZERO IDENTIFICATION LENGTH
 86)  ...........0            SKIPPED: NOT FIRST STEP
 87)  ...........0            SKIPPED: MISSING USERID
 88)  ...........0            WARNING: USERID '*' DETECTED
 89)  ...........0            COMMON ADDRESS SPACE WORK RECORDS PROCESSED

 90)  ...........0   SMF TYPE 230: ETF/R JOURNALLED EVENTS
 91)  ...........0            SKIPPED: UNKNOWN EVENT TYPES
 92)  ...........0            SKIPPED: DUE TO PROGRAM REQUEST
 93)  ...........0            FIRECALL RECORDS PROCESSED
```

This information is related to the RSS you have just updated. This is a RACF example.

Statistic line 6 shows how many unknown SMF event types were detected.

Statistic lines 7 and 8 show how many SMF sub-types that were skipped over due to program request. There are many reasons for this, duplicate data, improperly formatted data, and data that was not relevant to the required processing.

Statistic lines 9 through 75 report on how many times the RACF Update Overlay detected and processed the various RACF journal event types. Typically, you have the number of records detected, special considerations that normally causes records to be skipped, followed by how many were actually applied to the Masterfile. Normally, it is DETECTED, PROCESSED or SKIPPED. Certain types have various reasons for skipping activity.

If a record is redundant, incorrectly formatted or simply not needed for the type of reporting provided by E-SRF, the Update Overlay normally skips the record.

Statistic 76 through 80 shows how many JOB INIT SMF records processed. We do not recommend the use of these records. Whether or not to use these records is controlled by Update Function parameters. Despite the fact that these records are extremely efficient, they simply do not contain enough information to be useful for this process.

Statistic 82 through 89 shows how many Common Address Space work records were processed. These records contain everything the RACF Update Overlay requires, except the only one required is the first one for each job. You end up with at least one for each step. The Update function uses what it needs and skips the rest.

Our SMF file did not have any and reported all zeros. This is not typical because we recommend these records be present. The following below shows what these statistics could look like had the proper SMF records been provided:

```
82)  ......10,283   SMF TYPE 30: COMMON ADDRESS SPACE WORK
83)  .......5,099           SKIPPED: NOT SUBTYPE 4 (STEP TOTAL)
84)  ...........0           SKIPPED: ZERO IDENTIFICATION OFFSET
85)  ...........0           SKIPPED: ZERO IDENTIFICATION LENGTH
86)  .......2,900           SKIPPED: NOT FIRST STEP
87)  ...........0           SKIPPED: MISSING USERID
88)  .........187           WARNING: USERID '*' DETECTED
89)  .......2,284           COMMON ADDRESS SPACE WORK RECORDS PROCESSED
```

Statistic 90 through 93 shows how many user Type 230 records were processed by this Update. Despite the fact that this is RACF, these user 230 (*the same type ACF2 creates*) are produced by certain security extension products. If you get them it means you may be running one or more of these products.

Domain Summary

```
E734-$UCP DOMAIN SUMMARY FOR THIS UPDATE FUNCTION

DOMAIN    ASSIGNED  IMAGE     RSS           RECORDS  ERROR/EXCEPTIONS
--------  --------  --------  --------  -----------  ----------------
CHGO      CHGO****  CHICAGO   ACF2      .....76,632

E735-$UCP END OF DOMAIN SUMMARY
```

This section recaps the distribution of update activity to the various domains on the Masterfile. In our sample, which was from the ACF2 example, we processed an update file that contained SMF data for a single domain (CHGO). This display shows 76,632 update records were applied to the CHGO domain, which is a domain “assigned” to the ‘CHICAGO’ security IMAGE.

This example came from our ACF2 example. Our RACF update looked like this except it showed the names of the particular image and domain associated with the particular system that the journals came from with the record count of that particular set of journals.

Masterfile Update General Statistics

```
E-SRF MASTERFILE UPDATE GENERAL STATISTICS

  1)  ......77,146   RAW SMF RECORDS READ
  2)  ......76,632   ... SECURITY SMF RECORDS SELECTED

  3)  ......80,358   RESOURCE EVENTS ANALYZED FOR PROCESSING
  4)  ......11,569   ... GDG QUALIFIERS TRUNCATED
  5)  ..........0   ... MASKED QUALIFIER COMPRESSIONS PERFORMED
  6)  ..........0   ... DROPPED DUE TO OMITTED RESOURCE NAME
  7)  ..........0   ... GROUP EXCLUDE: (SYSTEM ENTRY EVENTS)
  8)  ..........0   ...            ... (LOGGED EVENTS)
  9)  ..........0   ...            ... (SPECIAL LOGGED EVENTS)
 10)  ..........0   ...            ... (VIOLATION EVENTS)
 11)  ..........0   TOTAL RESOURCE EVENTS DROPPED
 12)  ......80,358   TOTAL RESOURCE EVENTS RETAINED FOR PROCESSING
 13)  .......2,816   TOTAL RESOURCE NAMES ALTERED

 14)  .....491,381   TOTAL TRANSACTIONS PRESENTED FOR UPDATE

 15)  .......3,826   EVENT STEP DOWN SITUATIONS DETECTED DURING PROCESSING
 16)  .......3,826   ... EVENT STEP DOWN SITUATIONS PROCESSED

 17)  .......7,283   USERID HEADER STEP DOWNS DETECTED AND REJECTED

 18)  ..........0   EVENTS CONTAINING NO USERID SKIPPED
 19)  ..........2   DUPLICATE EVENTS SKIPPED
 20)  .......2,453   EVENTS SKIPPED DUE TO CLOSE INTERVAL

 21)  ..........0   TRANSACTIONS REJECTED DUE TO OBJECT RETENTION SET TO ZERO
 22)  .......2,455   TRANSACTIONS REJECTED DUE TO PROCESSING REQUEST
 23)  ..........0   RESOURCE TRANSACTIONS REJECTED DUE TO @CONTROL CLASS

 24)  ......24,581   HIGHEST NUMBER OF ELEMENTS FOUND ON UPDATED OBJECT
 25)  ...4,375,504   LARGEST NUMBER OF BYTES FOUND ON UPDATED OBJECT

 26)  ..........0   UNKNOWN UID INFORMATION RESOLUTIONS PERFORMED
 27)  ..........0   ... (RESOLVED BY EXAMINING USER HEADER)
 28)  ..........0   ... (RESOLVED BY REFERENCING LOOK-ASIDE)
 29)  ..........0   ... (UNKNOWN USERID UNABLE TO RESOLVE)
```

These statistics are provided by the E-SRF “Update Function” (from an ACF2 update).

Statistic 1 shows how many “RAW” SMF records were read into this process. This is a gross record count and mak contain records that will not get processed by this system.

Statistic 2 shows how many of these “RAW” SMF records were actually selected by the Update Function for processing.

> Records processed include the actual journals created by the security system as well as other system journals that may be required to augment this process.

Statistic 3 shows how many *RESOURCE* events that were detected and analyzed by the Update Function.

> Results of this analysis are shown in statistics 4 through 14.

Statistic 4 shows how mangy GDG dataset names were ‘compressed’ down to their base names followed by ‘G00V0000’ as their last prefix.

Statistic 5 shows how many ‘masked qualifiers’ were substituted based on your definitions.

UPDATE Control Report

Statistics 7 through 10 show what was dropped based on 'Group Exclude". These excludes were defined in your grouping rules instructing the Update Function to exclude from the Masterfile.

Statistic 11 shows the total number of events that were dropped for all reasons above.

Statistic 12 shows the total number of events that were retained and processed.

Statistic 13 shows the total number of resource names that were altered based on your update options.

> This would include GDG truncations, mask hits, and names that were invalid and were altered simply to be able to store them and access them in subsequent reports.

Statistic 14 shows how many actual transactions were presented for Masterfile updating. (A single security journal normally generates several update transactions).

Statistic 15 shows how many EVENT *stepdowns* were detected for Masterfile updating.

> A stepdown is detected when the date and time of the current event is less than the previous event processed.

Statistic 16 shows how many EVENT *stepdowns* were allowed to process.

> Your options control this. Normally, event stepdowns are OK and should be allowed to process. This number should be the same as statistic 15 (above), unless you do not allow stepdown processing.

Statistic 17 shows User Id stepdowns.

> These are not permitted but can and will occur. The event containing the User ID stepdown is processed (based on your stepdown options). The updating of the User ID Header (UA) Masterfile object is not performed because it would degrade the User ID data currently stored on the Masterfile with older data. For example, If you change userid header information (such as NAME), you replace what was there with the update. You only want the latest change to remain on this object after the update was completed.

Statistic 18 shown how many updates required a userid and one was not found.

Statistic 19 shows how many DUPLICATE transactions were dropped.

Statistic 20 shows how many events that will be treated as a duplicate were dropped.

> They are considered duplicate because they were the same except the time was within twenty seconds. Consider a TSO signon. There are always more than one system entry validation (SEV) call to get a TSO signon up and running. E-SRF reports this as only a single event.

Statistic 21 reports how many transactions were dropped because the retention for the targeted object was set to zero days.

Statistic 22 shows how many transactions were rejected due to programming request. Normally, this means they contain information not required for any processing or the data was illogical for the event being reported.

Statistic 23 shows how many transactions were rejected because the class was set to @CONTROL.

> This is reserved for E-SRF internal use and only certain activity actually gets stored on the Masterfile from this class during a normal Update Function.

Statistic 24 shows the maximum number of elements found on an object THAT WAS A TARGET FOR UPDATE in this particular update.

Statistic 25 shows the size of the largest object (in bytes) THAT WAS A TARGET FOR UPDATE in this particular update. Note: statistic 24 and 25 do not have to be reporting on the same object.

Statistics 26 to 29 indicate problems determining the E-SRF UID (User Identification) for particular events.

> The E-SRF UID is made up of data from the RSS and is used to assist in grouping data and help in the production of reports. Normally this data can be assembled by data contained in the event journal. If the data is not present in the event journal, attempts are made to locate it from another source.

Unexpired Rolloff Situations

```
UNEXPIRED EVENT ROLLOFF SITUATIONS


30)  .........135   UNEXPIRED ROLLOFF SITUATIONS ENCOUNTERED
31)  ......55,215   TOTAL UNEXPIRED EVENTS ROLLED OFF MASTERFILE

32)  ...........0   ... (CONSOLE CHRONOLOGICAL EVENTS)

33)  ......21,677   ... (RESOURCE CHRONOLOGICAL EVENTS)
34)  ...........0   ... (RESOURCE MAINTENANCE LOG EVENTS)
35)  ...........0   ... (RESOURCE RECAP EVENTS)
36)  ...........0   ... (RESOURCE STATISTICAL EVENTS)

37)  ...........0   ... (SOURCE RECAP EVENTS)
38)  ...........0   ... (SOURCE INVALID USERIDS)

39)  ...........0   ... (USER ADMINISTRATIVE EVENTS)
40)  ......25,358   ... (USER CHRONOLOGICAL EVENTS)
41)  ...........0   ... (USER FIRECALL EVENTS)
42)  ...........0   ... (USER MAINTENANCE LOG EVENTS)
43)  ...........0   ... (USER RECAP EVENTS)
44)  .......8,180   ... (USER RSS TRACE EVENTS)
45)  ...........0   ... (USER STATISTICAL)
```

This recap shows how many UNEXIPRED ROLLOFFS occurred during this update run.

UNEXPIRED ROLLOFFS represent data that is discarded from the Masterfile that is really not expired and normally you would want to retain.

UNEXPIRED ROLLOFFS represent the oldest elements that are dropped from Masterfile objects simply because there were more current elements being applied to the Masterfile than the ELEMENTS specification would allow. When this occurs, ten percent of the oldest data is dropped from the particular object to make room for more current data. These dropped elements are what are being reported on in this section.

This is a normal situation. If you investigate you will find that either your ELEMENTS value is too small, or you have rules that are logging access where a large number of accessed occur.

Normally, excessive loggings are one of the problems with security logging and needs to be addressed by careful analysis of your logging requirements. Simply reconfiguring your Masterfile to hold more data may not be the best solution to this situation.

Expired events not posted to the Masterfile

```
EXPIRED EVENTS NOT POSTED TO MASTERFILE

 46)  ..........0   TOTAL NUMBER OF EXPIRED EVENTS DETECTED

 47)  ..........0   ... (CONSOLE CHRONOLOGICAL EVENTS)

 48)  ..........0   ... (RESOURCE CHRONOLOGICAL EVENTS)
 49)  ..........0   ... (RESOURCE MAINTENANCE LOG EVENTS)
 50)  ..........0   ... (RESOURCE RECAP EVENTS)
 51)  ..........0   ... (RESOURCE STATISTICAL EVENTS)

 52)  ..........0   ... (SOURCE RECAP EVENTS)
 53)  ..........0   ... (SOURCE INVALID USERIDS)

 54)  ..........0   ... (USER ADMINISTRATIVE MAINTENANCE EVENTS)
 55)  ..........0   ... (USER CHRONOLOGICAL EVENTS)
 56)  ..........0   ... (USER FIRECALL LOG EVENTS)
 57)  ..........0   ... (USER MAINTENANCE LOG EVENTS)
 58)  ..........0   ... (USER RECAP EVENTS)
 59)  ..........0   ... (USER STATISTICAL EVENTS)
 60)  ..........0   ... (USER RSS EVENT TRACE EVENTS)
```

EXPIRED ROLLOFFS are events that have dates that are older than what is current on the specific Masterfile object being updated. They do not fall within the RETAIN specifications for the targeted object.

This recap shows how many EXIPRED ROLLOFFS occurred during this update run. In this example, there were none because nothing on the Masterfile was expired. This will occur until you run the Update Function for a period of time that begins to exceed the RETAIN specifications for your objects..

Masterfile event update statistics

```
MASTERFILE EVENT UPDATE STATISTICS

 61)  .....326,217   TOTAL MASTERFILE EVENT UPDATES

 62)  .......2,972   ... (NEW OBJECTS ADDED TO MASTERFILE)

 63)  .........263   ... (CONSOLE CHRONOLOGICAL EVENTS)
 64)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 65)  ......39,339   ... (RESOURCE CHRONOLOGICAL EVENTS)
 66)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 67)  .......1,542   ... (RESOURCE MAINTENANCE LOG)
 68)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 69)  ......39,339   ... (RESOURCE RECAP EVENTS)
 67)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 71)  ......39,339   ... (RESOURCE STATISTICAL EVENTS)
 72)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 73)  ......48,548   ... (SOURCE RECAP EVENTS)
 74)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 75)  .......1,965   ... (SOURCE INVALID USERIDS)
 76)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 77)  ...........0   ... (USER HEADER INFORMATION)
 78)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST
 79)  ...........0       +ACTIVE USERS DELETED
 80)  ...........0       +DELETED USERS REACTIVATED

 81)  ...........0   ... (USER HEADER SELECTIVE FIELD UPDATE)
 82)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST
 83)  ...........0       +USER SHELLS INSERTED ON MASTERFILE
 84)  ...........0       +ACTIVE USERS DELETED
 85)  ...........0       +DELETED USERS REACTIVATED

 86)  ...........0       +INDIVIDUAL FIELD UPDATE REQUESTS MADE
 87)  ...........0       +INDIVIDUAL FIELDS UPDATED
 88)  ...........0       +FIELDS NOT UPDATED WITH DUPLICATE DATA
 89)  ...........0       +FIELDS NOT UPDATED DUE TO STEPDOWN
 90)  ...........0       +FIELDS NOT UPDATED DUE INAPROPORATE DATA
 91)  ...........0       +FIELDS NOT UPDATED DUE TO ERROR

 92)  .......1,566   ... (USER ADMINISTRATIVE EVENTS)
 93)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 94)  ......46,583   ... (USER CHRONOLOGICAL EVENTS)
 95)  .......2,455       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 96)  ...........0   ... (USER FIRECALL LOG)
 97)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 98)  ..........24   ... (USER MAINTENANCE LOG)
 99)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

100)  ...........0   ... (USER RSS PROFILE INFORMATION)
101)  ...........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST
```

```
 102)  ..........0       +INDIVIDUAL FIELD UPDAE REQUESTS MADE
 103)  ..........0       +INDIVIDUAL FIELDS UPDATED
 104)  ..........0       +FIELDS NOT UPDATED WITH DUPLICATE DATA
 105)  ..........0       +FIELDS NOT UPDATED DUE TO STEPDOWN
 106)  ..........0       +FIELDS NOT UPDATED DUE INUPROPORATE DATA
 107)  ..........0       +FIELDS NOT UPDATED DUE TO ERROR

 108)  ......46,583   ... (USER RECAP EVENTS)
 109)  ..........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 110)  ......39,339   ... (USER STATISTICAL EVENTS)
 111)  ..........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST

 112)  ......20,219   ... (USER RSS TRACE EVENTS)
 113)  ..........0       +OTHERS SKIPPED DUE TO PROCESSING REQUEST
```

When the Update Overlay normalized a security event, it breaks the event down into individual transactions, which are used to update the various objects contained on the Masterfile.

This section recaps the distribution of transactions across the targeted Masterfile objects and how the Update Services components acted with the transactions.

Unexpired Rolloff object table

```
E727-$UCP START OF UNEXPIRED ROLLOFF OBJECT TABLE

DATASET  PROD.SYS.CONTROL........................... (R,C)  ......4,499
DATASET  EXR50.IIO.MODLIB........................... (R,C)  .....12,679
DATASET  IMS.TST.TESTLOAD........................... (R,C)  ......4,499
         IAPXP   CHICAGO............................ (U,C)  ......4,499
         IFNK26P CHICAGO............................ (U,C)  .....20,859
         O444    CHICAGO............................ (U,T)  ........818
         USER1   CHICAGO............................ (U,T)  ......7,362

E728-$UCP END OF UNEXPIRED ROLLOFF OBJECT TABLE, OCCURRANCES: 55,215, UNIQUE: 7
```

This recap shows a sample of individual UNEXIPRED rolloffs that could occur during an update run. Because of the low volume of our two examples, neither one actually had any unexpired rolloffs.

Unexpired rolloffs are the oldest elements that are dropped from Masterfile objects simply because there were more current elements being applied to the Masterfile than the ELEMENTS specification would allow. When this occurs, ten percent of the oldest data is dropped from the particular object to make room for more current data. These dropped elements are what are being reported on in this section.

You can use this information to examine the security definitions in an effort to determine if loggings should really be created, or if some of them should be excluded by using other E-SRF controls such as the EXCLUDE grouping rule attribute.

Invalid Userid list

```
E706-$UCP START OF INVALID USERID LIST

  ACF2.... 2      CB8DADSP 1      CB8D01P. 1      CB8D05P. 1      CB8D10P. 1      CB8D20P. 1
  <more invalid users>
  9WSTBC..... 1

E707-$UCP END OF INVALID USERIDS, OCCURRENCES: 1,965, UNIQUE: 1,017
```

When the Update Function detects an Invalid Userid, it does not create a User header for the Userid. If it do, you would end up with a Masterfile full of Invalid Userids created by terminal user 'typos'. Instead the following occurs.

The event will be applied to the SU (Source Userid) object (if it is active).

The event will be posted in this section.

Please note this may be the only place you can do to see this type of error.

This section will list each "pattern" of invalid userid and how many times it occurred for the duration of the current Update Function.

E707 shows how many times invalid userids occurred and how many unique "patterns" were detected.

END OF UPDATE notification.

```
E752-$UCP UPDATE FUNCTION COMPLETED
E418-$MCP MASTERFILE CONTROLS HAVE BEEN UPGRADED
E182-CMD  UPDATE COMPLETED, RETURN CODE: 0
```

E752 indicates the Update Function has been completed.

E418 indicates the status of the last Update Function has been stored on the Masterfile.

E752 shows the highest detected return code from the Update Function just executed.

This entire process will repeat itself each time the UPDATE command is presented. The Masterfile cleanup is however only performed once in a calendar day.

This page intentionally left blank

Chapter 13: Common Datanames

The E-SRF Masterfile contains various objects, which store data related to a particular person, place, or thing. The object, like any other data processing record, contains a KEY. The object's KEY describes what the remaining data is related to. In some objects, the key consists of all fields for that object. In others, it does not.

The term "*RESOURCE*" may be used in many segments to refer to the object type in that segment. For example, it may refer to a particular dataset or resource object maintained in the RESOURCE segment or a userid maintained in the USER segment.

Several datanames are common to all segments. These COMMON datanames may or may not apply in any one segment. In addition, the segments may contain particular datanames that are similar in content to the common datanames.

The key consists of datanames in the following order:

SEGMENT, CLASS, RESOURCE, VOLUME and OBJTYPE

Datanames common to all objects

CLASS	Object Resource Class

Description: Identifies a particular resource class. All dataset resources are classified as DATASET, SAF classes are represented as they are.

ACF2 "generalized resource rule" *types* are stored as three character type codes.

Format: KEY: Appears as an eight-character text field.

COMMENT	Group "comment" text

Description: Provides the optional one to forty-eight character grouping rule comment text associated with the current group. The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. It will only be present if the grouping rule associated with the object's resource name contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing additional information about the object. For example, if the object is in the Resource Segment, you can use this data to provide meaningful names to resources such as CICS and IMS transaction Ids.

Format: ARRAY: Provides reference to forty-eight-character group comment text associated to the specific object key (if available).

GROUP	Group Identification

Description: Provides the Resource Group name for the current object. This group name is dynamically determined during processing. The data presented is the result of interpreting the E-SRF grouping rules in the Resource Grouping Facility. For more information on how E-SRF uses groups, see the section*, "Grouping and Ownership*", in the *User Guide*.

Format: ARRAY: External reference to sixteen-character group name associated to the specific array element within the current object.

OBJTYPE — Object Type Code

Description: Identifies a particular object *type* code within a particular Masterfile segment.

Each segment contains one or more different object *types*. The SEGMENT and OBJECT type identifies the desired target segment.

Format: KEY: Single character object type code. (Dependent on segment code).

Object Identifier	Segment	Type	Description
FC	Console	Chronological	Detail list of console interactions with RSS
GA	Group	Header	Describes a group
MC	Maintenance	Chronological	Describes maintenance events
OA	Owner	Header	Describes an owner
RC	Resource	Chronological	Detail list of events
RM		Maintenance	Changes made to a resource
RR		Recap	Daily recap information
RS		Statistical	Daily Resource statistical by user
SR	Source	Recap	Detail list of events
SU		Chronological	Invalid Userid signon attempts
UA	User	Header	Describes a user (RSS)
UB		Sec Admin	Describes administration activity
UC		Chronological	Detail list of events
UF		Firecall	List of events because of Firecall
UM		Maintenance	Changes made to a user
UP		Profile	RACF Connect Group information
UR		Recap	Daily recap information
US		Statistical	Daily User statistical by resource
UT		Trace	List of events from RSS “Trace”

OWNER — Owner associated to current group.

Description: Provides the eight character OWNER id associated to the current GROUP for the current object. For more information, please reference*, “Grouping and Ownership*”, in the *User Guide*.

Format: ARRAY: Eight character OWNER associated to current group at the ARRAY processing level.

RESOURCE — Resource Name

Description: Provides the resource name for the current object. This may have different meaning based on the object’s segment and type.

Format: KEY: Resource name text, maximum of 44 characters (*un-tokenized*) and 1000 characters (*tokenized*), left justified padded with right blanks. This field will “shrink down” if there is not enough room on a page to print the specified line.

NOTE: Resource names contained on the Masterfile KEY are 44 characters long. Actual resource names are from 1 to 1000 characters in length.

In the case of resource names that actually refer to "resources", the KEY will contain a resource TOKEN. This token will represent the actual resource name that is physically stored in the Resource Object Token Dictionary. This is done automatically by E-SRF and you do not have to concern yourself with this, other than how big this data can actually be with respect to the appearance of your reports.

The following pertains to all datanames that refer to actual resource names:

All resource names are tokenized. Because such a large amount of data can be represented by a single data name, various report overlays may format the data in different ways:

In the CASE of ESRFLIST, it is impossible to contain a 1000 character field. As much room as possible will be allocated to the column. A *defacto* 44-character length is used and will be increased if extra room is available so the field can be as long as possible. As in previous releases, right truncation may and probably will occur.

If actual text ends up being truncated, a "*footnote*" will be appended to what is formatted in the column that will refer to the fully spelled out name appearing at the end of the report.

In the case of ESRFDXD, there is no truncation. If NOTRIM is set, the column will be 1000-characters wide.

Other report overlays attempt to format this data the best way possible.

Resource Token on Masterfile Key

Description: Provides actual token that may be associated with this object as a resource key.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this data name relates to the tokenized CLASS, RESOURCE and VOLUME of the Masterfile key. Only certain segments contain tokenized Masterfile keys. As of release 2.1, only the resource segment contains tokenized Masterfile Keys.

SEGMENT — Object Segment Code

Description: The E-SRF Masterfile is divided up into *segments*. A segment may be viewed as a "file within a file". Each segment contains objects that relate to a particular kind of data. For example, data that is *user related* is stored in the USER segment, data that is *resource related* is stored in the RESOURCE segment. Each segment has its own objects, which have their own format.

Format: KEY: Single character object segment code.

Segment Identifier	Segment Name	Description
C	Control	Contains E-SRF system control data areas. This segment is not accessible by users of E-SRF.
F	Console	Describes events related to the interaction of the Resident Security System and the Operating System's master console.
G	Group	Defines and describes E-SRF *group* properties.
H	Owner	Defines and describes E-SRF *owner* properties.
M	Maintenance	Details a list of maintenance operations for a particular resource or user.
R	Resource	Contains various objects that provide data related to particular resources (*things*).
S	Source	Summarizes events that occurred on particular sources (*places*).
U	User	Contains various objects that provide data related to particular users (*people*).

VOLUME — Object Resource Volume Information

Description: Identifies the VOLUME information for a particular resource. This is a carryover from DASD (Direct Access Storage Device) and Magnetic Tape storage media.

Format: KEY: Appears as a six-character text field.

Chapter 14: Console Segment Datanames

The CONSOLE segment stores information about console activity logged by a Resident Security System (RSS). These events normally relate to communications between the console operator and the RSS control programs.

NOTE: You may substitute "CONSOLE" for "FC" if desired.

Related to all CONSOLE Objects

FC.SYSTEM	Console event hosting system ID

Description: Identifies the system (SYSID) that hosted the console event.

Format: KEY: Eight character system identification (SYSID) text. In MVS the last four characters are normally blank.

Related to CONSOLE Chronological Object

The CHRONOLOGICAL object type consists of array elements describing particular console events. There are as many array elements as there were console events for the number of retention days specified for this object type.

The key for this object is: FC.SYSTEM

FC.COMMAND	Operator command issued

Description: Identifies a particular operator command issued against the Resident Security System.

Format: ARRAY: Appears as an eight-character text field.

FC.CONS-ID	Console status: console ID supplied

Description: Indicates whether the MVS console identifier was supplied with the console event logging.

Format: ARRAY: Condition on or off state, appears as "YES / NO" text.

FC.CONSOLE	MVS console identification

Description: Identifies the MVS console identifier used to perform the console operation.

Format: ARRAY: Stored as a binary numeric value, appears as a numeric value.

FC.DATE	Console event date

Description: Date console event took place.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

FC.DOMAIN **Console event hosting domain (sysid)**

Description: Domain (sysid) console event took place on.

Format: ARRAY: Stored as character data.

FC.GOOD **Console status: Good**

Description: Indicates whether or not console operation was successful.

Format: ARRAY: Condition on or off state, appears as “YES / NO” text.

FC.IMAGE **Console event hosting domain (sysid)**

Description: Security image relative to domain where console event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

FC.INITIAL **Console status: Initial operator input**

Description: Indicates whether or not this event contained the initial operator input.

Format: ARRAY: Condition on or off state, appears as “YES / NO” text.

FC.LOGONID **Console event Logonid**

Description: Identifies the USERID (Logonid) who maintained the console dialog. This information is not always present. *This dataname represents the same field as “CONSOLE.USERID”.*

Format: ARRAY: Eight character Userid text.

FC.PARMS **Console Status: Parameters used**

Description: Indicates whether or not console parameters were used during the processing of the current console event.

Format: ARRAY: Condition on or off state, appears as “YES / NO” text.

FC.TEXT **Console parameter text**

Description: Console event parameter text as presented to the operating system.

Format: ARRAY: One to sixty-four character parameter data text.

FC.TIME **Console event time**

Description: Time console event took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

FC.USERID	**Console event Userid**

Description: Identifies the USERID who maintained the console dialog. This information is not always present. *This dataname represents the same field as "CONSOLE.LOGONID".*

Format: ARRAY: Eight character Userid text.

This page intentionally left blank

Chapter 15: GROUP Segment Datanames

The GROUP segment stores information about E-SRF groups. The Resource Grouping Facility provides a means for "grouping" users, sources, and resources into logical groups. In order to process these groupings for automatic report distribution, the group names must be defined to E-SRF in this segment.

NOTE: You may substitute "GROUP" for "GA" if desired.

Related to all GROUP Objects

GA.ID	Group identification

Description: Contains the group identification name. This is what the group is called and referred to throughout E-SRF.

Format: KEY: Appears as a sixteen-character text field.

Related to GROUP Header Object

The HEADER object type consists of single data items that define a particular group.

The key for this object is: GA.ID

GA.APPL	Application name

Description: Identifies a user specified application field assigned to a particular group.

Format: SINGLE: Appears as a twenty-four-character text field.

GA.AUDIT	Audit code

Description: Identifies a user specified audit code to a particular group.

Format: SINGLE: Appears as a single character text field.

GA.CONTACT	Contact name

Description: Identifies a user specified contact name assigned to a particular group.

Format: SINGLE: Appears as a twenty-four-character text field.

GA.CRITICAL	Criticality code

Description: Identifies a user specified criticality code assigned to a particular group.

Format: SINGLE: Appears as a single character text field.

GA.DESC	Group description

Description: Contains user specified text to describe a particular group.

Format: SINGLE: Appears as a twenty-four-character text field.

GA.EMERGENCY **Emergency Contact name**

Description: Contains user specified text to identify an emergency contact for a particular group.

Format: SINGLE: Appears as a twenty-four-character text field.

GA.INTREGTY **Integrity code**

Description: Identifies a user specified integrity code assigned to a particular group.

Format: SINGLE: Appears as a single character text field.

GA.LOB **Line of business code**

Description: Identifies a user specified line of business code assigned to a particular group.

Format: SINGLE: Appears as a single character text field.

GA.NAME **Group name**

Description: Contains user specified text to identify the name of a particular group.

Format: SINGLE: Appears as a twenty-four-character text field.

GA.OWNER **Group OWNER identification**

Description: Contains user specified OWNER identification. There must be an OWNER object present for the specification made here. Group report distribution uses this specification to route reports to. If an owner object is not found, the OWNER "DEFAULT" will be used.

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY1 **Optional interested party 1**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY2 **Optional interested party 2**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY3 **Optional interested party 3**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY4 **Optional interested party 4**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY5 **Optional interested party 5**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY6 **Optional interested party 6**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY7 **Optional interested party 7**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY8 **Optional interested party 8**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY9 **Optional interested party 9**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY10 **Optional interested party 10**

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY11 Optional interested party 11

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY12 Optional interested party 12

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY13 Optional interested party 13

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY14 Optional interested party 14

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY15 Optional interested party 15

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.PARTY16 Optional interested party 16

Description: Contains an additional OWNER identification of an additional owner who really does not own the group, but is interested in receiving a copy of the group's reports. If this field is omitted, it is ignored. The processing rules are the same as "GA.OWNER".

Format: SINGLE: Appears as an eight-character text field.

GA.RETAIN Data retention specification

Description: Number of days data for this group is to be retained in E-SRF.

THIS VALUE IS IGNORED IN THE CURRENT RELEASE OF E-SRF.

Format: SINGLE: Appears as a numeric value.

GA.SENSITVY	**Data sensitivity classification code**

Description: Identifies a user specified sensitivity classification code assigned to a particular group.

Format: SINGLE: Appears as a single character text field.

GA.SYSTEM	**System name**

Description: Contains user specified text to associate a particular group to an application system.

Format: SINGLE: Appears as a twenty-four-character text field.

GA.USERDATA	**Installation required userdata**

Description: Contains userdata that relates to a particular installation. Any type of text may be placed here.

Format: SINGLE: Appears as one to sixty-four character text field.

This page intentionally left blank

Chapter 16: MAINTENANCE Segment Datanames

The MAINTENANCE segment manages information about changes made to Resident Security System access controls. The key consists of the resource CLASS followed by the resource name. In the case of user, the class is USER and the resource name is the userid.

Please note, this segment does not physically exist on the Masterfile. Objects contained in this segment are virtual. The data required is obtained from other objects in other segments on the Masterfile. The actual existence of the data is dependent on the characteristics of the objects that are used to provide the data.

Related to all MAINTENANCE Objects

Common Datanames: CLASS, RESOURCE and VOLUME

Related to MAINTENANCE Chronological Object

The CHRONOLOGICAL object type consists of array elements listing single data items that define a particular group.

Data contained on this object is provided by the Resource Maintenance and User maintenance object. The presence of this data is based on RETAIN DAYS(*nnn*) for these two objects. The fields are the same as these two objects with the exception of the MC prefix.

The key for this object is: CLASS, RESOURCE and VOLUME

MC.CHANGER	Changer's userid

Description: This field contains the user who made the change to the resource being reported on.

Format: ARRAY: Appears as an eight character standard userid.

MC.DATANAME	The name of the data item changed

Description: This field indicates the name of the item that was changed during resource maintenance

If the contents of this item is +ADMIN, this particular event represents a summary of the requested transaction. The detail items that were actually changed on the security database will follow. If all items presented in this transaction caused no change the security database, than this will be the only event that is reported.

Format: ARRAY: eight-character maintenance request data item.

MC.DATE	Time maintenance request event took place

Description: Date console event took place.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

MC.DOMAIN	Maintenance event hosting domain (sysid)

Description: Domain (sysid) where the maintenance event took place on.

Format: ARRAY: Stored as character data.

MC.IMAGE — Maintenance event Security Image ID

Description: Security image relative to domain where maintenance event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

MC.LOGONID — Logonid making the maintenance request event

Description: Identifies the LOGONID (Userid) who applied the maintenance to the target resource. *This dataname represents the same field as "MC.CHANGER".*

Format: ARRAY: Eight character Userid text.

MC.NEWDATA — User maintenance new data

Description: The information shown here is the actual new data related to the "dataname" that was applied to the security database for the named resource. Please note that this data is not always available.

Please note, if RM.DATANAME is +ADMIN, the NEWDATA field is used to indicate the type of request performed, such as INSERT, ALTER or DELETE.

Format: ARRAY: Stored as a twenty-four-character left justified data item.

MC.OLDDATA — User maintenance old data

Description: The information shown here is the actual old data related to the "dataname" that was applied to the security database for the named resource. Please note that this data is not always available.

Please note, if RM.DATANAME is +ADMIN, the OLDDATA field is used to maintain summary statistics on what was actually altered against what was requested and is represented by "CHANGED: *nnn*, NOCHG: *nnn*". The CHANGED number represents how many individual data items listed in the transaction were actually different and changed on the security database. The NOCHG number represents how many individual data items were requested to be changed, but were identical and therefore were not changed. The sum of the two numbers show how many items were presented for change.

Format: ARRAY: Stored as a twenty-four-character left justified data item.

MC.REQUEST — Maintenance event request type

Description: Identifies the type of maintenance performed.

Format: ARRAY: eight-character text field containing request type as specified for the Resident Security System. Please refer to *Appendix G* of this publication for information related to the particular Resident Security System responsible for the maintenance.

MC.RSS — Resident Security System that posted this event

Description: The normalized name of the Resident Security System (RSS) which was responsible for posting this event. . All RSS codes are listed in Appendix A of this guide.

Format: ARRAY: eight-character name of the Resident Security System and its primary component.

MC.SOURCE	**Source maintenance event was performed from**

Description: Identifies the "source" (place) the maintenance was performed on.

Format: ARRAY: eight-character text field (DELETE, INSERT or REPLACE).

MC.SYSTEM	**Maintenance event hosting sysid (Domain ID)**

Description: Sysid (Domain ID) where the maintenance event took place on.

Format: ARRAY: Stored as character data.

MC.TIME	**Time maintenance event was performed**

Description: Time console event took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

MC.TYPE	**Type of maintenance**

Description: Type of Resident Security System data that the maintenance was applied to.

Format: ARRAY: eight-character text field (DATASET, RESOURCE, SYSTEM or USERID).

MC.USERID	**Userid making the maintenance request event**

Description: Identifies the Userid who applied the maintenance to the target resource. *This dataname represents the same field as "MC.CHANGER".*

Format: ARRAY: Eight character Userid text.

This page intentionally left blank

Chapter 17: OWNER Segment Datanames

The OWNER segment stores information about E-SRF owners. The Resource Grouping Facility provides a means for "grouping" users, sources, and resources into logical groups and assigning ownership to these groups. OWNERS are defined in this segment.

NOTE: You may substitute "OWNER" for "OA" if desired.

Related to all OWNER Objects

OA.ID	Owner identification

Description: Contains the owner identification name. This is what the owner is called and referred to throughout E-SRF.

Format: KEY: Appears as an eight-character text field.

Related to OWNER Header Object

The HEADER object type consists of single data items that define a particular owner.

The key for this object is: OA.ID

OA.ADDRESS1	Owner address line 1

Description: Line 1 of the owner's address text.

Format: SINGLE: Appears as a twenty-four-character text field.

OA.ADDRESS2	Owner address line 2

Description: Line 2 of the owner's address text.

Format: SINGLE: Appears as a twenty-four-character text field.

OA.ADDRESS3	Owner address line 3

Description: Line 3 of the owner's address text.

Format: SINGLE: Appears as a twenty-four-character text field.

OA.ADDRESS4	Owner address line 4

Description: Line 4 of the owner's address text.

Format: SINGLE: Appears as a twenty-four-character text field.

OA.ADDRESS5	Owner address line 5

Description: Line 5 of the owner's address text.

Format: SINGLE: Appears as a twenty-four-character text field.

OA.DATE — Owner specific date format

Description: Single character date formatting specification unique to the specific owner.

This specification, if present, will override any previous DATEFORMAT specification that may be in effect for a particular report execution while processing under Automatic Report Distribution.

For more information on the characteristics of the specifications of this field, please refer to the DATEFORMAT information contained on either the SET or RUN command in the *E-SRF Event System Command Guide*.

Format: SINGLE:

blank	default to system specification.
I	International date format
U	USA date format

OA.JESCLASS — MVS JES SYSOUT class

Description: Specifies the JES SYSOUT CLASS=x that will be used in the event the owner's DD was not supplied in the execution's JCL by E-SRF's dynamic allocation routines.

Format: SINGLE: Single character identifying the JES SYSOUT class for the owner.

OA.JESDEST — MVS JES SYSOUT destination

Description: Specifies the JES SYSOUT DEST=*destination* that will be used in the event the owner's DD was not supplied in the execution's JCL. If this field is blank, no destination is used in the dynamic allocation.

Format: SINGLE: One to forty-eight character string identifying the JES destination for the owner.

OA.JESWTR — MVS JES External Writer name

Description: Specifies the JES SYSOUT *External Writer* that will be used in the event the owner's DD was not supplied in the execution's JCL. If this field is blank, no Eternal Writer name will be used in the allocation.

Format: SINGLE: One to eight character string identifying the JES External Writer name for the owner.

OA.LAF — Look and feel override

Description: Single character optional report "Look and Feel" specification unique to the specific owner.

This specification, if present, will override any previous LAF specification that may be in effect for a particular report execution while processing under Automatic Report Distribution.

For more information on the characteristics of the specifications of this field, please refer to the LAF information contained on either the SET or RUN command in the *E-SRF Event System Command Guide*.

Format: SINGLE: One character specification. Blank indicates no owner level specification.

OA.NAME — Owner name

Description: Contains user specified text to identify the name of a particular owner.

Format: SINGLE: Appears as a twenty-four-character text field.

OA.PHONE — Owner phone number

Description: Contains telephone number of owner.

Format: SINGLE: Appears as a twelve-character text in any format.

OA.ROUTING — Owner report routing information

Description: Contains textual information that describes how to route the printed report output.

Format: SINGLE: Appears as a twenty-four-character text in any format.

OA.TIME — Owner specific time format

Description: Single character time of day formatting specification unique to the specific owner.

This specification, if present, will override any previous TIMEFORMAT specification that may be in effect for a particular report execution while processing under Automatic Report Distribution.

For more information on the characteristics of the specifications of this field, please refer to the TIMEFORMAT information contained on either the SET or RUN command in the *E-SRF Event System Command Guide*.

Format: SINGLE:

blank	default to system specification.
M	Military (24 hour) time representation
S	Standard 12 hour clock representation

OA.USERDATA — Installation required userdata

Description: Contains userdata that relates to a particular installation. Any type of text may be placed here.

Format: SINGLE: Appears as one to sixty-four character text field.

This page intentionally left blank

Chapter 18: RESOURCE Segment Datanames

The RESOURCE segment stores information about resources (*things*) secured by the Resident Security System (RSS). The RSS logs information based on access control parameters specified for resources. All logged access controls for resources are represented in this segment. This segment contains multiple object types.

Related to all RESOURCE Objects

Common Datanames: CLASS, RESOURCE, VOLUME and RESTOKEN

Related to RESOURCE Chronological Object

The CHRONOLOGICAL object type consists of array elements describing particular logged access event activity. Logged access includes ALLOW but LOG and VIOLATION information.

There are as many array elements as there are events multiplied by the number of retention days specified for this object type.

The key for this object is: CLASS, RESOURCE and VOLUME

RC.ACCESS	Type of resource access requested

Description: Indicates the "normalized" type of access requested. Please refer to Appendix B for a list of all ACCESS specifications.

Format: ARRAY: Appears as an eight-character keyword identifying the type of access requested.

RC.ACTION	Type of action taken based on RSS security controls in effect at access time

Description: Indicates the "normalized" action taken with respect to the requester's ability to access the requested resource in the manner dictated by the security request. Please refer to Appendix C for a list of all ACTION specifications.

Format: ARRAY: Appears as a nine-character keyword identifying the type of access requested.

RC.DATE	Date the event occurred

Description: The date an event took place according to the Resident Security System.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RC.DOMAIN	Resource event hosting domain (sysid)

Description: Domain (sysid) where the resource event took place on.

Format: ARRAY: Stored as character data.

RC.ETF — EKC's ETF/* Status Information

Description: Indicates the actual ETF/* involvement status for this request (if any). All Status codes are listed in Appendix E of this guide.

Format: ARRAY: Stored as character data.

RC.ETF-ACT — EKC's ETF/* was active in this event

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility was involved in this event.

Format: ARRAY: Stored as a flag. Refer to "RC.ETF" for additional information.

RC.ETF-TEST — EKC's ETF/* performed a TEST access control function

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility logged this event as a result of a test access control request. If the flag is on, the logged access was determined using "test" access controls. The result of the test was NOT used to determine access to the resource.

Format: ARRAY: Stored as a flag. Refer to "RC.ETF" for additional information.

RC.IMAGE — Resource event Security Image ID

Description: Security image relative to domain where resource event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

RC.JOBNAME — Current executing job according to the Operating System

Description: The jobname as the operating system knows it. In MVS, this information comes from the name field of the "JOB" card.

Format: ARRAY: Eight character jobname.

RC.KEY — Resource access RSS KEY information

Description: This information represents what the RSS used to determine access to the resource. This field is a shortened (eight character) version of RC.PROFILE. Each RSS uses a different approach and the data will represent this.

If the RSS is ACF2, and the resource was a DATASET, the KEY will be the RULEKEY used to determine the actual access. The RULEKEY is the actual eight-character VSAM record key for the RULESET. For other RESOURCE requests, the first eight characters of the resource key will be present.

If the RSS is RACF, this field represents the first eight characters of the RACF profile used in determining the access.

Please refer to UC.PROFILE, as this may be a better choice depending on your needs.

Format: ARRAY: Eight character RSS access control key information. Possible right truncated if the access key was longer than eight characters.

RC.KEYTOKEN — Resource access RSS KEY information TOKEN

Description: Provides actual token that represents the RC.KEY data.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: ARRAY: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this dataname relates to the tokenized RSS key (RC.KEY) data.

RC.LOGONID — Current user's LOGONID (USERID) accountable for the event

Description: The user's LOGONID (USERID) associated with this request, the one signed on (or attempting to signon) to the environment making the request.

Format: ARRAY: Eight character USERID.

RC.PRI-UID — EKC's ETF/* used the PRIMARY UID to determine access

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility used the user's PRIMARY UID to determine access.

Format: ARRAY: Stored as a flag. Refer to "RC.ETF" for additional information.

RC.PROFILE — Resource access profile name information

Description: This information represents what the RSS used to determine access to the resource. This field represents the full forty-four-character profile name.

If the RSS is ACF2, and the resource was a DATASET, the PROFILE would be the eight character RULEKEY used to determine the actual access. For other resources, this field contains the resource rule RULEKEY n the following format: R(*ttt*)*resource_rule_key*.

If the RSS is RACF, this field represents the RACF profile determining the access.

Format: ARRAY: Forty-four character "shrinkable" RSS profile name information. Possible right truncated if the access key was longer than forty-four characters, or if this field was shrunk to make it fit on a report.

RC.PROGRAM — Current executing program name according to the Operating System

Description: The program name as the operating system knows it. In MVS, this means the "Load Module" name of the problem program that is hosting the request.

Format: ARRAY: eight-character load module program name.

RC.PROTOKEN — Profile Name TOKEN

Description: Provides actual token that represents the RC.PROFILE data.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: ARRAY: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this dataname relates to the tokenized RSS profile (RC.PROFILE) data.

RC.REASON — Actual reason action taken by the RSS based on the type of request made

Description: Indicates the actual reason action was taken. All reasons are unique, knowledge of the actual request is not required. All action reason codes are listed in Appendix D of this guide.

Format: ARRAY: Appears as a twenty-character keyword identifying the RSS dependent reason for the action taken.

RC.RECTYPE — E-SRF event type normalized classification

Description: E-SRF normalizes the event journalize reason into three classifications which are common to all Resident Security Systems.

ENTRY: Event logged as a System Entry (signon/signoff) request
LOG: Access granted and logged by normal validation procedures.
SPECIAL: Access granted and logged due to special conditioning
VIO: Access was denied

Consult the User's Guide for information on RECTYPE and other related normalization considerations.

Format: ARRAY: RECTYPE value assigned to this event by E-SRF.

RC.RESCOMMENT — Resource Group Comment

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. (*see*: RC.RESGROUP). It will only be present if the grouping rule associated with this resource contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing actual names to resources such as CICS and IMS transaction Ids.

Format: ARRAY: Up to forty-eight characters of comment data.

RC.RESGROUP — Current E-SRF resource owning group ID

Description: The Resource Group name which was assigned to this resource. This group name is dynamically determined during processing and it is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

RC.RESOWNER — Current E-SRF resource owner ID

Description: The Resource Owner that owns the group which was assigned to this resource for this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: ARRAY: eight-character owner ID.

RC.RSS — Resident Security System that posted this event

Description: The normalized name of the Resident Security System (RSS) which was responsible for posting this event. . All RSS codes are listed in Appendix A of this guide.

Format: ARRAY: eight-character name of the Resident Security System and its primary component.

RC.SEC-UID	EKC's ETF/* used a SECONDARY UID to determine access

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility used one of the user's SECONDARY UID specifications to determine access.

Format: ARRAY: Stored as a flag. Refer to "RC.ETF" for additional information.

RC.SOURCE	Current location identification of request

Description: The source represents the name of the point to entry for the request. It may be a JES RJE *nodename*, or a particular VTAM terminal *nodename*. In ACF2, it could be a source group name.

Format: ARRAY: eight-character source name.

RC.SRCCOMMENT	SOURCE Comment

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. (*see*: RC.SRCGROUP). It will only be present if the grouping rule associated with the source involved in the event contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing actual names to sources that are related to events.

Format: ARRAY: Up to forty-eight characters of comment data.

RC.SRCGROUP	Current E-SRF "source" owning group name

Description: The EKC Grouping Facility Group name which was assigned to the particular "source" name. This group name is dynamically determined during processing and it is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

RC.SRCOWNER	Current E-SRF "source" owner ID

Description: The E-SRF Owner name that owns the group that was assigned to the source involved in this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: ARRAY: eight-character owner ID.

RC.SYSTEM	Resource event hosting sysid (Domain ID)

Description: Sysid (Domain ID) where the maintenance event took place on.

Format: ARRAY: Stored as character data.

RC.TIME	Time the event occurred

Description: The time an event took place according to the Resident Security System.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RC.UID — Universal User Identification

Description: The user's E-SRF Universal User Identification (UID) associated with this request. This information is RSS dependent, which relate to the specific event being journalized.

In ACF2, it is the ACF2 UID string data.

In RACF, it is a composite of the OWNER, DEFAULT GROUP (if available) and the USERID.

Format: ARRAY: Twenty-four character data.

RC.UIDTOKEN — UID TOKEN

Description: Provides actual token that represents the RC.UID data.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: ARRAY: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this dataname relates to the tokenized E-SRF UID (RC.UID data.

RC.USERID — Current user's USERID (LOGONID) accountable for the event

Description: The user's USERID (LOGONID) associated with this request, the one signed on (or attempting to signon) to the environment making the request.

Format: ARRAY: Eight character USERID.

RC.USRCOMMENT — USERID Comment

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. (*see*: RC.USRGROUP). It will only be present if the grouping rule associated with the userid involved in the event contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing additional information relating to the userid involved in the event being journalized.

Format: ARRAY: Up to forty-eight characters of comment data.

RC.USRGROUP — Current E-SRF "userid" owning group name

Description: The Resource Group name which was assigned to the particular userid named in this event. This group name is dynamically determined during processing and is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

RC.USROWNER — Current E-SRF "user" owner ID

Description: The Resource Owner that owns the group that was assigned to the user involved in this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: ARRAY Eight character owner ID.

Related to RESOURCE Maintenance Object

The Resource Maintenance object type consists of array elements summarizing all maintenance events that occurred for a particular "resource" definition.

This object consists of resource definitions as they were made to a specific RSS,

The information contained is the changer's userid, the date, time, and type of change made.

The key for this object is: CLASS, RESOURCE and VOLUME

RM.CHANGER	**Changer's userid**

Description: This field contains the user who made the change to the resource being reported on.

Format: ARRAY: Appears as an eight character standard userid.

RM.DATANAME	**The name of the data item changed**

Description: This field indicates the name of the item that was changed during resource maintenance

If the contents of this item is +ADMIN, this particular event represents a summary of the requested transaction. The detail items that were actually changed on the security database will follow. If all items presented in this transaction caused no change the security database, than this will be the only event that is reported.

Format: ARRAY: eight-character maintenance request data item.

RM.DATE	**Date the resource maintenance event occurred**

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RM.DOMAIN	**Resource maintenance event hosting Domain (sysid)**

Description: Domain (sysid) where the resource maintenance event took place on.

Format: ARRAY: Stored as character data.

RM.IMAGE	**Resource maintenance event Security Image ID**

Description: Security image relative to domain where resource maintenance event took place. The security databases for the named image was the target of the maintenance.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

RM.LOGONID	**Userid making the maintenance request event**

Description: Identifies the Userid who applied the maintenance to the target resource. *This dataname represents the same field as "RM.CHANGER".*

Format: ARRAY: Eight character Userid text.

RM.NEWDATA **User maintenance new data**

Description: The information shown here is the actual new data related to the "dataname" that was applied to the security database for the named resource. Please note that this data is not always available.

Please note, if RM.DATANAME is +ADMIN, the NEWDATA field is used to indicate the type of request performed, such as INSERT, ALTER or DELETE.

Format: ARRAY: Stored as a twenty-four-character left justified data item.

RM.OLDDATA **User maintenance old data**

Description: The information shown here is the actual old data related to the "dataname" that was applied to the security database for the named resource. Please note that this data is not always available.

Please note, if RM.DATANAME is +ADMIN, the OLDDATA field is used to maintain summary statistics on what was actually altered against what was requested and is represented by "CHANGED: *nnn*, NOCHG: *nnn*". The CHANGED number represents how many individual data items listed in the transaction were actually different and changed on the security database. The NOCHG number represents how many individual data items were requested to be changed, but were identical and therefore were not changed. The sum of the two numbers show how many items were presented for change.

Format: ARRAY: Stored as a twenty-four-character left justified data item.

RM.REQUEST **Resource maintenance request type**

Description: The type of maintenance request made for this particular event.

Format: ARRAY: eight-character text field containing request type as specified for the Resident Security System. Please refer to *Appendix G* of this publication for information related to the particular Resident Security System responsible for the maintenance.

RM.RSS **Resident Security System that posted this event**

Description: The normalized name of the Resident Security System (RSS) which was responsible for posting this event. . All RSS codes are listed in Appendix A of this guide.

Format: ARRAY: eight-character name of the Resident Security System and its primary component.

RM.SOURCE **Source maintenance event was performed from**

Description: Identifies the "source" (place) the maintenance was performed on.

Format: **ARRAY: eight-character text field (DELETE, INSERT or REPLRM.SYSTEM Resource maintenance event hosting sysid (Domain ID)**

Description: Sysid (Domain ID) where the resource maintenance event took place on.

Format: ARRAY: Stored as character data.

RM.TIME	**Time the resource maintenance event occurred**

Description: The time an event took place according to the Resident Security System.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RM.TYPE	**Type of maintenance**

Description: Type of Resident Security System data that the maintenance was applied to.

Format: ARRAY: eight-character text field (DATASET, RESOURCE, SYSTEM or USERID).

RM.USERID	**Userid making the maintenance request event**

Description: Identifies the Userid who applied the maintenance to the target resource. *This dataname represents the same field as "RM.CHANGER".*

Format: ARRAY: Eight character Userid text.

Related to RESOURCE Recap Summary Object

The Recap Summary object type consists of array elements summarizing all events that took place *for a particular resource* in a single array element. For example, if a particular resource had two hundred logged accesses and fifteen prevented accesses (violations), a single array element will be present summarizing this information.

The key for this object is: CLASS, RESOURCE and VOLUME

RR.DATE	Date the events occurred

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RR.DOMAIN	Resource events hosting Domain (sysid)

Description: Domain (sysid) where the resource event took place on.

Format: ARRAY: Stored as character data.

RR.EXIT	Number of installation user exit access allows

Description: The installation exit granted access to particular resources and logged this situation. This field represents how many times this occurred in a given day.

Format: ARRAY: Appears as a standard numeric data field.

RR.GROUP	Current E-SRF resource owning group name

Description: The Resource Group name which was assigned to this object's resource. This group name is dynamically determined during processing, and is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: sixteen-character group name.

RR.IMAGE	Resource events Security Image ID

Description: Security image relative to domain where resource maintenance event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

RR.LOGFIRST	Time of first logged event

Description: The time that the first *logged* event summarized in a particular array element took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RR.LOGLAST — Time of last logged event

Description: The time which the last *logged* event summarized in a particular array element took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RR.LOGS — Number of logged events

Description: The Resident Security System allowed an access to particular resources and determined that the access should be logged. This count represents the number of access loggings that occurred for a given day.

This is the total of ALL logged events.

Format: ARRAY: Appears as a standard numeric data field.

RR.NON-CNCL — Number of NON-CNCL access allows

Description: The Resident Security System allowed an access to particular resources because the requester had the non-cancelable attribute. This count represents the number of non-cancel access loggings that occurred for a given day.

Format: ARRAY: Appears as a standard numeric data field.

RR.OWNED — Number of owned access allows

Description: The Resident Security System allowed an access to particular resources because the requester owned it. This count represents the number of owned access loggings that occurred for a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

RR.OWNER — Current E-SRF resource owner ID

Description: The Resource Owner that owns the group which was assigned to this resource for this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: ARRAY: eight-character owner ID.

RR.READALL — Number of forced "read-only" access allows

Description: The Resident Security System allowed a "read-only" access to particular resources because the requester had the "readall" attribute. This count represents the number of forced read only loggings that occurred for a given day.

Format: ARRAY: Appears as a standard numeric data field.

RR.RULE-LOG **Number of allow accesses that were logged**

Description: The Resident Security System allowed an access based on its security "runes" in place for particular resources. The particular access allow was made with the condition of logging the fact that it was made for a particular requester. This count represents the number of access allow loggings that occurred for a given day.

Format: ARRAY: Appears as a standard numeric data field.

RR.SECURITY **Number of security officer forced access allows**

Description: The Resident Security System allowed an access to particular resources because the requester was considered a security officer for a particular resource. This count represents the number of accesses granted for this reason.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

RR.SPECIAL **Number of forced allows due to "special privileges" RECTYPE classification**

Description: This represents the number of allowed accesses due to reasons that would cause E-SRF to assign a RECTYPE of SPECIAL to the event. Please refer to the E-SRF User's Guide for additional information on what type of attributes dictate this type of classification.

Format: ARRAY: Appears as a standard numeric data field.

RR.SYSTEM **Resource maintenance event hosting sysid (Domain ID)**

Description: Sysid (Domain ID) where the resource events took place on.

Format: ARRAY: Stored as character data.

RR.VIOFIRST **Time of first logged violation event**

Description: The time that the first *logged violation* event summarized in a particular array element took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RR.VIOLAST **Time of last logged violation event**

Description: The time that the last *logged violation* event summarized in a particular array element took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RR.VIOS	**Number of logged violation events**

Description: The Resident Security System denied access to particular resources, considered them to be violations, and determined that the access attempt should be logged. This count represents the number of access attempt violation loggings that occurred for a given day.

All violations are grouped together and represented by this field. There is no individual breakout of different types of violations in the current release of E-SRF.

Format: ARRAY: Appears as a standard numeric data field.

Related to RESOURCE Statistical Summary Object

The Statistical Summary object type consists of a single array element summarizing all journalized events that involved resource access for a particular day. One of these array elements will be created for each USERID who accesses a specific resource on a given day.

For example, if a particular resource had a single user who had two hundred logged accesses and a single prevented access (violation), a single array element will be created indicating the summarization of these events.

The key for this object is: CLASS, RESOURCE and VOLUME

RS.ALLOWS	**Number of summarized events**

Description: Sum of all "RECTYPE" LOG, and SPECIAL events that occurred.

Format: ARRAY: This field is dynamically calculated on demand based on the sum of RS.LOGS and RS.SPECIAL. Appears as a standard numeric data field.

RS.DATE	**Date the event occurred**

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RS.DOMAIN	**Resource event hosting Domain (sysid)**

Description: Domain (sysid) where the resource event took place on.

Format: ARRAY: Stored as character data.

RS.EVENTS	**Total number of Journalized Security Events**

Description: Sum of all "RECTYPE" LOG, SPECIAL and VIOLATION events that occurred. This count is the total number of journalized event for this user for the current resource for the current day.

Format: ARRAY: This field is dynamically calculated on demand based on the sum of RS.LOGS, RS.SPECIAL and RS.VIOS

RS.EXIT	**Number of installation user exit access allows**

Description: The installation exit granted access to the particular resource and journalized it. This field represents how many times this occurred in a given day.

Format: ARRAY: Appears as a standard numeric data field.

RS.FIRST	**Time of first event**

Description: The time that the first event summarized in a particular array element took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RS.IMAGE	Security Image ID

Description: Security image relative to domain where the events took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

RS.LAST	Time of last event

Description: The time that the last event for a particular user summarized in a particular array element took place on a given day.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

RS.LOGONID	Userid for which the event(s) are associated to

Description: The summarized array element reports occurrences of particular events for a particular userid, which is identified by this field. *This is the same field as RS.USERID.*

Format: ARRAY: Appears as an eight character standard userid.

RS.LOGS	Number of "normal" access loggings

Description: Number of times access was journalized by the RSS for the particular user on a day. These loggings were classified as RECTYPE(LOG) by the RSS update program.

Format: ARRAY: Appears as a standard numeric data field.

RS.NON-CNCL	Number of NON-CNCL access allows

Description: The Resident Security System allowed an access to particular resources because the requester has an attribute that forces allow access despite the security access controls in place for the resource. Access granted to a resource via the ACF2 NON-CNCL or RACF OPERATIONS attribute are examples of this type of logging. This count represents the number of such loggings that occurred for a particular userid on a given day.

Format: ARRAY: Appears as a standard numeric data field.

RS.OWNED	Number of owned access allows

Description: The Resident Security System allowed access and journalized the access to a particular resource because the requester owns it. This count represents the number of such loggings that occurred for a particular user on a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

RS.READALL	Number of forced "read-only" access allows

Description: The Resident Security System allowed "read-only" access to particular resources because the requester had an attribute that permitted that type of access overriding the normal access controls governing the ability to obtain the resource for read only access. This count represents the number of such loggings that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

RS.RESGROUP **Group which this resource is associated with.**

Description: The Resource Group name assigned to this resource. This group name is dynamically determined during processing, (it is not stored on the E-SRF Masterfile). The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: sixteen-character group name.

RS.RESOWNER **Owner ID of this resource**

Description: The Resource Owner that owns the group, which was assigned to this resource, for this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile, using the dynamically associated group ID.

Format: ARRAY: eight-character owner ID.

RS.RULE-LOG **Number of journalized resource access logs granted via resource definitions.**

Description: The Resident Security System allowed an access based on its security resource access definitions. The particular granting definition journalized the event. This count represents the number of such loggings that occurred for a given day.

Format: ARRAY: Appears as a standard numeric data field.

RS.SECURITY **Number of security officer forced access allows**

Description: The Resident Security System allowed access to particular resources because the requester was considered a security officer. This count represents the number of such loggings that occurred for a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

RS.SPECIAL **Number of "special" access loggings**

Description: Number of times access was journalized by the RSS for the particular user on a day. These loggings were classified as RECTYPE(SPECIAL) by the RSS update program.

Format: ARRAY: Appears as a standard numeric data field.

RS.SYSTEM **Sysid (Domain ID) hosting representing these events**

Description: Sysid (Domain ID) where the maintenance event took place on.

Format: ARRAY: Stored as character data.

RS.USERID **Userid for which the event(s) are associated to**

Description: The summarized array element reports occurrences of particular events for a particular userid, which is identified by this field. *This is the same field as RS.LOGONID.*

Format: ARRAY: Appears as an eight character standard userid.

RS.USRGROUP	Group that the user is related to

Description: The Resource Group name assigned to the user named in this array. This group name is dynamically determined during processing, (it is not stored on the E-SRF Masterfile). The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

RS.USROWNER	Current user's owner ID

Description: The Resource Owner that owns the group, that was assigned to the user named in the array. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile, using the dynamically associated user's group ID.

Format: *ARRAY Eight character owner ID.*

RS.VIOS	Number of logged violation events

Description: The Resident Security System denied access to a particular resource. This count represents the number of violations that occurred for a given day.

Format: ARRAY: Appears as a standard numeric data field.

This page intentionally left blank

Chapter 19: SOURCE Segment Datanames

The SOURCE segment stores information about sources *(places)* where events occurred. This segment contains a single object type.

Related to all SOURCE Objects

SOURCE	Current location identification of request

Description: The source represents the name of the point to entry of the request. It may be a JES RJE *nodename*, or a particular VTAM terminal *nodename*. In the case of ACF2, it could be a source group name.

Format: ARRAY: eight-character source name.

Related to SOURCE Recap Object

The RECAP object type consists of array elements describing particular access logged event activity. Logged access includes ALLOW but LOG and VIOLATION information. There are as many array elements as there were events for the number of retention days specified for this object type.

The key for this object is: SOURCE

SR.DATE	Security activity date

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

SR.DOMAIN	Domain (sysid) hosting this source and its events

Description: Domain (sysid) where this source exists and where all events reported on this object occurred on.

Format: ARRAY: Stored as character data.

SR.DS-LOG	Number of dataset access loggings

Description: Number of times dataset access was granted by the Resident Security System on the particular source with the condition of logging the event on a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.DS-VIO	Number of dataset access violation loggings

Description: Number of times dataset access was denied by the Resident Security System on the particular source and the event was logged during a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.EXIT	Number of installation user exit access allows

Description: The installation exit granted access on a particular source to particular resources and logged this situation. This field represents the times this occurred in a day.

Format: ARRAY: Appears as a standard numeric data field.

SR.FIRST	Time of first event

Description: The time that the first event on a particular source took place on a given day.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

SR.IMAGE	Security Image ID related to Domain hosting this source and its events

Description: Security image relative to domain where this source exists and where all events reported on this object occurred on.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

SR.LAST	Time of last event

Description: The time that the last event on a particular source took place on a given day.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

SR.NON-CNCL	Number of NON-CNCL access allows

Description: The Resident Security System allowed an access to particular resources because the requester had the NON-CANCEL attribute. This count represents the number of non-cancel access loggings that occurred for a given day on a particular source.

Format: ARRAY: Appears as a standard numeric data field.

SR.NOTAVAIL	Number of times signon was denied because userid was unavailable

Description: The Resident Security System disallows a user from signing on because the userid is not available. The reasons vary with each RSS. This field represents the number of such situations that have occurred on a particular source on a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.OWNED	Number of owned access allows

Description: The Resident Security System allowed access to particular resources because the requester owned it. This count represents the number of owned access loggings that occurred on a particular source for a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

SR.PGM-VIO	Number of violations due to unauthorized program usage

Description: The Resident Security System disallowed access to particular resources because the requester accessed them from an unauthorized program. This count represents the number of such violations that occurred on a particular source in a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

SR.PSWD-VIO	Number of signon denials due to password violation

Description: The Resident Security System disallowed signon by users due to invalid password attempts. This count represents the number of such violations that occurred on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.READALL	Number of forced "read-only" access allows

Description: The Resident Security System allowed a "read-only" access to particular resources because the requester had the READALL attribute. This count represents the number of such loggings that occurred on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.RES-LOG	Number of non-dataset resource access loggings

Description: Number of times non-dataset resource access was granted by the Resident Security System on the particular source and logged the event in a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.RES-VIO	Number of non-dataset access violation loggings

Description: Number of times non-dataset access was denied by the Resident Security System on the particular source and the event was logged in a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.RULE-LOG	Number of logged accesses granted by access control rules

Description: Number of times access was granted by the Resident Security System based on access control rules with the condition that access be logged on the particular source on a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.RULE-VIO	Number of access denials based on access control rules

Description: Number of times access was denied by the Resident Security System based on access control rules on the particular source and logged the event on a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.SECURITY **Number of security officer forced access allows**

Description: The Resident Security System allowed access to particular resources because the requester was considered a security officer for a particular resource. This count represents the number of such accesses on the particular source for a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

SR.SHFT-VIO **Number of access denials based on access control "shift" rules**

Description: Number of times access was denied by the Resident Security System based on "shift" (time of day access) rules on the particular source on a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.SIGN-SUS **Number of signon denials due to suspended userids**

Description: The Resident Security System disallowed signon by users because their userids were suspended for one reason or another. This count represents the number of this type of violation that occurred on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.SIGN-UKN **Number of unknown signon requests**

Description: The Resident Security System disallowed signon for unknown reasons. This count represents the number of such situations that occurred on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.SIGN-VIO **Number of signon denials for all reasons**

Description: The Resident Security System disallowed signon for users. This count represents the number of violations that occurred on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.SIGNON **Number of signon requests performed**

Description: The Resident Security System performed signon operations for users attempting to access the system. This count represents the number of signons performed on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.SRCCOMMENT **SOURCE Comment**

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. (*see*: SR.SRCGROUP). It will only be present if the grouping rule associated with the source involved in the event contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing actual names to sources that are related to events.

Format: ARRAY: Up to forty-eight characters of comment data.

SR.SRCE-VIO **Number of signon denials based on requester's source**

Description: Number of times signon access was denied by the Resident Security System based on source controls on the particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.SRCGROUP **Current E-SRF "source" owning group name**

Description: The EKC Grouping Facility Group name which was assigned to the particular "source" name. This group name is dynamically determined during processing and it is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

SR.SRCOWNER **Current E-SRF "source" owner ID**

Description: The E-SRF Owner name that owns the group that was assigned to the source involved in this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: ARRAY: eight-character owner ID.

SR.SUSPEND **Number of violations due to user being suspended**

Description: The Resident Security System disallowed access to particular resources because the user is "suspended". This count represents the number of such violations that occurred on a particular source in a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

SR.SYSTEM **Sysid (Domain ID) hosting this source and its events**

Description: Domain (sysid) where this source exists and where all events reported on this object occurred on.

Format: ARRAY: Stored as character data.

SR.UKN-USER **Number of unknown user signon attempts**

Description: The Resident Security System disallowed signon because the userid specified was not known. This is the only place this is tracked in E-SRF. This count represents the number of such situations that occurred on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.UNKNOWN **Number of unknown non-signon security requests**

Description: This is a "catch-all" used to track a non-signon security event. Events include unknown events, information altered by user exits, and new events implemented after this release of E-SRF was made Generally Available. This count represents the number of such situations that occurred on a particular source for a given day.

Format: ARRAY: Appears as a standard numeric data field.

SR.USER1	**First of the five last users to interact with this source**

Description: The last five users who caused a security event to take place on a given day for a particular source is tracked. This is the most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

SR.USER2	**Second of the five last users to interact with this source**

Description: The last five users who caused a security event to take place on a given day for a particular source is tracked. This is the second most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

SR.USER3	**Third of the five last users to interact with this source**

Description: The last five users who caused a security event to take place on a given day for a particular source is tracked. This is the third most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

SR.USER4	**Fourth of the five last users to interact with this source**

Description: The last five users who caused a security event to take place on a given day for a particular source is tracked. This is the fourth most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

SR.USER5	**Fifth of the five last users to interact with this source**

Description: The last five users who caused a security event to take place on a given day for a particular source is tracked. This is the fifth most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

Related to SOURCE Userid Chronological Object

The Userid Chronological object type consists of array elements describing particular signon request that failed because the userid character string did not match any userids contained on the security system's userid database.

When a user enters an invalid userid, the Resident Security System rejects the attempt and follows up with a journal indicating the actual character string presented as well as which "source" the incident occurred on.

E-SRF does not maintain Invalid Userids on its Masterfile's user Segment. Doing so would clutter up the User Segment with what would amount to typographical errors and would have little or no meaning.

The Update Function publishes a summary of each invalid userid's character pattern and how many times the pattern was presented relative to the set of update journals being processed.

To make post update reporting of invalid userids possible, these events are also recorded within the SU (Source Userid) object (*if this object is being retained*). The event is recorded treating the character string as a "resource". By activating the SU object, and preparing ESRFLIST reports, you can produce reports showing some (or all) invalid userid signon attempts.

The key for this object is: SOURCE

SU.DATE	**Security activity date**

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

SU.DOMAIN	**Domain (sysid) hosting this source and its events**

Description: Domain (sysid) where this source exists and where all events reported on this object occurred on.

Format: ARRAY: Stored as character data.

SU.IMAGE	**Security Image ID related to Domain hosting this source and its events**

Description: Security image relative to domain where this source exists and where all events reported on this object occurred on.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

SU.LOGONID	**Logon (userid) character string as presented for signon**

Description: Character text that was intended to be the Logonid to be signed on to the source with. This character string contains an invalid Logonid and is the exact same as using the SU.USERID field.

Format: ARRAY: Stored as character data.

SU.SYSTEM	**Sysid (Domain ID) hosting this source and its events**

Description: Domain (sysid) where this source exists and where all events reported on this object occurred on.

Format: ARRAY: Stored as character data.

SU.TIME	**Time the event occurred**

Description: The time an event took place according to the Resident Security System.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

SU.USERID	**Userid character string as presented for signon**

Description: Character text that was intended to be the userid to be signed on to the source with. This character string contains an invalid Userid and is the exact same as using the SU.LOGONID field.

Format: ARRAY: Stored as character data.

Chapter 20: USER Segment Datanames

The USER segment stores information about users *(people)* who caused events to occur. This segment contains multiple object types.

Related to all USER Objects

IMAGE	Security image

Description: Provides the Security Image ID the current user. The Userid *and* Image ID comprise the key that identifies a user to the user Segment. The objects under this key contain all data for the domains associated to the particular image.

If you have more than one image, and the users that are in multiple images have the same userids, the activity for these users will be split between the images in relation to the activity occurring across the domains assigned to the images.

For more information, please refer to the ASSIGN command in the *Event System Command Reference*.

Format: KEY: External reference to eight character Security Image ID.

LOGONID	User's Logonid (userid)

Description: The identification assigned the requester (user) which makes the requester known to the Resident Security System (RSS). This is the same as USER or USERID

Format: KEY: Appears as an eight character standard userid.

USER	User's userid

Description: The identification assigned the requester (user) which makes the requester known to the Resident Security System (RSS). This is the same as USERID or LOGONID.

Format: KEY: Appears as an eight character standard userid.

USERID	User's userid

Description: The identification assigned the requester (user) which makes the requester known to the Resident Security System (RSS). This is the same as USER or LOGONID.

Format: KEY: Appears as an eight character standard userid.

Related to USER Header Object

The HEADER object type consists of information describing the user that includes information from the Resident Security System. Most of the fields relate directly to the RSS.

In ACF2, the ACF2 Field Definition Record (FDR) is included in the dictionary. A copy of every current ACF2 Logonid Record is stored in this object. When the RSS changes the data, a journal record is produced which E-SRF uses as a signal to dynamically retrieve a new copy of the complete Logonid Record that has changed.

In RACF, the fields were named by E-SRF development. These fields were obtained by using the downloaded RACF extracted user information. As of release 1.4, some of this information may NOT be dynamically upgraded on the Masterfile. This means the data is as recent as the last time the image was SYNCHRONIZED. Research is in progress to eliminate this potential restriction, and expand the User Header upgrade capability for RACF. Until that time, it is recommended that the image be synchronized when these fields are used, if they are not part of the set of fields that are dynamically updated. Please refer to *Appendix H* for more information.

The key for this object is: USERID and IMAGE

UA.COMMENT — Grouping Comment Text

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname (*see*: UA.GROUP). It will only be present if the grouping rule associated with the userid contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing additional information contained in the grouping rule.

Format: Up to forty-eight characters of comment data.

UA.DELETED — User RSS delete date

Description: The RSS deleted this user from its database. This user no longer exists on the RSS. E-SRF will mark this user (within the current IMAGE ID) as "deleted" effective the date the RSS issued the delete request.

E-SRF will not actually remove the user from its Masterfile because other objects, such as resource violations in the Resource Segment, refer to the user causing the violation.

This field may be tested to keep deleted users from appearing on various reports.

See the *E-SRF User Guide* for information on how users actually get purged from the Masterfile.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UA.GROUP — Group Identification

Description: Provides the Resource Group name for the current userid. This group name is dynamically determined during processing. The data presented is the result of interpreting the E-SRF grouping rules in the Resource Grouping Facility. For more information on how E-SRF uses groups, see the section*, "Grouping and Ownership*", in the *User Guide*.

Format: SINGLE: External reference to sixteen-character group name associated to the current userid. Please note that the userid is a composite of the actual userid and the IMAGE the user belongs to.

UA.NAME — User's spelled out name

Description: The user's name comes from the creation of the userid information from the Resident Security System data. It is stored independently of the name information in the RSS data areas carried on this object.

Format: SINGLE: Appears as a twenty-character name text field.

UA.OWNER — Current user owner ID

Description: The Resource Owner that owns the group which was assigned to the user. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: SINGLE: eight-character owner ID.

UA.RSS — Resident Security System this user belongs to

Description: The user definition is related to a particular IMAGE. The IMAGE is associated to a particular Resident Security System (RSS). This data item identifies the normalized name of the Resident Security System (RSS) that the user has been associated with. All RSS codes are listed in Appendix A of this guide.

Format: ARRAY: eight-character name of the Resident Security System and its primary component.

UA.UID — User's Universal Identification

Description: The user's Universal Identification (UID) comes from the creation of the userid information from the Resident Security System data. It is stored independently of the data used to compile the field.

E-SRF's UA.UID information is a copy of the user's ACF2 UID string.

Format: SINGLE: Appears as a twenty four-character UID text field.

RSS Dependent fields related to USER Header Object

The HEADER object type consists of information describing the user that includes information from the Resident Security System. These fields are dependent on the RSS itself and are defined to the specific RSS presiding over an E-SRF security Image.

These fields are formatted in the dictionary when a specific security IMAGE is CONFIGURED. Because there is no way to know the fields and characteristics, they are not shown in this reference. To find out what they are and determine their characteristics, please run report overlay ESRFDICT.

ACF2.*xxxxxxxx* — ACF2 RSS FDR data fields relative to a particular Image ID

Description: During the RSS E-SRF IMAGE configuration process, the ACF2's Field Definition Record (FDR) was used to map all FDR Logonid fields into the data dictionary.

When referencing an ACF2 Logonid Record field, specify ACF2.*xxxxxxxx.* The *xxxxxxxx* is the field name from the ACF2 FDR.

To get a list of these fields as they exist in E-SRF, use the E-SRF command processor to run the ESRFDICT report overlay. This overlay will print a list of the entire E-SRF Data Dictionary, which includes all Security Images and their associated Image Dictionaries. If these fields do not appear on this report, or are not synchronized with the *current* FDR and the Image's RSS is ACF2, this indicates that E-SRF was not properly configured for the Image in question.

Format: SINGLE: Each Logonid Record field will appear in the format defined in ACF2 for a particular Security Image.

RACF.*xxxxxxxx* — RACF RSS data fields relative to a particular Image ID

Description: During the RSS RACF IMAGE configuration process, names of particular RACF data were established into the particular IMAGE's data dictionary.

When referencing an RACF Userid record field, specify RACF.*xxxxxxxx.* The *xxxxxxxx* is the field name given the actual piece of data.

To get a list of these fields as they exist in E-SRF, use the E-SRF command processor to run the ESRFDICT report overlay. This overlay will print a list of the entire E-SRF Data Dictionary, which includes all Security Images and their associated Image Dictionaries.

Format: SINGLE: Each Userid Record field will appear in an assigned data format. Consult the Data Dictionary for the formats of these fields.

Related to USER Security Maintenance Summary Object

The SECURITY MAINTENANCE SUMMARY object type is a summary of security administration events. Only users who administer security changes for the Resident Security System will have this object. There will be a single array element for each day the user performed security maintenance.

The key for this object is: USERID

UB.CHANGES	**Total security changes performed**

Description: This field tracks the number of security maintenance events performed by the user.

Format: ARRAY: Appears as a standard numeric data field.

UB.DATE	**Security maintenance event(s) date**

Description: The date the array element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UB.DOMAIN	**Hosting Domain (sysid) where administration functions were carried out**

Description: Domain (*sysid*) where the security administration took place.

Format: ARRAY: Stored as character data.

UB.DS	**Total dataset access changes performed**

Description: This field tracks the number of all dataset access rule security maintenance events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.DS-ADD	**Total dataset access adds performed**

Description: This field tracks the number of dataset access rule security maintenance *add* events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.DS-CHG	**Total dataset access changes performed**

Description: This field tracks the number of dataset access rule security maintenance *change* events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.DS-DEL	**Total dataset access deletes performed**

Description: This field tracks the number of dataset access rule security maintenance *delete* events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.IMAGE **Security Image that hosted security administration**

Description: Security image relative to domain where the reported security administration event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

UB.OTHER **Total unknown security maintenance events performed**

Description: This field tracks the number of security administration events unknown to the current release of E-SRF the user performed on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.PASSWORD **Total password changes performed**

Description: This field tracks the number of userid password change events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.PROF-ADD **Total RSS "profile," adds performed**

Description: This field tracks the number of RSS "profiles" that were *added* during security administration activity on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.PROF-CHG **Total RSS "profile" changes performed**

Description: This field tracks the number of RSS "profiles" that were *changed* during security administration activity on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.PROF-DEL **Total RSS "profile" deletes performed**

Description: This field tracks the number of RSS "profiles" that were *deleted* during security administration activity on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.PROFILE **Total RSS "profile" changes performed**

Description: This field tracks the number of all dataset access rule security maintenance events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.RES **Total non-dataset resource access changes performed**

Description: This field tracks the number of all non-dataset resource rule security maintenance events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.RES-ADD **Total non-dataset resource access adds performed**

Description: This field tracks the number of non-dataset resource rule security maintenance *add* events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.RES-CHG **Total non-dataset resource access changes performed**

Description: This field tracks the number of non-dataset resource rule security maintenance *change* events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.RES-DEL **Total non-dataset resource access deletes performed**

Description: This field tracks the number of non-dataset resource rule security maintenance *delete* events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.SYS **Total system control changes performed**

Description: This field tracks the number of all Resident Security System operational system changes performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.SYS-ADD **Total system control adds performed**

Description: This field tracks the number of Resident Security System operational system *adds* performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.SYS-CHG **Total system control changes performed**

Description: This field tracks the number of Resident Security System operational system *changes* performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.SYS-DEL **Total system control deletes performed**

Description: This field tracks the number of Resident Security System operational system *deletes* performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.SYSTEM **Hosting Sysid (Domain) where administration functions were carried out**

Description: Sysid (Domain ID) where the security administration took place.

Format: ARRAY: Stored as character data.

UB.USER	**Total user changes performed**

Description: This field tracks the number of all user security maintenance events performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.USER -ADD	**Total user adds performed**

Description: This field tracks the number of user security maintenance *adds* performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.USER -CHG	**Total user changes performed**

Description: This field tracks total number of user security maintenance *changes* performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UB.USER -DEL	**Total user deletes performed**

Description: This field tracks the number of user security maintenance *deletes* performed by the user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

Related to USER Chronological Object

The CHRONOLOGICAL object type consists of array elements describing particular logged access event activity. Logged accesses include ALLOW but LOG and VIOLATION information. There are as many array elements as there were events for the number of retention days specified for this object type.

The key for this object is: USERID and IMAGE

UC.ACCESS	Type of resource access requested

Description: Indicates the "normalized" type of access requested. Please refer to Appendix B for a list of all ACCESS specifications.

Format: ARRAY: Appears as an eight-character keyword identifying the type of access requested.

UC.ACTION	Type of action taken based on RSS security controls in effect at access time

Description: Indicates the "normalized" action taken with respect to the requester's ability to access the requested resource in the manner dictated by the security request. Please refer to Appendix C for a list of all ACTION specifications.

Format: ARRAY: Appears as a nine-character keyword identifying the type of access requested.

UC.CLASS	Resource Class

Description: Identifies a particular resource class. All dataset resources are classified as DATASET, SAF classes are represented as they are.

ACF2 "generalized resource rule" *types* are stored as their three character type codes, followed by blank padding.

Format: ARRAY: Appears as an eight-character text field.

UC.DATE	Security event date

Description: The date the array element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UC.DOMAIN	User event hosting domain (sysid)

Description: Domain (sysid) where the user related event took place on.

Format: ARRAY: Stored as character data.

UC.ETF	EKC's ETF/* Status Information

Description: Indicates the actual ETF/* involvement status for this request (if any). All Status codes are listed in Appendix E of this guide.

Format: ARRAY: Stored as character data.

UC.ETF-ACT	EKC's ETF/* was active in this event

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility was involved in this event.

Format: ARRAY: Stored as a flag. Refer to "UC.ETF" for additional information.

UC.ETF-TEST	EKC's ETF/* performed a TEST access control function

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility logged this event as a result of a test access control request. If the flag is on, the logged access was determined using "test" access controls. The result of the test was NOT used to determine access to the resource.

Format: ARRAY: Stored as a flag. Refer to "UC.ETF" for additional information.

UC.IMAGE	Resource event Security Image ID

Description: Security image relative to domain where user related event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

UC.JOBNAME	Current executing job according to the Operating System

Description: The jobname as the operating system knows it. In MVS, this information comes from the name field of the "JOB" card.

Format: ARRAY: Eight character jobname.

UC.KEY	Resource access RSS KEY information

Description: This information represents what the RSS used to determine access to the resource. This field is a shortened (eight character) version of UC.PROFILE. Each RSS does this using a different approach and the data will represent this.

If the RSS is ACF2, and the resource was a DATASET, the KEY will be the RULEKEY used to determine the actual access. The RULEKEY is the actual eight-character VSAM record key for the RULESET. For other RESOURCE requests, the first eight characters of the resource key will be present.

If the RSS is RACF, this field represents the first eight characters of the RACF profile used in determining the access.

Please refer to UC.PROFILE, as this may be a better choice depending on your needs.

Format: ARRAY: Eight character RSS access control key information. Possible right truncated if the access key was longer than eight characters.

UC.KEYTOKEN	Resource access RSS KEY information TOKEN

Description: Provides actual token that represents the UC.KEY data.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: ARRAY: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this dataname relates to the tokenized RSS key (UC.KEY) data.

UC.PRI-UID	EKC's ETF/* used the PRIMARY UID to determine access

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility used the user's PRIMARY UID to determine access.

Format: ARRAY: Stored as a flag. Refer to "UC.ETF" for additional information.

UC.PROFILE	Resource access profile name information

Description: This information represents what the RSS used to determine access to the resource. This field represents the full forty-four-character profile name.

If the RSS is ACF2, and the resource was a DATASET, the PROFILE would be the eight character RULEKEY used to determine the actual access. For other resources, this field contains the resource rule RULEKEY n the following format: R(*ttt*)*resource_rule_key*.

If the RSS is RACF, this field represents the RACF profile in determining the access.

Format: ARRAY: Forty-four character "shrinkable" RSS profile name information. Possible right truncated if the access key was longer than forty-four characters, or if this field was shrunk to make it fit on a report.

UC.PROGRAM	Current executing program according to the Operating System

Description: The program name as the operating system knows it. In MVS, this means the "Load Module" name of the problem program that is hosting the request.

Format: ARRAY: eight-character load module program name.

UC.PROTOKEN	Profile Name TOKEN

Description: Provides actual token that represents the UC.PROFILE data.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: ARRAY: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this dataname relates to the tokenized RSS profile (UC.PROFILE) data.

UC.REASON	Actual reason action was taken by the RSS based on type of request made

Description: Indicates the actual reason action was taken. All reasons are unique. Knowledge of the actual request is not required. All action reason codes are listed in Appendix D of this guide.

Format: ARRAY: Appears as a twenty-character keyword identifying the RSS dependent reason for the action taken.

UC.RECTYPE E-SRF event type normalized classification

Description: E-SRF normalizes the event journalize reason into three classifications which are common to all Resident Security Systems.

ENTRY: Event logged as a System Entry (signon/signoff) request
LOG: Access granted and logged by normal validation procedures.
SPECIAL: Access granted and logged due to special conditioning
VIO: Access was denied

Consult the User's Guide for information on RECTYPE and other related normalization considerations.

Format: ARRAY: RECTYPE value assigned to this event by E-SRF.

UC.RESCOMMENT Resource Group Comment

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. (*see*: UC.RESGROUP). It will only be present if the grouping rule associated with this resource contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing actual names to resources such as CICS and IMS transaction Ids.

Format: ARRAY: Up to forty-eight characters of comment data.

UC.RESGROUP Current E-SRF resource owning group name

Description: The Resource Group name which is assigned to this resource. This group name is dynamically determined during processing it is not stored in E-SRF. The data presented is the result of interpreting the Resource grouping rules.

Format: ARRAY: sixteen-character group name.

UC.RESOURCE Resource name

Description: Provides the resource name involved in the event.

Format: ARRAY: Resource name text, currently maximum of 44 characters, left justified padded with right blanks. This field will "shrink down" if there is not enough room on a page to print the specified line.

UC.RESOWNER Current E-SRF resource owner ID

Description: The Resource Owner that owns the group which was assigned to this resource for this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: ARRAY: eight-character owner ID.

UC.RESTOKEN Resource name TOKEN

Description: Provides actual token that represents the UC.RESOURCE data.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: ARRAY: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this dataname relates to the tokenized RSS key (UC.RESOURCE) data.

UC.RSS	Resident Security System that posted this event

Description: The normalized name of the Resident Security System (RSS) which was responsible for posting this event. All RSS codes are listed in Appendix D of this guide.

Format: ARRAY: eight-character name of the Resident Security System and its primary component.

UC.SEC-UID	EKC's ETF/* used a SECONDARY UID to determine access

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ETF/* facility used one of the user's SECONDARY UID specifications to determine access.

Format: ARRAY: Stored as a flag. Refer to "UC.ETF" for additional information.

UC.SOURCE	Current location identification of request

Description: The source represents the name of the point to entry of the request. It may be a JES RJE *nodename*, or a particular VTAM terminal *nodename*. In the case of ACF2, it could be a source group name.

Format: ARRAY: eight-character source name.

UC.SRCCOMMENT	SOURCE Comment

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. (*see*: UC.SRCGROUP). It will only be present if the grouping rule associated with the source involved in the event contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing actual names to sources that are related to events.

Format: ARRAY: Up to forty-eight characters of comment data.

UC.SRCGROUP	Current E-SRF "source" owning group name

Description: The EKC Grouping Facility Group name which is assigned to the particular access "source" name. This group name is dynamically determined during processing. It is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: sixteen-character group name.

UC.SRCOWNER	Current E-SRF "source" owner ID

Description: The E-SRF Owner name that owns the group that was assigned to the source involved in this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: ARRAY: eight-character owner ID.

UC.SUBMITTR — Original userid who submitted request

Description: Validations may occur on behalf of other users. In this case, the userid that the event was posted against may or may not be the userid originating the request. Depending on how the journal was created, the "submitter's" userid may be available. If the submitter's userid is available, it will be provided in this field. If there was no submitter's id, or it was not available, this field will remain blank.

In many cases, if this field is blank, the userid associated with this event may have been the submitter.

Format: ARRAY: Stored as character data.

UC.SYSTEM — Resource event hosting sysid (Domain ID)

Description: Sysid (Domain ID) where the maintenance event took place on.

Format: ARRAY: Stored as character data.

UC.TIME — Time the event occurred

Description: The time the chronological event took place according to the Resident Security System.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UC.USRCOMMENT — USERID Comment

Description: The Resource Grouping Facility can provide "comment" data that may be processed using this dataname. (*see*: UC.USRGROUP). It will only be present if the grouping rule associated with the userid involved in the event contains comment data, or a default comment was present in the grouping *rule set.*

This data may be useful in providing additional information relating to the userid involved in the event being journalized.

Format: ARRAY: Up to forty-eight characters of comment data.

UC.USRGROUP — Current E-SRF "userid" owning group name

Description: The Resource Group name which was assigned to the particular userid named in this event. This group name is dynamically determined during processing, and is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

UC.USROWNER — Current E-SRF "user" owner ID

Description: The Resource Owner that owns the group that was assigned to the user involved in this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: *ARRAY Eight character owner ID.*

UC.VOLUME	**Resource volume information**

Description: Identifies the VOLUME information for a particular resource. This is a carryover from DASD (Direct Access Storage Device) and Magnetic Tape storage media.

Format: ARRAY: Appears as a six-character text field.

Related to USER ETF/ Firecall Detail Events Object*

The ETF/* "Firecall" facility creates a journal record each time a user activates and customizes it to suit the "Firecall" emergency needs. ETF/* refers to any of EKC's Emergency Access Facilities: ETF/A, ETF/R, or ETF/T.

E-SRF stores the majority of this information in an array allowing reporting of the use of the ETF/* Firecall facility.

To conserve space on the E-SRF Masterfile, only the first line of the general comment is stored on the Masterfile.

The key for this object is: USERID and IMAGE

UF.AUDIT	**EKC's ETF/* Firecall requested "AUDIT"**

Description: This is a status indicator that may be tested or displayed which indicates whether or not the user requested the AUDIT attribute.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

UF.AUTH-REQ	**EKC's ETF/* Firecall requested "authorization"**

Description: This is a status indicator that may be tested or displayed which indicates whether or not authorization was requested by the user.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

UF.DATE	**Firecall event date**

Description: The date the array element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UF.DOMAIN	**Firecall event hosting domain (sysid)**

Description: Domain (*sysid*) where the Firecall event took place on.

Format: ARRAY: Stored as character data.

UF.E/DATE	**Firecall event end (termination) date**

Description: The date the current Firecall session ended.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UF.E/TIME	**Firecall event end (termination) time**

Description: The time the current Firecall session ended.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UF.GROUP **EKC's ETF/* Firecall requested "GROUP"**

Description: This is a status indicator that may be tested or displayed which indicates whether or not the GROUP attribute was requested by the user.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

UF.GROUPID **EKC's ETF/* Firecall GROUP identifier data**

Description: The user's Group identifier data as entered to the Firecall session. This data has different meaning based on the RSS being processed. In ACF2, it represents the Firecall Universal Identification (UID) String created in ETF/A.

Format: SINGLE: Appears as a twenty four-character text field.

UF.JESID **Current executing job identification according to the Job Entry Subsystem**

Description: The JES ID (*as the operating system's Job Entry Subsystem* (JES) *assigned and knows it*). This is normally a three-character prefix, followed by a five position sequential number.

Format: ARRAY: Eight character JES job assigned identification.

UF.JOBNAME **Current executing job according to the Operating System**

Description: The jobname as the operating system knows it. In MVS, this information comes from the name field of the "JOB" card.

Format: ARRAY: Eight character jobname.

UF.NON-CNCL **EKC's ETF/* Firecall requested "Non Cancel" ability**

Description: This is a status indicator that may be tested or displayed which indicates whether or not the NON-CNCL "Firecall" attribute was requested by the user.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

UF.PASSWORD **EKC's ETF/* Firecall requested Password Change authority**

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ability to do password maintenance was requested by the user.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

UF.READALL **EKC's ETF/* Firecall requested "readall" authority**

Description: This is a status indicator that may be tested or displayed that indicates whether the ability to access any dataset for "READ" access was requested by the user.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

UF.REASON **Free form text user supplied when requesting EKC's ETF/* Firecall facility**

Description: This text was entered by the user when the Firecall request was made. Only line 1 is captured in E-SRF. Other lines are ignored for the sake of conserving space on the Masterfile.

Format: ARRAY: Eighty character reason text

UF.REQUEST **EKC's ETF/* Request codes**

Description: Indicates the actual Firecall request being made. All attribute codes are listed in Appendix F of this guide.

Format: ARRAY: Stored as character data.

UF.RSS **Resident Security System that posted this event**

Description: The normalized name of the Resident Security System (RSS) which was responsible for posting this event. All RSS codes are listed in Appendix D of this guide.

Format: ARRAY: eight-character name of the Resident Security System and its primary component.

UF.S/DATE **Firecall event start date**

Description: The date the current Firecall session started.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UF.S/TIME **Firecall event start time**

Description: The time the current Firecall session started.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UF.SECURITY **EKC's ETF/* Firecall requested "SECURITY" authority**

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ability to administer the security system was requested by the user.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

UF.SYSTEM **Firecall event hosting Sysid (Domain ID)**

Description: Sysid (Domain ID) where the Firecall event took place on.

Format: ARRAY: Stored as character data.

UF.TIME **Firecall event time**

Description: The time the array element is representing.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UF.USERID **EKC's ETF/* Firecall USERID string data**

Description: The user's USERID override data as entered to the Firecall session.

Format: SINGLE: Appears as a twenty four-character text field.

UF.X-AUTH	EKC's ETF/* Firecall Cross Authorization authority

Description: This is a status indicator that may be tested or displayed which indicates whether or not the ability to propagate the Firecall session across address spaces was requested by the user.

Format: ARRAY: Stored as a flag. Refer to "APPENDIX F" for additional information.

Related to USER Maintenance Object

The User Maintenance object type consists of array elements showing all maintenance events that occurred for a specific "userid" definition.

This object consists of new or changed resource definitions that were made to a specific RSS,

The information contained includes the changer's userid, the date, time, type of change, and the individual old/new user data elements that changed.

The key for this object is: USERID and IMAGE

UM.CHANGER	**Changer's userid**

Description: This field contains the Logonid of the user who made the change to the userid being reported on.

Format: ARRAY: Appears as an eight character standard userid.

UM.DATANAME	**The name of the data item changed**

Description: This field indicates the name of the item that was changed during userid maintenance

If the contents of this item is +ADMIN, this particular event represents a summary of the requested transaction. The detail items that were actually changed on the security database will follow. If all items presented in this transaction caused no change the security database, than this will be the only event that is reported.

Format: ARRAY: Eight-character maintenance request data item

UM.DATE	**Date the userid maintenance event occurred**

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UM.DOMAIN	**User maintenance event hosting Domain (sysid)**

Description: Domain (sysid) where the userid maintenance event took place on.

Format: ARRAY: Stored as character data.

UM.IMAGE	**Security Image ID**

Description: Security image relative to domain where the event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

UM.LOGONID	**Changer's Loginid**

Description: This field contains the Logonid of the user who made the change to the userid being reported on. This is the same as UM.CHANGER.

Format: ARRAY: Appears as an eight character standard userid.

UM.NEWDATA	User maintenance new data

Description: The information shown here is the actual new data related to the "dataname" that was applied to the security database for the named userid.

Please note, if UM.DATANAME is +ADMIN, the NEWDATA field is used to indicate the type of request performed, such as INSERT, ALTER or DELETE.

Format: ARRAY: Stored as a twenty-four-character left justified data item.

UM.OLDDATA	User maintenance old data

Description: The information shown here is the actual old data related to the "dataname" that was applied to the security database for the named userid. Please note that this data is not always available.

Please note, if UM.DATANAME is +ADMIN, the OLDDATA field is used to maintain summary statistics on what was actually altered against what was requested and is represented by "CHANGED: *nnn*, NOCHG: *nnn*".

The CHANGED number represents how many individual data items listed in the transaction were actually different and changed on the security database.

The NOCHG number represents how many individual data items were requested to be changed, but were identical and therefore were not changed. The sum of the two numbers shows how many items were presented for change.

Format: ARRAY: Stored as a twenty-four-character left justified data item.

UM.REQUEST	User maintenance request type

Description: The type of maintenance request made for this particular event.

Format: ARRAY: eight-character text field containing request type as specified for the Resident Security System. Please refer to *Appendix G* of this publication for information related to the particular Resident Security System responsible for the maintenance.

UM.RSS	Resident Security System that posted this event

Description: The normalized name of the Resident Security System (RSS) which was responsible for posting this event. . All RSS codes are listed in Appendix A of this guide.

Format: ARRAY: eight-character name of the Resident Security System and its primary component.

UM.SOURCE	Source maintenance event was performed from

Description: Identifies the "source" (place) the maintenance was performed on.

Format: ARRAY: eight-character text field (DELETE, INSERT or REPL

UM.SYSTEM	Resource maintenance event hosting sysid (Domain ID)

Description: Sysid (Domain ID) where the resource maintenance event took place on.

Format: ARRAY: Stored as character data.

UM.TIME	**Time the resource maintenance event occurred**

Description: The time an event took place according to the Resident Security System.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UM.TYPE	**Type of maintenance**

Description: Type of Resident Security System data that the maintenance was applied to.

Format: ARRAY: eight-character text field (DATASET, RESOURCE, SYSTEM or USERID).

UM.USERID	**Changer's Userid**

Description: This field contains the Userid of the user who made the change to the userid being reported on. This is the same as UM.CHANGER.

Format: ARRAY: Appears as an eight character standard userid.

UM.USRGROUP	**Current E-SRF "userid" owning group name**

Description: The Resource Group name which was assigned to the particular userid named in these events. This group name is dynamically determined during processing, and is not stored in E-SRF. The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

UM.USROWNER	**Current E-SRF "user" owner ID**

Description: The Resource Owner that owns the group that was assigned to the user involved in these events. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile.

Format: *ARRAY Eight character owner ID.*

Related to USER Profile Object

The Profile object type is used to store a list of RACF CONNECT groups and their attributes. Although this object contains array elements, they are not subject to ROLLOFF processing.

Each time a RACF group is CONNECTED to the user, the group (along with its various connect attributes) is inserted as an array element on this object.

When maintenance is performed which alters the connect attributes, the information is upgraded for the particular group.

When the connect group is removed (disassociated) from the user, the corresponding array element is removed from the object.

There can be as many connect group elements stored on the object as dictated by the ELEMENTS parameter. If an attempt is made to add more CONNECT groups than supported by the ELEMENTS parameter, the update is rejected. Under normal circumstances, this should never present itself as a problem.

Please note, the user's DEFAULT group is treated as a CONNECT group in RACF processing so therefore the DEFAULT group will be contained on this object.

The key for this object is: USERID and IMAGE

UP.ADSP	**RACF Automatic Data Set Protection attribute**

Description: Indicates the ADSP attribute is active for this CONNECTION.

Format: ARRAY: Stored as a "YES/NO" flag.

UP.AUDITOR	**RACF "auditor" attribute**

Description: Indicates the group AUDITOR attribute is active for this CONNECTION.

Format: ARRAY: Stored as a "YES/NO" flag.

UP.AUTH	**RACF "level of authority" for this group**

Description: This field contains the level of authority: USE CREATE CONNECT and JOIN the user is to have for this group.

Format: ARRAY: Stored as a character string.

UP.CONDATE	**Date user CONNECTED to this group**

Description: The date the user was CONNECTED to the group.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UP.DEFAULT	**Indicates group is the user's DEFAULT group**

Description: Indicates the group is defined as the user's DEFAULT group and was not actually CONNECTED in the classical form.

Format: ARRAY: Stored as a "YES/NO" flag.

UP.GROUP **RACF group name connected to the user**

Description: This is the actual RACF group that is "connected" to the user.

Format: ARRAY: Stored as an eight-character name.

UP.GRPACC **Indicates group access of Data Sets is in effect**

Description: Indicates the group access provision is in effect for this group, providing access to group datasets defined by this user to other members of the group.

Format: ARRAY: Stored as a "YES/NO" flag.

UP.NOTERMU **Indicates NOTERMUACC attribute**

Description: Indicates whether or not the NOTERMUACC attribute is in effect when connected to this group.

Format: ARRAY: Stored as a "YES/NO" flag.

UP.OPER **Indicates group OPERATIONS attribute**

Description: Indicates whether or not the GROUP OPERATIONS attribute is in effect when connected to this group.

Format: ARRAY: Stored as a "YES/NO" flag.

UP.OWNER **RACF user (or group) that owns the CONNECTION**

Description: This is the RACF user or group ID that "owns" the CONNECTION to the group.

Format: ARRAY: Stored as an eight-character name.

UP.RESDATE **CONNECTION resume date**

Description: The date the user will be permitted access to the system when CONNECTED to this group.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UP.REVDATE **CONNECTION revoke date**

Description: The date the user will be not be permitted access to the system when CONNECTED to this group.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UP.SPECIAL **Indicates group SPECIAL attribute**

Description: Indicates whether or not the GROUP SPECIAL attribute is in effect when connected to this group.

Format: ARRAY: Stored as a "YES/NO" flag.

UP.TYPE — Indicates "type" of group

Description: Indicates the type of group defined. Currently there may be two types of groups associated with a user. The normal RACF CONNECT group (which includes the user's DEFAULT group), and additional groups afforded by EKC's ETF/R Firecall facility.

Format: ARRAY: Stored as an eight-character description of the group type.

CONNECT The character string "CONNECT" will appear for normal RACF CONNECT groups.

FIRECALL The character string "FIRECALL" will appear for groups provided by the EKC ETF/R Firecall facility.

Please note: Firecall groups will not be associated with the user for access unless the user is running in FIRECALL mode as described in the EKC ETF/R product documentation.

UP.UACC — Default UACC specification

Description: This is the UACC that will be used as a default for resources the user defines while connected to this group.

Format: ARRAY: Stored as an eight-character text field.

Related to USER Recap Summary Object

The Recap Summary object type consists of array elements summarizing all events that took place *for a particular user* in a single array element. For example, if a given user had two hundred logged accesses and fifteen access violations, a single array element will be presented summarizing this information.

The key for this object is: USERID and IMAGE

UR.DATE	**Date the events occurred**

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UR.DOMAIN	**Host Domain (sysid)**

Description: Domain (sysid) where the events represented in this array occurred on.

Format: ARRAY: Stored as an eight-character data field.

UR.DS-LOG	**Number of dataset access loggings**

Description: Number of times dataset access was granted by the Resident Security System for a particular user with the condition of logging the event on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.DS-VIO	**Number of dataset access violation loggings**

Description: Number of times dataset access was denied by the Resident Security System for a particular user and the event was logged during a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.EXIT	**Number of installation user exit access allows**

Description: The installation exit granted access on a particular source to particular resources and logged this situation. This field represents how many times this occurred in a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.FIRECALL	**Number of ETF/* Firecall events**

Description: This count represents the number of interactions the user had with the ETF/* Firecall facility.

Format: ARRAY: Appears as a standard numeric data field.

UR.FIRST	Time of first event

Description: The time that the first event occurred for a particular user summarized in a particular array element took place on a given day.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UR.IMAGE	Security Image ID

Description: Security image relative to domain where the event took place.

Format: ARRAY: Determined by relating DOMAIN ID to current Domain assignments within the E-SRF control parameters.

UR.LAST	Time of last event

Description: The time that the last event occurred for a particular user summarized in a particular array element took place on a given day.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

UR.NON-CNCL	Number of NON-CNCL access allows

Description: The Resident Security System allowed an access to particular resources because the requester had the NON-CNCL attribute. This count represents the number of non-cancel access loggings that occurred for a particular userid on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.NOTAVAIL	Number of times signon was denied because userid was unavailable

Description: The Resident Security System will disallow a user from signing on the system because the userid is not available. The reasons vary with each RSS. An example would be the userid was canceled. This field represents the number of these situations that has occurred on a particular source on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.OWNED	Number of owned access allows

Description: The Resident Security System allowed access to particular resources because the requester owned it. This count represents the number of owned access loggings that occurred for a particular user on a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

UR.PGM-VIO **Number of violations due to unauthorized program usage**

Description: The Resident Security System disallowed access to particular resources because the requester accessed them from a program that the RSS deemed unauthorized. This count represents the number of this type of violation that occurred for a particular use on a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

UR.PSWD-VIO **Number of signon denials due to password violation**

Description: The Resident Security System disallowed signon by users due to invalid password attempts. This count represents the number of this type of violation that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.READALL **Number of forced "read-only" access allows**

Description: The Resident Security System allowed a "read-only" access to particular resources because the requester had the READALL attribute. This count represents the number of forced read only loggings that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.RES-LOG **Number of non-dataset resource access loggings**

Description: Number of times non-dataset resource access was granted by the Resident Security System for a particular user with the condition of logging the event in a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.RES-VIO **Number of non-dataset access violation loggings**

Description: Number of times non-dataset access was denied by the Resident Security System for a particular user and the event was logged in a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.RULE-LOG **Number of logged accesses granted by access control rules**

Description: Number of times access was granted by the Resident Security System based on access control rules with the condition that the access was logged for a particular user with the condition of logging the event on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.RULE-VIO **Number of access denials based on access control rules**

Description: Number of times access was denied by the Resident Security System based on access control rules for a particular user with the condition of logging the event on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SECURITY	Number of security officer forced access allows

Description: The Resident Security System allowed an access to particular resources because the requester was considered a security officer for a particular resource. This count represents the number of accesses granted for this reason.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

UR.SHFT-VIO	Number of access denials based on access control "shift" rules

Description: Number of times access was denied by the Resident Security System based on access control "shift" (time of day access) rules for a particular user in a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SIGN-SUS	Number of signon denials due to suspended userids

Description: The Resident Security System disallowed signon by users because their particular userids were suspended for one reason or another. This count represents the number of this type of violation that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SIGN-UKN	Number of unknown signon requests

Description: The Resident Security System disallowed signon for reasons unclassified by E-SRF. This count represents the number of this type of situation that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SIGN-VIO	Number of signon denials for all reasons

Description: The Resident Security System disallowed signon for users. This count represents the number of violations that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SIGNON	Number of signon requests performed

Description: The Resident Security System performed signon operations on behalf of users attempting to access the system. This count represents the number of signons performed for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SOURCE1	First of the last five sources hosting security events

Description: The last five sources that hosted a security event in a given day for a particular user are tracked in this array element. This is the first most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

UR.SOURCE2 Second of the last five sources hosting security events

Description: The last five sources that hosted a security event in a given day for a particular user are tracked in this array element. This is the second most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

UR.SOURCE3 Third of the last five sources hosting security events

Description: The last five sources that hosted a security event in a given day for a particular user are tracked in this array element. This is the third most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

UR.SOURCE4 Fourth of the last five sources hosting security events

Description: The last five sources that hosted a security event in a given day for a particular user are tracked in this array element. This is the forth most recent of the five.

Format: ARRAY: Appears as an eight-character userid text field.

UR.SOURCE5 Fifth of the last five sources hosting security events

Description: The last five sources that hosted a security event in a given day for a particular user are tracked in this array element. This is the fifth most recent of the five.

Format: *ARRAY: Appears as an eight-character userid text field.*

UR.SPECIAL Number of forced allows due to "special privileges" RECTYPE classification

Description: This represents the number of allowed accesses due to reasons that would cause E-SRF to assign a RECTYPE of SPECIAL to the event. Please refer to the *E-SRF User Guide* for additional information on what attributes would be assigned this classification.

Format: ARRAY: Appears as a standard numeric data field.

UR.SRCE-VIO Number of signon denials due to unauthorized source

Description: The Resident Security System disallowed the signon because the user was not authorized to access the system from the source (terminal identification or other source controls) in effect for the user. This count represents the number of this type of violation that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SUSPEND Number of times the user was suspended

Description: The Resident Security System suspended the user for reasons dictated by the security policy controls in effect for a specific RSS. This count represents the number of this type of violation that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

UR.SYSTEM Host Sysid (Domain ID)

Description: Sysid (Domain ID) where the events represented in this array occurred on.

Format: ARRAY: Stored as an eight-character data field.

UR.UNKNOWN	**Number security events that occurred that were unknown to E-SRF**

Description: The Resident Security System issued some sort of event or action that is unknown to E-SRF. This should always be zero unless either a new classification now exists in a particular RSS that E-SRF does not know about, or a USER EXIT was involved and altered enough information that E-SRF was unable to determine what type of event was taking place. This count represents the number of this type of violation that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

Related to USER Statistical Summary Object

The Statistical Summary object type consists of a single array element summarizing all journalized events that involved Userid resource access on a particular day. One of these array elements will be created for each USERID that a specific user accessed during a particular day contained on the current IMAGE.

For example, if a single user had journalized access loggings for one hundred resources on a given day, there will be one hundred array elements for the user on the day the access was attempted.

The key for this object is: USERID and IMAGE

US.ALLOWS	**Number of summarized events**

Description: Sum of all "RECTYPE" LOG, and SPECIAL events that occurred.

Format: ARRAY: This field is dynamically calculated on demand based on the sum of US.LOGS and US.SPECIAL. Appears as a standard numeric data field.

US.CLASS	**Resource Class**

Description: Identifies the particular resource class associated with this object. . All dataset resources are classified as DATASET; SAF classes are represented as they are.

ACF2 "generalized resource rule" *types* are stored as their three character type codes, followed by blank padding.

Format: ARRAY: Appears as an eight-character text field.

US.DATE	**Date the event occurred**

Description: The date the element is representing.

Format: ARRAY: Stored as a binary integer in absolute date format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

US.DOMAIN	**Resource event hosting Domain (sysid)**

Description: Domain (sysid) where the resource event took place on.

Format: ARRAY: Stored as character data.

US.EVENTS	**Total number of Journalized Security Events**

Description: Sum of all "RECTYPE" LOG, SPECIAL and VIOLATION events that occurred. This count is the total number of journalized event for this user for the current resource for the current day.

Format: ARRAY: This field is dynamically calculated on demand based on the sum of US.LOGS, US.SPECIAL and US.VIOS

US.EXIT	Number of installation user exit access allows

Description: The installation exit granted access to the particular resource and journalized it. This field represents how many times this occurred in a given day.

Format: ARRAY: Appears as a standard numeric data field.

US.FIRST	Time of first event

Description: The time that the first event summarized in a particular array element took place.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

US.LAST	Time of last event

Description: The time that the last event for a particular user summarized in a particular array element took place on a given day.

Format: ARRAY: Time of day format stored in a binary format. Display format is dictated by user options in the SET and RUN commands, as well as specified in the OWNER header.

US.LOGS	Number of "normal" access loggings

Description: Number of times access was journalized by the RSS for the particular user on a day. These loggings were classified as RECTYPE(LOG) by the RSS update program.

Format: ARRAY: Appears as a standard numeric data field.

US.NON-CNCL	Number of NON-CNCL access allows

Description: The Resident Security System allowed an access to particular resources because the requester had an attribute that would force allow access despite the security access controls in place for the resource. Access granted to a resource via the ACF2 NON-CNCL or RACF OPERATIONS attribute are examples of this type of logging. This count represents the number of such loggings that occurred for a particular userid on a given day.

Format: ARRAY: Appears as a standard numeric data field.

US.OWNED	Number of owned access allows

Description: The Resident Security System allowed access and journalized the access to a particular resource because the requester owned it. This count represents the number of such loggings that occurred for a particular user on a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

US.READALL **Number of forced "read-only" access allows**

Description: The Resident Security System allowed "read-only" access to particular resources because the requester had an attribute that permitted that type of access overriding the normal access controls governing the ability to obtain the resource for read only access. This count represents the number of such loggings that occurred for a particular user on a given day.

Format: ARRAY: Appears as a standard numeric data field.

US.RESGROUP **Group which this resource is associated with.**

Description: The Resource Group name assigned to this resource. This group name is dynamically determined during processing, (it is not stored on the E-SRF Masterfile). The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: sixteen-character group name.

US.RESOURCE **Resource name**

Description: Provides the resource name related to this object.

Format: ARRAY: Resource name text, currently maximum of 44 characters, left justified padded with right blanks.

US.RESOWNER **Owner ID of this resource**

Description: The Resource Owner that owns the group, which was assigned to this resource, for this event. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile, using the dynamically associated group ID.

Format: ARRAY: eight-character owner ID.

US.RESTOKEN **Resource name TOKEN**

Description: Provides actual token that represents the US.RESOURCE data.

This is NOT a field that you will be normally be reporting on. It is here in the event that it must be referenced for a special purpose.

Format: ARRAY: E-SRF Masterfile binary token representation of a specific piece of data. In this case, this dataname relates to the tokenized RSS key (UC.RESOURCE) data.

US.RULE-LOG **Number of journalized resource access logs granted via resource definitions.**

Description: The Resident Security System allowed an access based on its security resource access definitions. The particular granting definition journalized the event. This count represents the number of such loggings that occurred for a given day.

Format: ARRAY: Appears as a standard numeric data field.

US.SECURITY	Number of security officer forced access allows

Description: The Resident Security System allowed access to particular resources because the requester was considered a security officer. This count represents the number of such loggings that occurred for a given day.

NOTE: Not all RSS provide loggings for this event.

Format: ARRAY: Appears as a standard numeric data field.

US.SPECIAL	Number of "special" access loggings

Description: Number of times access was journalized by the RSS for the particular user on a day. These loggings were classified as RECTYPE(SPECIAL) by the RSS update program.

Format: ARRAY: Appears as a standard numeric data field.

US.SYSTEM	Sysid (Domain ID) hosting representing these events

Description: Sysid (Domain ID) where the maintenance event took place on.

Format: ARRAY: Stored as character data.

US.USRGROUP	Group that the user is related to

Description: The Resource Group name assigned to the user named in this array. This group name is dynamically determined during processing, (it is not stored on the E-SRF Masterfile). The data presented is the result of interpreting the Resource Grouping rules.

Format: ARRAY: Sixteen character group ID.

US.USROWNER	Current user's owner ID

Description: The Resource Owner that owns the group, that was assigned to the user named in the array. The owner ID was determined by the E-SRF GROUP/OWNER definitions contained on the Masterfile, using the dynamically associated user's group ID.

Format: *ARRAY Eight character owner ID.*

US.VIOS	Number of logged violation events

Description: The Resident Security System denied access to a particular resource. This count represents the number of violations that occurred for a given day.

Format: ARRAY: Appears as a standard numeric data field.

US.VOLUME	Resource volume information

Description: Identifies the VOLUME information for the particular resource related to this object. This is a carryover from DASD (Direct Access Storage Device) and Magnetic Tape storage media physical identification.

Format: ARRAY: Appears as a six-character text field.

Related to USER "TRACE" Chronological Object

The User "TRACE" Chronological object type consists of array elements describing a specific logged access event activity. Logged accesses include ALLOW but LOG and VIOLATION information, as well as all allowed activity. There are as many array elements as there are events for the number of retention days specified for this object type.

This record type is the result of activating the TRACE facility for a specific RSS.

Please note that not every RSS provides this facility. Additionally, not every RSS trace facility provides all the data that is represented in this object.

The key for this object is: USERID and IMAGE

An event trace is considered the same as an event journal. All fields carried in the User Chronological object are also contained in the User Trace object.

Please consult the User Chronological dictionary datanames to access this object. Instead of using UC as a dataname prefix, use UT instead.

For example, if you want the journal date, the UC.DATE dataname would be used, except, please specify UT.DATE instead.

Chapter 21: Appendix A: Resident Security System (RSS) Types

The following is a list of Resident Security System types known to E-SRF:

ACF2	ACF2 Generic
ACF2-SEV	ACF2 Signon/Signoff Processing
ACF2-RSC	ACF2 Generalized Resource Rule Processing
ACF2-DS	ACF2 Dataset Access Rule Processing
RACF	RACF Generic
RACF-SEV	RACF Signon/Signoff Processing
RACF-RSC	RACF Generalized Resource Rule Processing
RACF-DS	RACF Dataset Access Rule Processing
TSS	Top Secret Generic

This page intentionally left blank

Chapter 22: Appendix B: Access Specifications

Access Type	ACF2 Description	RACF Description
ADD	Resource rule service “READ”. Use is dependent on security implementation. In ACF2 CICS, it indicates the ability to read records from files defined in the CICS File Control Table (FCT)	**ADD**
ADDVOL	**N/A**	RACF resource “*ADDVOL*” access requested
ALLOCATE	Dataset rule granting CREATE ability.	RACF resource “*DEFINE*” access requested
ALTER	**N/A**	Resource access request with the RACF *ALTER* requirement.
CHGVOL	**N/A**	RACF resource “*CHGVOL*” access requested
CONTROL	**N/A**	Resource access request with the RACF *CONTROL* requirement.
DELETE	Resource rule service “DELETE”. Use is dependent on security implementation. In ACF2 CICS, it indicates the ability to delete records from files defined in the CICS File Control Table (FCT).	RACF resource “*DELETE*” access requested
DELVOL	**N/A**	RACF resource “*DELVOL*” access requested
EXEC	Dataset rule “EXECUTE” access. Ability to read the target dataset only to load programs into storage and execute them. They cannot be read for any other purpose.	Resource access request with the RACF *EXECUTE* requirement.
GENERAL	Resource rule access without service specification. Use is dependent on security implementation. In ACF2 CICS, this type of rules would apply to secured items such as transactions and programs.	**N/A**
MAINT	**N/A**	A RACF maintenance item was journalized
N/A		
NONE	**N/A**	Resource access request with the RACF *NONE* requirement.
READ	Dataset rule granting read ability. Resource rule service “READ”. Use is dependent on security implementation. In ACF2 CICS, it indicates the ability to read records from files defined in the CICS File Control Table (FCT).	Resource access request with the RACF *READ* requirement.
SIGNOFF	**N/A**	System entry access that includes any type of *SIGN OFF* operation.
SIGNON	System entry access that includes any type of *SIGN ON*.	System entry access that includes any type of *SIGN ON* operation.
UPDATE	Resource rule service “UPDATE”. Use is dependent on security implementation. In ACF2 CICS, it indicates the ability to update (alter) records from files defined in the CICS File Control Table (FCT).	Resource access request with the RACF *UPDATE* requirement.
WRITE	Dataset rule granting write ability.	**N/A**

This page intentionally left blank

Chapter 23: Appendix C: Action Specifications

Action type	ACF2 Description	RACF Description
DENIED	The Resident Security System denied the signon request.	The Resident Security System denied the signon request.
FIRECALL	Access was granted due to EKC Firecall facility	Access was granted due to EKC Firecall facility
LOG	General log	General log
LOG-RULE	The access was permitted by ACF2 rules, but journalized by rule request.	The access was permitted by RACF profiles. But journalized by profile request.
LOG-EXIT	The access was permitted by an installation written ACF2 user exit, but journalized for reporting purposes.	The access was permitted by an installation written RACF user exit, but journalized for reporting purposes.
LOG-NCNL	The access was permitted only because the user had the NON-CNCL attribute at the time of the access, which makes the access non cancelable.	The access was permitted only because the user had the OPERATIONS attribute at the time of the access, which makes the access non cancelable
LOG-OWND	**N/A**	**N/A**
LOG-RDALL	The access was permitted because the user had the READALL attribute at the time of the access.	The access was permitted because the user had the AUDITOR attribute at the time of the access.
LOG-SEC	The access was permitted only because the user had the SECURITY attribute at the time of the access.	**N/A**
NOT-AVAIL	Userid was not available for signon attempt because it was expired, canceled, or not active yet, the node was not available, or some other ACF2 reason.	Userid was not available for signon attempt because it was REVOKED or not active or some other RACF reason.
PASSWORD	The password specified during signon was invalid.	The password specified during signon was invalid.
PROGRAM	The program did not allowed access.	The application was not authorized
SEC-VIO	The access was denied due to ACF2 rule request to prevent or no rule condition.	The access was considered to be a violation to RACF.
SHIFT-VIO	The access was denied due to shift access controls.	The access was denied due to shift access controls.
SIGNOFF	N/A	A signoff was successfully performed.
SIGNON	A signon (System Entry Validation) was successfully performed.	A signon (System Entry Validation) was successfully performed.
SIGNON-R	A signon (System Entry Validation) was performed on behalf of a job submission (restricted signon).	**N/A**
SOURCE-ER	Request denied because source did not allow access.	Request denied because the source terminal or console was invalid
SUSPEND	Access denied because the userid was suspended at the time of the access attempt.	**N/A**
TRACE	ACF2 trace event.	RACF trace event
VIO	General violation	General violation

This page intentionally left blank

Chapter 24: Appendix D: Reason Specifications for ACF2

Part 1: ACF2 System Entry Validation

Information contained in this table represents return codes from ACF2 System Entry Validation (SEV) calls made by facilities attempting to "signon" users.

The REASON types are what appear in E-SRF reports. The ACF2 "code" is what ACF2 posts in the journal record created for the event. NOTES are a brief description of the condition.

To get more information about SEV return codes, please refer to the ACF2/MVS Messages Guide. The message prefix is ACF01, followed by the three character ACF2 code shown in this table.

Reason type	ACF2 code	Notes
ACF2 NOT RUNNING	Return code: 098	Signon request initiated before ACF2 was in a position to process the request. ACF2 may not be fully initialized
ALREADY IN USE	Return code: 022	Signon attempt on a facility that enforces a single use of a Logonid was attempted with a Logonid that was already signed on to that facility.
CANNOT ALTER PSWD	Return code: 021	Password expired and cannot be altered.
DENIED BY USER EXIT	Return code: 026	Access was denied because an installation provided exit routine prevented the access.
EXIT DENIED NEW PSWD	Return code: 037	Altering the Logonid's password was denied because an installation provided exit routine prevented the action.
EXIT FAILED REQUEST	Return code: 035	An installation provided exit routine was called and its completion code was 8 (fail the request).
EXPIRED PASSWORD	Return code: 017	The Logonid's password is expired.
INHERITANCE DENIED	Return code: 038	A job required the inheritance of a logonid that was not authorized to be inherited.
INVALID GROUPING	Return code: 039	Source validation error involving source grouping.
INVALID NEW PSWD	Return code: 018	Current password expired. New password syntax was invalid.
INVALID PSWD SYNTAX	Return code: 015	Password supplied was syntactically incorrect.
INVALID SOURCE	Return code: 032	Attempt to signon from an unauthorized source.
INVALID SUB-FUNCTION	Return code: 002	Programming error: Invalid ACVALD sub-function code.
LOGONID CANCELED	Return code: 010	Logonid used for signon was CANCELLED.
LOGONID EXPIRED	Return code: 014	Logonid used for signon was EXPIRED.
LOGONID HAS STC	Return code: 030	Logonid used for signon had the STC attribute.
LOGONID NOT ACTIVE	Return code: 025	Logonid used for signon was not "active". ACTIVE date was still in the future.
LOGONID REQUIRED	Return code: 016	Signon attempted with no Logonid.
LOGONID SUSPENDED	Return code: 011	Logonid used for signon was SUSPENDED.
INVALID EXIT RETURN	Return code: 036	An invalid return code from an installation provided exit routine was detected.
MINDAYS ERROR	Return code: 136	Password change attempted too soon after a previous password change.
NEW PSWD = EXP PSWD	Return code: 023	Attempted to change an expired password with the Logonid's current expired password.
NEW PSWD = CUR PSWD	Return code: 131	Attempted to change a password with the Logonid's current password.

NEW PSWD NOT ALLOWED	Return code: 132	Site does not allow password changes during signon.
NEW PSWD NOT SET	Return code: 128	New password supplied was syntactically incorrect
NEW PSWD SHORT	Return code: 019	New Password supplied was less than the site's minimum length. New password was required, signon not performed.
NEW PSWD SHORT	Return code: 130	New Password supplied was less than the site's minimum length. Password not changed, signon continues.
NJE INVALID NEW PSWD	Return code: 033	New password supplied during NJE signon was syntactically incorrect.
NJE NEW PSWD SHORT	Return code: 034	New Password supplied during NJE signon was less than the site's minimum length. Password not changed, signon continues.
NO INHERITED LOGONID	Return code: 097	No Logonid to inherit and no DEFAULT to specified on system signing on to.
NO SHIFT RECORD	Return code: 063	SHIFT record specified on Logonid could not be located.
NO STC ATTRIBUTE	Return code: 031	Signon for a Started Task was denied because Logonid did not have the STC attribute.
NO ZONE RECORD	Return code: 060	ZONE record specified on Logonid could not be located.
NOT AUTH FOR GROUP	Return code: 100	Signon denied because a group access was requested that was not authorized.
NOT AUTHORIZED	Return code: 001	Facility requesting the signon operation was not permitted to issue signon requests.
OFF SHIFT	Return code: 061	Signon was attempted outside the TIME specified by the Logonid's SHIFT record.
OFF-SHIFT OVERRIDE	Return code: 135	Signon was allowed outside SHIFT controls because the Logonid had the LOGSHIFT attribute.
PASSWORD ALTERED	Return code: 129	Password was altered during signon operation.
PASSWORD NOT ALLOWED	Return code: 006	Signon with a password was attempted for a Logonid that had the RESTRICT attribute.
PASSWORD REQUIRED	Return code: 007	A signon was attempted without a password when a password was required.
PASSWORD VIO SUSPEND	Return code: 013	Logonid was suspended because of excessive password violations.
PASSWORD WILL EXPIRE	Return code: 134	Signon completed with a password expiration warning.
PGM SUBMISSION ERR	Return code: 009	Signon attempted for a Logonid by a program that does mot match the PROGRAM criteria controls that were in place for the Logonid.
PSWD EXPD NEW SHORT	Return code: 020	Old password was expired and new Password supplied was less than the site's minimum length.
PSWD VERIFY FAILURE	Return code: 029	Password re-verification request failed because the supplied password was not matched.
SOURCE UNAUTHORIZED	Return code: 008	Signon attempted from a non-APF authorized program with a Logonid that was provisioned to be signed on by an APF authorized program only.
UNMATCHED PASSWORD	Return code: 012	Signon request failed because the supplied password was not matched.
USER AUTH DENIAL	Return code: 055	Signon request failed because a user authentication facility routine (*normally site specified*) denied the request.

Part 2: ACF2 Resource Access Validation

Information contained in this table represents return codes from ACF2 resource validation processing that occur when access to resources are attempted.

The REASON types are what appear in E-SRF reports. The "Dataset Code" is the name provided by the ACF2 SMF dataset access journal record DSECT. The number afterwards is the actual code passed in the SMF journal record.

The RES column indicates non-dataset resource validation also produces this type of journal.

NOTES are a brief description of the condition.

Reason type	Dataset Code		Res	Notes
@BLPPGM	A$SLD110	10		**BLP-PGM** Access permitted because program requesting dataset access was permitted to open dataset with BYPASS LABEL PROCESSING specified.
@MAINT	A$SLCD19	9		**MAINT-PGM** Access permitted because access environment matches a GSO MAINT record specification.
BAD TAPE LABEL	A$SLD114	14		Access was prevented because the tape label could not be read during OPEN.
CMD MOD NOT FROM APF	A$SLD118	18		**NON-APF** Access was prevented because the command module requested was not loaded from an APF authorized library.
DASD NOT RES/SECVOL	A$SLCD17	7		**DASDUNSC** Access permitted because access was for an unprotected volume and no DSNGEN exit was taken.
DUMP NOT AUTHORIZED	A$SLD236	136		**DUMPAUTH** Access was prevented because the user attempted to create a dump and did not have the proper access rights (such as DUMPAUTH).
INSTALLATION EXIT	A$SLD231	131	**R**	**EXITVIO** Access was prevented because the site user exit denied access to the dataset.
INVALID ACSXP PARM	A$SLD232	132		Access was prevented because the parameters specified in the ACSXP (exit parameter block) were invalid.
INVALID CMD	A$SLD116	16		Access was prevented because the command structure was invalid.
INVALID JOB/STEP LIB	A$SLD115	15		**INV-JSL** Access was prevented because the program pathing feature was unable to correctly determine the library for the JOBSTEP program. An invalid return code was detected from one of the information gathering routines of BLDL. The library was defaulted to SYS1.LINKLIB.
INVALID TMP	A$SLD117	17		**INV-TMP** Access was prevented because the control block structure for the TSO TMP was found to be invalid. One of the programs that make up the TMP (or front-ends the TMP) was not from an APF authorized library).

LMP LOGGING	A$SLD153	53		**LMPLOG** Access was permitted because ACF2 was in LOG MODE due to a LMP product key failure.
LOGGED ACCESS	A$SLCD10	0		Normal LOG/VIO record.
LOGGED DUE TO EXIT	A$SLD154	54		Journalized due to EXIT request.
MAX VIO REACHED	A$SLD160	60		**MAXVIO** Access was prevented because the maximum number of violations has been reached for the current executing job.
MODE = LOG	A$SLD254	154		Access was permitted because the site is in LOG MODE.
MODE = QUIET	A$SLD253	153		Access was permitted because the site is in QUIET MODE.
MODE = WARN	A$SLD255	155		Access was permitted because the site is in WARN MODE.
NEXTKEY LOOP	A$SLD238	138		**NKEYLOOP** Access was prevented because the NEXTKEY parameter pointed to itself.
NEXTKEY OVERFLOW	A$SLD257	157		**KEYEXCES** Access was prevented because the number of NEXTKEY specifications exceeded an ACF2 implied limit.
NO ACUCB	A$SLD112	12		**NOACUCB** Access was permitted because the ACUCB control area could not be located and the console operator acknowledged the access.
NO ALLOW RULE	A$SLD229	29	**R**	**NORULE** Access was prevented because no rule line matched the environment.
NO CDE FOR PROGRAM	A$SLD123	23		**NOCDE** Access was prevented because no CDE was available to determine the active program name for program pathing.
NO MODE = ABORT	A$SLD247	147		**NOMODEAB** Access was denied because the site was in RULE MODE and the *no-$MODE* parameter in the GSO OPTS MODE record indicated ABORT.
NO MODE = LOG	A$SLD245	145		**NOMODELG** Access was permitted because the site was in RULE MODE and the *no-$MODE* parameter in the GSO OPTS MODE record indicated LOG.
NO MODE = QUIET	A$SLD244	144		**NOMODEQT** Access was permitted because the site was in RULE MODE and the *no-$MODE* parameter in the GSO OPTS MODE record indicated QUIET.
NO MODE = WARN	A$SLD246	146		**NOMODEWN** Access was permitted because the site was in RULE MODE and the *no-$MODE* parameter in the GSO OPTS MODE record indicated WARN.
NO REC = ABORT	A$SLD251	151		**NORECAB** Access was prevented because the site was in RULE MODE and no rule set applied, and the *no-rule* parameter in the GSO OPTS MODE record indicated ABORT.
NO REC = LOG	A$SLD249	149		**NORECLG** Access was prevented because the site was in RULE mode and no rule set applied, and the *no-rule* parameter in the GSO OPTS MODE record indicated

				LOG.
NO REC = QUIET	A$SLD248	148		**NORECQT** Access was prevented because no rule set existed and the GSO OPTS MODE was set to (RULE,QUIET,no-$mode).
NO REC = WARN	A$SLD250	150		**NORECWR** Access was prevented because no rule set exists and the GSO OPTS MODE is set to (RULE,WARN,no-$mode).
NO RULESET	A$SLD230	130	**R**	**NORECORD** NO access rule set existed that matched the environment. The site was in ABORT, LOG or WARN (not RULE mode) and the access rule set did not exist when the access was attempted.
NON-CANCEL	A$SLCD12	2	**R**	**NON-CNCL** Access was permitted because the requester had the NON-CNCL logonid privilege.
NOT ON @PPPGM LIST	A$SLD111	11		**NOTPPGM** Access was permitted because program was not on the site's protected program list.
O/S SPECIAL	A$SLD113	13		**SPECIAL** Access was permitted because of special authority associated with the request. This request normally applies to some implicit operation performed on behalf of a user who did not directly request the access.
ON @PPGM LIST	A$SLD234	134		**PPGMVIO** Access was prevented because a user requested access to a program on the site's TSO PPGM list and did not have the required authority.
OWNED PREFIX	A$SLCD14	4		**OWNED** Access was permitted because the dataset's high level index matches the user's PREFIX mask in the logonid record.
PROGRAM PATHING ERR	A$SLD119	19		**PATHERR** Access was prevented because of an invalid or unknown program pathing error.
PROTECTED PROGRAM	A$SLD124	24		**PGM-LOG** Access was permitted despite PPGM violation because the user had PPGM in the Logonid.
READALL	A$SLCD13	3		**READALL** Access was permitted because the user had the READALL logonid privilege.
RULE $MODE(ABORT)	A$SLD243	143		**$MODEAB** Access was prevented because a $MODE (ABORT) statement was specified in the rule.
RULE $MODE(LOG)	A$SLD241	141		**$MODELG** Access was permitted because the rule contained a $MODE(LOG) specification.
RULE $MODE(QUIET)	A$SLD240	140		**$MODEQT** Access was permitted because the rule contained a $MODE(QUIET) specification.
RULE $MODE(WARN)	A$SLD242	142		**$MODEWR** Access was permitted because the rule contained a $MODE(WARN) specification.
RULE ALLOW	A$SLCD15	5		**ACCESS** Access was permitted because the access rule that permitted access was found to match the environment.
RULE ALLOW WITH LOG	A$SLD252	152	**R**	Access to a dataset was permitted by a rule and was journalized because the rule specified LOG.

RULE INTERPRET ERROR	A$SLD161	61	**R**	Access was prevented because an error occurred during the rule interpretation process.
RULE PREVENT	A$SLD228	128	**R**	**NOACCESS** Access was prevented because the rule requested prevent.
SCOPE INIT FAILURE	A$SLD121	21		Access was prevented because scoping was required to determine the access and the scope record was unable to be built.
SCOPE PROCESS FAILED	A$SLD122	22		Access was prevented because scoping was required to determine the access and the scoping process encountered an error.
SCOPED SECURITY	A$SLD120	20	**R**	**SCOPESEC** Access was permitted because the user is a scoped security officer and the scope matched access requirements.
SECURITY	A$SLCD11	1		**SECURITY** Access was permitted because the user had the SECURITY logonid privilege.
TAPE NOT SEC	A$SLCD18	8		**TAPEUNSC** Access was permitted because the tape volume was unprotected and TAPEDSN was set to NO.
TAPEBLP UNAUTHORIZED	A$SLD233	133		**BLPVIO** Access was prevented because the user requested BLP (Bypass Label Processing) and was not authorized to do so.
TEST CMD NOT AUTH	A$SLD237	137		**NOTEST** An invalid path for dataset access existed. A rule permitted access to a dataset through program pathing but the program was executed under TSO TEST and the user did not have the DUMPAUTH attribute.
UNCONDITIONAL ABORT	A$SLD190	90		
USER EXIT ALLOW	A$SLCD16	6	**R**	**EXITALLW** Access was permitted because the site's user exit permitted the access.

Part 3: ACF2 System Extension Reasons

Information contained in this table represents return codes provided by journals created by processing function "extensions". Extensions are third party products that augment ACF2 processing.

The REASON types are what appear in E-SRF reports. NOTES are a brief description of the condition.

To get more information about these journals, please consult the appropriate documentation provided by the extension posting the journal.

Reason type	Notes
E-PAL ALLOW	E-PAL granted the access
ETFA FIRECALL ACCESS	ETF/A's Firecall granted the access
UNKNOWN	Reason unknown to E-SRF

Chapter 25: Appendix D: Reason Specifications for RACF

Part 1: RACF System Entry Validation

The descriptions contained on this list describe the Event 1 Qualifier Codes resulting from a LOGON or LOGO RACINIT SAF call.

Reason type	Event 1	Notes
EXPIRED PASSWORD	25	Current password has expired
GOOD RACINIT DELETE	13	Successful RACINIT delete
GOOD RACINIT INIT	12	Successful RACINIT initiation
GOOD SIGNOFF	08	Successful termination
GOOD SIGNON	0	Successful Initiation
GROUP ACCESS REVOKED	28	Group access has been revoked
INIT WITH PASSTICKET	32	Successful initiation using Pass Ticket
INSUF SEC LABEL AUTH	20	WARNING - Insufficient security label authority
INVALID APPLICATION	5	Invalid application
INVALID GROUP	2	Invalid group
INVALID NEW PASSWORD	26	Invalid new password
INVALID OID CARD	3	Invalid OIDCARD
INVALID PASSWORD	1	Invalid password
INVALID SOURCE	4	Invalid terminal/console
MISSING SEC LABEL	21	WARNING - Security label missing from user, job or profile
NEEDS MORE AUTH	14	System now requires more authority
NJE JOB NOT AUTH	30	Network Job Entry - Job not authorized
NOTAUTH RJE	15	Remote Job Entry - job not authorized
NOTAUTH SEC LABEL	11	Not authorized to security label
NOTAUTH TO SEC LABEL	18	Submitter not authorized to security label
NOTAUTH TO SEC LABEL	22	WARNING - Not authorized to security label
OID CARD IS REQUIRED	29	OIDCARD is required
PASSTICKET REPLAY	33	Attempted replay of Pass Ticket
REVOKED USER	6	Revoked user attempting access
SEC LABEL BAD	10	Insufficient security label authority
SEC LABEL NOT COMP	23	Security labels not compatible
SUB NOTAUTH USER	17	Submitter is not authorized by user
SURROGATE CLS INACT	16	SURROGAT class is inactive
UNK USER FROM NODE	31	WARNING - Unknown user from trusted node propagated
UNKNOWN USERID	09	Undefined User ID
USER NOTAUTH TO JOB	19	User is not authorized to job
USER NOW REVOKED	7	Userid automatically revoked
VER FAILED BY INST	27	Verification failed by installation
W SEC LABEL NOT COMP	24	WARNING - Security labels not compatible

Part 2: RACF Resource Access validation

The descriptions contained on this list describe the Event 2 Qualifier Codes resulting from a resource validation issued via RACHECK and DIRAUTH function and VMXEVENT auditing.

Reason type	Event 2	Event 3	Event 4	Event 5	Event 6	Event 7	Notes
ALREADY DEFINED			04			4	Rename target resource already defined
DATASET NOT CATALGED	10						WARNING – dataset not cataloged
FAILED BY PROTECTALL	4						Access failed due to PROTECTALL
GROUP UNDEFINED			1			2	Invalid RACF GROUP
INS CAT OR SECLEVEL	6						Insufficient CATEGORY/SECLEVEL
INS CAT OR SECLEVEL	2						WARNING – Insufficient CATEGORY/SECLABEL
INS SECLBL AUTH	7	2	10 14			11	Insufficient security label authority
INVALID VOLUME				2			Invalid volume identification
LS PROF DIF SECLBL		4	9			12	Less specific profile exists with a different security label
LOGGED ACCESS	0						Successful resource access
NEWNAME NOT SECLBL			12				New name not protected by security label
NOT AUTHORIZED	1	1	3			3	Insufficient authority
NOT PROTECTED						6	Resource not protected
NOT PROT BY SECLBL			11				Resource not protected by security label
PROFILE NOT FOUND	2						Profile not found – RACFIND specified on macro
PROFILE NOT FOUND	11						Profile not found – Required for authority checking
RESOURCE NOT FOUND				1			Target resource not found
SECLBL DOMINATE OLD			13				New security label must dominate old security label
SUCCESSFUL DEFINE						0	Successful definition
SUCCESSFUL DELETE				0	0		Successful scratch
SUCCESSFUL NEWVOL		0					RACF "*NEWVOL*" request was successful
SUCCESSFUL RENAME			0				RACF "*RENAME*" request was successful
UNDEFINED USER			5			5	User not defined to RACF
USER NOT IN GROUP			2			2	User not in RACF group
WARN BY PROTECTALL	5						Warning issued due to PROTECTALL
WARN MODE ALLOW	3						Access permitted due to warning
2Q USER UNDEFINED			8			10	User in second qualifier is not RACF defined
W INS SECLBL AUTH	9					9	
W SECLBL DOMINATE OLD			17				Warning: New security label; must dominate old security label

W NEWNAME NOT SECLBL			16				Warning: New name not protected by security label
W NOT PROTECTED			7			7	WARNING: Resource not protected
W NOT PROT BY SECLBL			15				WARNING: Resource not protected by security label
W SECLBL MISSING	8					8	WARNING – Security label missing from job, user or profile

Part 3: RACF System Extension Reasons

Reason type	ACF2 Code	Notes
E-PAL ALLOW		E-PAL granted the access
ETF/X FIRECALL ACCESS		ETF/A's Firecall granted the access
UNKNOWN		Reason unknown to E-SRF

This page intentionally left blank

Chapter 26: Appendix E: EKC's ETF/* Status Codes

The following is a list of Status Codes for EKC's Security system product extensions. (The second column refers to the code seen in ETF/*x* reports.)

E_PAL		Access controls provided by the E-PAL security system extension
ACTIVE		ETF/A active for this event
ALT-UID	-M	ETF/A was involved with this access using the user's **ALTERNATE UID** value
FIRECALL	-F	ETF/A **FIRECALL** facility was involved in this event
PRI-UID	-P	ETF/A was involved with this access using the user's **PRIMARY UID** value
TEST-ALT-UID	-MT	ETF/A was involved with this access in **TEST RULE** mode with this access using the user's **PRIMARY UID** value.
TEST-PRI-UID	-PT	ETF/A was involved with this access in **TEST RULE** mode with this access using the user's **SECONDARY UID** value.

This page intentionally left blank

Chapter 27: Appendix F: EKC's ETF/* FIRECALL function requests

The following is a list of FUNCTION CODES for EKC's ETF/* Firecall facility

ALTER	**ALTER** ETF/* FIRECALL session Firecall options and requests were altered
CANCEL	**CANCEL** (terminate) ETF/* FIRECALL session Firecall was terminated on the TSO session
INITIATE	**INITIATE** ETF/* FIRECALL session Firecall was started up under a TSO session

This page intentionally left blank

Chapter 28: Appendix G: Maintenance Request types for ACF2

Request type	Description	Notes
DELETE	Delete existing definition	
INSERT	Insert a new definition	
REPLACE	Replace existing definition	

This page intentionally left blank

Chapter 29: Appendix G: Maintenance Request types for RACF

The descriptions contained on this list describe RACF maintenance commands as presented to SMF Journalizing by RACF.

Request type	Description	Notes
ADDVOL		
CHGVOL		
DEFINE		
NEWNAME		
DELETE		
ADDSD		
ADDGROUP		
ADDUSER		
ALTDSD		
ALTGROUP		
ALTUSER		
CONNECT		
DELDSD		
DELGROUP		
DELUSER		
PASSWORD		
PERMIT		
RALTER		
RDEFINE		
RDELETE		
REMOVE		
SETROPTS		
RVARY		

This page intentionally left blank

Chapter 30: Appendix H: RACF User Header Image dictionary entries

The following table shows the E-SRF User header object Image Data Dictionary entries and how they are constructed and maintained on the Masterfile.

Dictionary name	Shelled in	Synchronize	Journal Update process	Description
RACF.NAME	Current Name	0200 extract	E10/13: 0 and 2 and from Transaction	User's name
RACF.USER	Current Userid	any presence	From transaction	User's userid
RACF.UID	String of OWNER, Default Group and Userid	String of OWNER, Default Group and Userid	Maintained as appropriate based on presence of data from	Universal user identification information used for grouping
General				
RACF.ACC.DATE	ZEROS	0200 extract	E01: 0, 12 and 32	User's last system entry access date
RACF.ACC-CNT	ZEROS	ZEROS	E01: 0, 12 and 32	Times successful signon occurred
RACF.ACC-DATE	ZEROS	any presence		Date of last known system access
RACF.ACC-FRI	YES	0200 extract		Access system on Friday
RACF.ACC-FROM	ZEROS	0200 extract		User access start time
RACF.ACC-MON	YES	0200 extract		Access system on Monday
RACF.ACC-SAT	YES	0200 extract		Access system on Saturday
RACF.ACC-SUN	YES	0200 extract		Access system on Sunday
RACF.ACC-THU	YES	0200 extract		Access system on Thursday
RACF.ACC-TIME	ZEROS	0200 extract	E01: 0, 12 and 32	User's last system entry access time
RACF.ACC-TO	ZEROS	0200 extract		User access end time

RACF.ACC-TUE	YES	0200 extract		Access system on Tuesday
RACF.ACC-WED	YES	0200 extract		Access system on Wednesday
RACF.ADSP	NO	0200 extract	E10/13: 0 and 2	ADSP
RACF.ALT-DATE	Trans Date	ZEROS	E13: 0 and 2	Userid Creation Date
RACF.ALT-TIME	Trans Time	ZEROS	E13: 0 and 2	Userid Creation Time
RACF.AUDITOR	NO	0200 extract	E10/13: 0 and 2	AUDITOR
RACF.CRE-DATE	Trans Date	0200 extract	E10: 0 and 2	Userid Creation Date
RACF.CRE-TIME	Trans Time	ZEROS	E10: 0 and 2	Userid Creation Time
RACF.DFTGROUP	Current group	0200 extract	E10/13: 0 and 2 Also upgrades the ESRF universal identifier 9-16	User's Default Group
RACF.GRPACC	NO	0200 extract	E10/13: 0 and 2	GRPACC
RACF.INSTDATA	Blanked out	0200 extract	E10/13: 0 and 2	Installation data
RACF.LOGALL	NO	0200 extract	E10/13: 0 and 2	Log all access activity.
RACF.MAXDAYS	ZEROS	0200 extract	E18: 1 and 2	Password change interval
RACF.MODEL	Blanked out	0200 extract		Default dataset profile model
RACF.OIDCARD	NO	0200 extract	E10/13: 0 and 2	OIDCARD
RACF.OPER	NO	0200 extract	E10/13: 0 and 2	OPERATIONS
RACF.OWNER	!ESRF	0200 extract	E10/13: 0 and 2 Also upgrades the ESRF universal identifier 1-8	User's Owner
RACF.PRI-LANG	Blanked out	0200 extract		Primary language
RACF.PSWD-CNT	ZEROS	ZEROS	E18: 0 and 2	Times password was successfully changed
RACF.PSWD-DAT	ZEROS	0200 extract	E18: 0 and 2	Last password change date.
RACF.PSWD-GEN	ZEROS	0200 extract		Password generation count.

RACF.PSWD-TIM	ZEROS	ZEROS	E18: 0 and 2	Last password change time.
RACF.PVIO-CNT	ZEROS	ZEROS	E01: 1	Times password violations occurred
RACF.PVIO-DAT	ZEROS	ZEROS	E01: 1	User's last password violation date
RACF.PVIO-TIM	ZEROS	ZEROS	E01: 1	User's last password violation time
RACF.RES-DATE	ZEROS	0200 extract	E10/13: 0 and 2	User RESUME effective date
RACF.RESTRICT	NO	0200 extract	E10/13: 0 and 2	Ability to signon without a password
RACF.REV-DATE	ZEROS	0200 extract	E10/13: 0 and 2	User REVOKE effective date
RACF.REVOKE	NO	0200 extract	E10/13: 0 and 2	REVOKED
RACF.SECLABEL	Blanked out	0200 extract		Security label
RACF.SEC-LANG	Blanked out	0200 extract		Secondary language
RACF.SECLEVEL	ZEROS	0200 extract		Security level
RACF.SPECIAL	NO	0200 extract	E10/13: 0 and 2	SPECIAL
RACF.SVIO-CNT	ZEROS	ZEROS	E02: 1, 2, 4, 6, 7, 11 and 12 E03: 1, 2 and 3 E04: 1, 2, 3, 8, 9, 10, 13 and 14 E05: 2 E07: 1, 2, 3, 10, 11 and 12	Times a resource access security violation occurred.
RACF.SVIO-DAT	ZEROS	ZEROS	E02: 1, 2, 4, 6, 7, 11 and 12	User's last security violation date
RACF.SVIO-TIM	ZEROS	ZEROS	E02: 1, 2, 4, 6, 7, 11 and 12	User's last security violation time

DFP Segment				
RACF.DFPAPPL	Blanked out	0210 extract		DFP application name
RACF.DFPCLASS	Blanked out	0210 extract		DFP data class name
RACF.DFPMGMT	Blanked out	0210 extract		DFP management class name
RACF.DFPSTOR	Blanked out	0210 extract		DFP storage class
TSO Segment				
RACF.TSOACCT	Blanked out	0220 extract		TSO account number
RACF.TSOCMND	Blanked out	0220 extract		TSO Command character string
RACF.TSODEST	Blanked out	0220 extract		TSO DESTINATION
RACF.TSODLBL	Blanked out	0220 extract		TSO logon security label
RACF.TSOHCLAS	Blanked out	0220 extract		TSO HOLDCLASS
RACF.TSOJCLAS	Blanked out	0220 extract		TSO JOBCLASS
RACF.TSOMCLAS	Blanked out	0220 extract		TSO MESSAGECLASS
RACF.TSOMSIZE	ZEROS	0220 extract		TSO maximum REGION size
RACF.TSOOCLAS	Blanked out	0220 extract		TSO SYSOUTCLASS
RACF.TSOPREF	ZEROS	0220 extract		TSO performance group
RACF.TSOPROC	Blanked out	0220 extract		TSO default logon PROCNAME
RACF.TSOSIZE	ZEROS	0220 extract		TSO default REGION size
RACF.TSOUDATA	Blanked out	0220 extract		TSO nibblized hex data
RACF.TSOUNIT	Blanked out	0220 extract		TSO default UNIT

RACF.CICSCLAS	all set to NO	0231 extract		CICS BMS Operator class flags
RACF.CICSOID	Blanked out	0230 extract		CICS BMS Operator identifier
RACF.CICSPRTY	ZEROS	0230 extract		CICS operator related transaction priority
RACF.CICSTOUT	ZEROS	0230 extract		CICS terminal related timeout HH;MM
RACF.NOFORCE	NO	0230 extract		CICS Extended recovery NOFORCE in effect.
OMVS Segment				
RACF.ADRSPMAX	ZEROS	0270 extract		ASSIZEMAX: address space size RLIMIT_AS limit
RACF.CPUMAX	ZEROS	0270 extract		CPUTIMEMAX: RLIMIT_CPU maximum CPU limit
RACF.FILEMAX	ZEROS	0270 extract		FILEPROCMAX: Maximum number of files a user may have open or active at one time.
RACF.HOME	Blanked out	0270 extract		HOME: Specifies the HFS initial directory pathname.
RACF.MAXSTOR	ZEROS	0270 extract		MMARAREAMAX: Maximum amount of data space storage (in pages) that may be mapped in memory.
RACF.PROGRAM	Blanked out	0270 extract		PROGRAM: Unix shell program pathname.
RACF.PROMAX	ZEROS	0270 extract		PROCUSERMAX: Maximum number of processes a user may have active.

RACF.THREADS	ZEROS	0270 extract		THREADSMAX: Maximum number of PTHREADS that may be available.
WORKATTR Segment				
RACF.WAACNT	Blanked out	0260 extract		Account number
RACF.WAADDR1	Blanked out	0260 extract		Address line 1
RACF.WAADDR2	Blanked out	0260 extract		Address line 2
RACF.WAADDR3	Blanked out	0260 extract		Address line 3
RACF.WAADDR4	Blanked out	0260 extract		Address line 4
RACF.WABLDG	Blanked out	0260 extract		Building information
RACF.WADEPT	Blanked out	0260 extract		Department information
RACF.WANAME	Blanked out	0260 extract		Name information
RACF.WAROOM	Blanked out	0260 extract		Room Information

Chapter 31: INDEX

L

M

O

P

R

S

T

U

V