RACF enhanced PassTicket Support:
V1.00 APARs: RACF OA59196, SAF OA59197

Summary of Changes		
Version	**Date**	**Nature of Change**
V1.00	12/2020	Initial version: RACF APAR OA59196 & SAF APAR OA59197

1 Introduction

Support for enhanced PassTickets:
Enhanced PassTickets are authentication tokens which can be generated by authorized applications to authenticate users to other z/OS applications. Enhanced PassTickets are functionally similar to the existing PassTicket support, now referred to as legacy PassTickets, but use an updated algorithm.

Support is added to RACROUTE REQUEST=VERIFY and initACEE authentication processing to validate users with an enhanced PassTicket.

Support is added to RCVTPTGN, R_GenSec and R_Ticketserv to generate and evaluate an enhanced PassTicket.

For more details on using RACF to generate and evaluate an enhanced PassTicket, please refer to updated publication sections below.

Enhanced PassTickets Configuration:
The security administrator can create profiles in the PTKTDATA class to configure how an enhanced PassTicket is generated and evaluated. The profiles can be used to control options such as which key is used to generate and evaluate the enhanced PassTicket and its validity period.

Support for enhanced PassTickets must be enabled by activating the PTKTDATA class and defining an enhanced PassTicket key for each application before they can be generated or evaluated.

For more details on configuring enhanced PassTicket profiles, please refer to the *z/OS Security Server RACF Command Language Reference* publication section below.

Restriction: The ISPF panels and TSO helps are not updated for the new command operands with OA59196 and OA59197.

2 Planning

When installing service like this, consider the following before making changes:

- Create a backup copy of your RACF database.
- Apply the RACF enhanced PassTicket APARs to all systems sharing the RACF database.

2.1 Create a backup copy of your RACF database

Creating a backup of the RACF database is recommended whenever significant changes are being made to RACF and the RACF database.

2.2 Apply the RACF enhanced PassTicket APARs to all systems that share the RACF database

Make sure that the service is applied on all sharing systems, and that all the ++HOLD documentation has been reviewed.

2.3 RACF exit considerations

The ICHRIX01 preprocessing and ICHRIX02 postprocessing exits can alter the behavior of RACROUTE REQ=VERIFY authentication processing. When the PTKTDATA class is active and an EPTKEYLABEL value is configured for the target application, RACROUTE REQ=VERIFY and initACEE will begin evaluating a specified enhanced PassTicket.

Enhanced PassTickets is a new way to authenticate a user with RACF. Before activating the PTKTDATA class and configuring the EPTKEYLABEL keyword, the installation must ensure that any RACROUTE ICHRIX01 and ICHRIX02 exits are compatible with enhanced PassTicket processing. For example, if these exits inspect the password parameter to make processing decisions, they must take into account the new enhanced PassTicket processing.

2.4 Performance considerations

When enhanced PassTickets are configured via new keywords in the SSIGNON segment, they will be generated and evaluated by RACF APIs. Generation and evaluation of enhanced PassTickets in RACF uses ICSF HMAC APIs and keys and may have different performance characteristics than legacy PassTickets. Installations that wish to migrate from legacy PassTickets to enhanced PassTickets should evaluate the performance characteristics on a test system before implementing in production.

3 Updated RACF publications

Chapters of the following RACF publications are affected by the new function:

Publication Name	Publication Number
z/OS Security Server RACF Security Administrator's Guide	SA23-2289
z/OS Security Server RACF Command Language Reference	SA23-2292
z/OS Security Server RACF Callable Services	SA23-2293
z/OS Security Server RACF Macros and Interfaces	SA23-2288
z/OS Security Server RACF Data Areas	GA32-0885
z/OS Security Server RACF Messages and Codes	SA23-2291

In the following sections, highlighting is used to denote changed information in existing documentation. Sections, tables, messages, command keywords, etc. without highlighting contain new information.

3.1 z/OS Security Server RACF Security Administrator's Guide

This information supplements the following chapters:

- Chapter: '*Using PassTickets*'

3.1.1 Using PassTickets

This chapter is updated to add details about enhanced PassTickets. Updated sections are listed below with additions highlighted.

Introduction:

If your installation includes workstations and client machines that are operating in a client/server environment, you might want to use RACF PassTickets to provide enhanced security across a network. A PassTicket provides an alternative to the RACF password and password phrase which allows workstations and client machines to communicate with a host without using a RACF password or password phrase.

Use of a PassTicket removes the need to send RACF passwords and password phrases across the network and allows you to move the user authentication part of signing on to a host from RACF to another product or function. End users of an application can use the PassTicket to authenticate their user IDs and log on to computer systems that contain RACF.

This chapter describes the PassTicket and how to set up the PassTicket environment. It includes

information about:

- Activating the PTKTDATA class
- Defining profiles in the PTKTDATA class
- How RACF processes the PassTicket
- Enabling the use of PassTickets
- Auditing the use of PassTickets

For information about the programming that is needed for an application to generate a PassTicket, see *z/OS Security Server RACF System Programmer's Guide*.

The RACF PassTicket

The RACF PassTicket is a one-time-only[2] password that is generated by a requesting product or function. It is an alternative to the RACF password and password phrase that removes the need to send RACF passwords and password phrases across the network in clear text. It makes it possible to move the authentication of a mainframe application user ID from RACF to another authorized function executing on the host system or to the workstation local area network (LAN) environment.

Legacy PassTickets and enhanced PassTickets

RACF PassTickets can be configured with two different algorithms:

- The legacy PassTicket algorithm

- The enhanced PassTicket algorithm

The legacy PassTicket algorithm is the original PassTicket implementation and the enhanced PassTicket algorithm is an updated version of the PassTicket algorithm. Enhanced PassTickets function much in the same way as legacy PassTickets but contain a number of usability and security enhancements.

RACF supports generation and evaluation of PassTickets with either the legacy PassTicket algorithm or the enhanced PassTicket algorithm per application based on PTKTDATA class profile configuration. Both legacy PassTickets and enhanced PassTickets can be generated by appropriately authorized applications to authenticate z/OS users to other z/OS applications. In either case, the generated PassTicket value is supplied to z/OS applications as an 8-character value in the password field. Both legacy PassTickets and enhanced PassTickets are generated and evaluated using a shared secret key. Both legacy PassTickets and enhanced PassTickets may be generated on z/OS or on other platforms and both PassTicket generation algorithms are documented in *z/OS Security Server RACF Macros and Interfaces*.

While the legacy PassTicket algorithm uses a secret 64-bit DES key, the enhanced PassTicket algorithm uses a 256-2048 bit HMAC secret key. While legacy PassTicket key material may be optionally masked in the RACF database, enhanced PassTickets keys must be stored encrypted in ICSF. For more information on PassTicket keys, see “Protecting PassTicket keys”.

While the legacy PassTickets character set uses only uppercase characters A-Z and digits 0-9, enhanced PassTickets can optionally use an expanded character set which also includes the lowercase characters a-z and two special symbols. By supporting a much larger set of possible valid values, enhanced PassTickets have more variability and are therefore more secure against certain attack vectors than legacy PassTickets. The enhanced PassTicket character set can be configured in the PTKTDATA class profile with the TYPE(MIXED) or TYPE(UPPER) keywords in the SSIGNON segment.

While legacy PassTickets are valid 10 minutes before or after they are generated, enhanced PassTickets provides a configurable validity period which can be set between 1 second and 10 minutes. By configuring a shorter validity period, installations can limit the amount of time that enhanced PassTickets are valid. The enhanced PassTicket validity period can be configured in the PTKTDATA class profile with the TIMEOUT keyword in the SSIGNON segment.

For more information on configuring legacy and enhanced PassTickets, refer to the SSIGNON segment for the RDEFINE and RALTER commands in the *z/OS Security Server RACF Command Language Reference*.

Note: IBM strongly recommends using the enhanced PassTicket algorithm as it provides the same capabilities as the legacy PassTicket algorithm but also provides increased security.

2 Because it can only ~~gives one user access to~~ be used to authenticate to a specific application for a limited time interval, a RACF PassTicket is resistant to reuse. For most applications, once a particular PassTicket is used, the same user cannot use it again for the same application ~~during the same 10-minute interval~~. For performance reasons, RACF uses main memory for this storage. If an application can run on more than one computer with individual memory at the same time, this level of reuse protection might not be available.

Activating the PTKTDATA class

Before you can use PassTickets, you must activate the PTKTDATA class. The PTKTDATA class is the class to which all profiles that contain PassTicket information are defined. To activate the class and the function, enter:

```
SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
```

After you activate the PTKTDATA class, you can define the necessary profiles.
Note: After you define or change the profiles, you need to refresh the class by entering:

```
SETROPTS RACLIST (PTKTDATA) REFRESH.
```

Defining profiles in the PTKTDATA class

For each application that users can gain access to with the PassTicket, you must create at least one profile in the PTKTDATA class. The profile associates a PassTicket key with a particular application on a particular system. The profiles can be created so they apply to:

- All users
- Users who belong to a specific RACF group
- A specific RACF user, when connected to a specific RACF group
- A specific RACF user

To define the profile, use the RDEFINE command:

```
RDEFINE PTKTDATA profile-name SSIGNON(key-description)
UACC(NONE)
```

where:

PTKTDATA
specifies the PassTicket key class.

profile-name
is the name of the profile (see “Determining PTKTDATA profile names”).
For the PTKTDATA class, the profile must be a discrete profile. Because each application must be uniquely defined, you cannot specify a generic profile in the PTKTDATA class. If you specify a generic profile, it is ignored during PassTicket processing for the application, and PassTickets cannot be used to authenticate users for that application.

key-description

defines the PassTicket keys and related configuration settings.

For legacy PassTickets:

- A subset of these keywords specify the method RACF is to use to protect the legacy PassTicket key in the RACF database on the host. You can specify either masking or encryption for the method (see "Protecting legacy PassTicket keys").
- Legacy PassTicket keys are 64-bit Data Encryption Standard (DES) keys. With DES, eight of the 64 bits are reserved for use as parity bits, so those eight bits are not part of the 56-bit key. In hexadecimal notation, the DES parity bits are: X'0101 0101 0101 0101'. Any two 64-bit keys are equivalent DES keys if their only difference is in one or more of these parity bits.

For enhanced PassTickets:

- A subset of these keywords identify the enhanced PassTicket keys and related configuration settings to be used to generate and evaluate an enhanced PassTicket. Enhanced PassTicket keys are 256-2048 bit HMAC keys.

Determining PTKTDATA profile names

(This section is unchanged and is not included in this document.)

Protecting PassTicket keys

PassTicket keys are sensitive and must be protected from unauthorized disclosure. Entities with access to the configured application PassTicket keys for can generate valid PassTickets for that application.

When you define legacy PassTicket keys, RACF either masks or encrypts each key. If the system has ICSF installed and available, you can store PassTicket keys in ICSF for added protection. When you define enhanced PassTicket keys they must be stored in ICSF. For more information, see "Storing legacy PassTicket Keys Masked in RACF" and "Storing PassTicket keys encrypted in ICSF".

Storing legacy PassTicket keys masked in RACF

Legacy PassTicket keys can be stored encrypted in ICSF or masked in RACF with a proprietary masking algorithm when you define or alter it.
The masking algorithm is designed to provide protection against casual viewing of the PassTicket masked keys. The algorithm is not a cryptographic algorithm and cannot provide the level of security for the PassTicket keys that the use of cryptography can provide.

Note: IBM **STRONGLY** recommends that masked PassTicket keys are not used outside of a test environment.

To mask a legacy PassTicket key when you define or alter it, use the SSIGNON operand and KEYMASKED value with the RDEFINE or RALTER command.

You can use the ENCRYPTKEY keyword to encrypt a masked key and move it into the CKDS. See “Converting legacy PassTicket masked keys to encrypted keys”.

Note: To prevent unauthorized users from looking at the PassTicket keys that are stored in the RACF database, make sure the universal access authority (UACC) of the RACF database is NONE. This prevents unauthorized users from listing or copying the RACF data set that contains these sensitive keys.

Storing PassTicket keys encrypted in ICSF

ICSF can be used to store PassTicket keys in the CKDS, encrypted under the master key. Using ICSF ensures the maximum possible security for the PassTicket keys.

For legacy PassTickets there are two options for defining the key to ICSF:

- Use the SSIGNON operand and the KEYLABEL keyword to identify the CKDS key label to use for the particular PTKTDATA profile being added or altered. The key must refer to a DES key with a type of DATA and a length of 8 bytes. KEYLABEL is the recommended option as it allows for secure key entry and the use of your own naming convention for keys. You are responsible for adding the appropriate key to the CKDS, with the specified label, before it is used in a PassTicket operation.
- Use the SSIGNON operand and KEYENCRYPTED keyword to enter the key value to use for the particular PTKTDATA profile being added or altered. RACF will generate a key label value in the form IRR.SSIGNON.sysname.mmddyyyy.hhmmss.nnnnnn and add the key to the CKDS. The key label name is not user configurable.

For enhanced PassTickets the key must be defined in ICSF:

- Use the SSIGNON operand and the EPTKEYLABEL keyword to identify the CKDS key label to use for the particular PTKTDATA profile being added or altered.
- The key label must refer to an ICSF HMAC key with a key algorithm of HMAC, a key type of MAC and the key usage fields must indicate GENERATE. The supported HMAC key size range is from 32 to 256 bytes. The recommended minimum key size is 64 bytes.
- You are responsible for adding the appropriate key to the CKDS, with the specified label, before it is used in a PassTicket operation.
- The RACF enhanced PassTicket support uses ICSF HMAC keys which require that the ICSF CKDS is defined in either the variable length record format or common record format (KDSR). For more information on ICSF CKDS formats please refer to Chapter 1 of the z/OS: Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide (SC14-7507-09)

The RLIST command displays the key label used for an encrypted key.

When the RACF database is shared, the use of ICSF is simplest when the CKDS and RACF database are shared across a common set of systems. RACF always uses the local system's CKDS when generating or evaluating a PassTicket. If the PassTicket is generated

on one system, and then evaluated on a different system, the evaluation will fail if RACF is unable to retrieve the key from the local CKDS. If the ICSF CKDS is not shared across systems which share the RACF database, ICSF services must be used to export the key label from the system on which the PassTicket key was defined. The key must then be imported to the ICSF CKDS of all other systems which share the RACF database. The ICSF CSNDSYX and CSNDSYI services can be used to export and import PassTicket keys from the ICSF CKDS. The ICSF CSNBKEX and CSNBKIM services can also be used to export and import PassTicket keys from the ICSF CKDS. There is a similar consideration if you are using the remote sharing facility to propagate commands that update PTKTDATA class profiles. If the target of the propagation is a multisystem node, the CKDS in use on the remote node's MAIN system will be the only CKDS updated with the new PassTicket key.

Note that older versions of RACF might have stored a legacy PassTicket key token in the profile instead of a key label. The creation of a legacy PassTicket key token is also possible if the user entering the RACF command lacks authorization to the CSFKEYS profile protecting the key label name, or to the CSFKRC or CSFKRW service. Like a normal KEYENCRYPTED key, a key token is also encrypted under the CKDS master key, but it is stored in RACF instead of the CKDS, and thus there is no key label. RACF updates the key token when a master key change is detected. RACF only updates a key token when it is used in a PassTicket operation. If the master key is changed twice between use of a specific key token, the key token is rendered unusable. When the RACF database is shared, and the CKDS is not shared across the generating and evaluating systems, the CKDS master keys must be the same.

The RLIST command will indicate the presence of a legacy PassTicket key token. You can use the ENCRYPTKEY keyword of the RALTER command to move this token into the CKDS using a RACF-generated key label name. See “Converting legacy PassTicket masked keys to encrypted keys”.

Important: RACF does not delete keys from the CKDS. Before deleting or changing an encrypted key, take note of the current key label value so that it can be deleted from the CKDS, using ICSF interfaces.

Converting masked legacy PassTicket keys to encrypted keys

(This section is unchanged and is not included in this document.)

Authorization requirements for managing PassTicket keys

If SSIGNON(KEYENCRYPTED) is specified for legacy PassTickets on an RDEFINE PTKTDATA or RALTER PTKTDATA command, access to the following ICSF services needs to be defined:

- CSFCKI
- CSFKRC
- CSFKRW

• CSFKRD

If SSIGNON(KEYENCRYPTED) or SSIGNON(ENCRYPTKEY) is specified for legacy PassTickets on an RDEFINE PTKTDATA or RALTER PTKTDATA command, the user requires READ access to keys in the form of IRR.SSIGNON.sysname.* using profiles in the CSFKEYS class. Sysname is the name of the system where the keyword was specified. For information on protecting ICSF resources, see z/OS Cryptographic Services ICSF Administrator's Guide.

If RACF field level access checking is enabled, the user issuing the RDEFINE or RALTER command specifying the SSIGNON segment must have UPDATE access to the appropriate fields. For legacy PassTickets, UPDATE access is required to the PTKTDATA.SSIGNON.SSKEY resource in the FIELD class. Note that the ENCRYPTKEY, KEYENCRYPTED, KEYMASKED, KEYLABEL and NOLEGACYKEY keywords all store data into the SSKEY field, and thus they are all protected by the same resource profile. For details on field level access control, see “Field-level access checking” on page 196.

Examples of defining PTKTDATA class profiles
Suppose you want to define a profile for TSO in the PTKTDATA class. The system programmer has told you that a VTAM generic resource name for TSO is not being used, and that the SMF identifier of the system on which the TSO application is to run is R001. The universal access is to be the default for the PTKTDATA class (NONE).

For legacy PassTickets:

You want to encrypt the legacy PassTicket key and specify a key value of X'E001193519561977'.

To define the profile for legacy PassTickets, enter:

```
RDEFINE PTKTDATA TSOR001
SSIGNON(KEYENCRYPTED(E001193519561977))
```

For enhanced PassTickets:

You want to set the enhanced PassTicket ICSF key label to the value of ‘TSOR001.EPTKEY01' and the character set type to MIXED.

To define the profile for enhanced PassTickets, enter:

```
RDEFINE PTKTDATA TSOR001 SSIGNON(EPTKEYLABEL
(TSOR001.EPTKEY01) TYPE(MIXED))
```

When the profile definitions are complete
After you define the PTKTDATA class profile for the application program that is to generate a PassTicket, the program can be installed and used.

RACF provides several services by which a z/OS application can request the generation of

a PassTicket. For details on the R_GenSec and R_ticketserv services, see *z/OS Security Server Callable Server RACF Callable Services*. For details on the RCVTPTGN service, see *z/OS Security Server RACF Macros and Interfaces*.

~~For information on how to code an application program to generate a PassTicket, see *z/OS Security Server RACF Callable Services*.~~

How RACF processes the PassTicket

To validate ~~a password~~ or PassTicket, RACF does the following:

1. Determines whether a profile has been defined for the application in the PTKTDATA class.
   - If a profile has not been defined and the value does not match the user's password, the user receives a message from the application indicating that the password is not valid.[3]
   - If the application is defined in the PTKTDATA class, processing continues.
2. Evaluates the value entered in the password field. The evaluation determines whether:
   - The value is a PassTicket consistent with this user ID, application, and time range.
   - For enhanced PassTickets, the PassTicket value also must have been generated with the same character set type (UPPER or MIXED) as the evaluator.
   - ~~It has been used previously on this computer system for this user ID, application, and time range.~~

   Time Considerations:
   - A PassTicket is considered to be within the valid time range when the time of generation, with respect to the clock on the generating computer, is within the acceptable validity period of the time of evaluation, with respect to the clock on the evaluating computer. For legacy PassTickets the acceptable validity period is 10 minutes before or after the generation time. For enhanced PassTickets the acceptable validity period is configurable between 1 second and 10 minutes before or after the generation time.
   - Be sure that your MVS system and the evaluating computer use clock values that are within that time range. RACF uses the value stored for coordinated universal time (UTC), formerly called Greenwich mean time (GMT), in the algorithms that process PassTickets.
   - One way to ensure that reasonably synchronized values are used is to set UTC in the GMT value of the MVS time of day (TOD) clock and to set a similar value in each of the other systems with which RACF shares PassTicket information. You can still use the MVS local time for local timestamp information, and resetting the local time does not affect the GMT value kept in the TOD clock.
   - **Important:** Before setting the TOD clock's GMT value to UTC, make sure that the subsystems and applications you use are not affected.
   - To be sure the MVS system clock is set properly, the system console operator should issue:
     ```
     DISPLAY T
     ```
   - The system displays the time with information similar to the following:
     ```
     IEE136I LOCAL: TIME=14.06.18 DATE=1997.309
     ```



```
GMT: TIME=19.06.18 DATE=1997.309
```

- **Important:** If the MVS DISPLAY T command indicates that your system clock is not set correctly for GMT, you need to analyze the consequences of resetting the clock. It is possible that other programs that execute on the system have been adjusted to tolerate an incorrect GMT setting. You might need to readjust those programs before resetting the system clock.
- See z/OS MVS Initialization and Tuning Reference and *z/OS MVS System Commands* for more information on setting clocks. See *z/OS Security Server RACF Macros and Interfaces* for more information on the algorithms.

~~Determines whether the value is a valid PassTicket.~~
- ~~If the PassTicket is valid, RACF gives the user access to the desired application.~~
- ~~If the value is not valid, the host application sends a message to the user indicating that the password is not valid (assuming the value also did not evaluate correctly as the user's RACF password).~~

3. Determines if the PassTicket has been used previously on this computer system for this user ID, application, and time range.
   - If the value was used before, and if PassTicket replay protection has not been bypassed, the user receives a message from the application[4] indicating that the password is not valid.
   - If the value was not used before, or PassTicket replay protection has been bypassed, the PassTicket is considered valid and processing continues.
4. Allows or denies access to the target application.
   - If the PassTicket is valid, RACF gives the user access to the desired application.
   - If the value is not valid, the host application sends a message to the user indicating that the password is not valid (assuming the value also did not evaluate correctly as the user's RACF password).

Note: If the PassTicket key is stored in ICSF with the KEYENCRYPTED, ENCRYPTKEY, KEYLABEL or EPTKEYLABEL keywords, ICSF must be active when RACF tries to authenticate the PassTicket. If it is not active, RACF cannot validate the PassTicket. The resulting message indicates that the logon attempt failed.

3 - RACF sends a message to the SYSLOG and to the security console. The application rejects the logon request the same way it rejects an incorrect password. The text of the message the user receives depends on the application.
4 - RACF sends a message to the SYSLOG and to the security console. The application rejects the logon request the same way it rejects an incorrect password. The text of the message the user receives depends on the application.

Bypassing PassTicket replay protection
You might use the option to bypass PassTicket replay protection when the threat of PassTicket replay is not a security concern, such as in the following cases:

- Applications which save the password and use it for multiple logons on a user's behalf. (Note that the multiple logons must occur within the ~~10 minute~~ PassTicket validity window for this type of application to work with PassTickets.)
- Trusted registry domains that exchange PassTickets as a method of establishing trust.
- Applications that request PassTickets for a particular USERID/APPLID combination more than once during a one-second time interval.

The option to bypass PassTicket replay protection allows ~~the plus-or-minus-10-minute~~ PassTicket replay protection to be bypassed for selected applications or combinations of selected applications, users, or groups.

Note:
1. The option to bypass PassTicket replay protection should only be used in secure environments where access to generated PassTickets is limited within a secure or internal network.

Bypassing legacy PassTicket replay protection

You indicate that replay protection is to be bypassed for legacy PassTickets for a particular application by adding the text string `NO REPLAY PROTECTION` to the APPLDATA field of the PTKTDATA profile for that application. You must separate each word in the string with a single blank space, alphanumeric character, or keyboard symbol.
The `NO REPLAY PROTECTION` text string will always be translated to upper case by the RALTER or RDEFINE commands.
The `NO REPLAY PROTECTION` text string can appear anywhere within the APPLDATA field, allowing for the existence of other information already in the field, or for new information that might be added in the future.
The following are examples of commands that will cause legacy PassTicket replay protection to be bypassed.

Examples:

```
RALTER PTKTDATA profile-name APPLDATA('NO REPLAY PROTECTION')
RDEFINE PTKTDATA profile-name APPLDATA('NO REPLAY PROTECTION')
RDEFINE PTKTDATA profile-name
APPLDATA('FOR THIS APPLICATION NO REPLAY PROTECTION IS IN
EFFECT')
```

Note:
1. Other than the APPLDATA (application data) field of the application profile containing the text string, `NO REPLAY PROTECTION`, there is no other external indication that replay protection is bypassed.
2. The APPLDATA field replay protection only applies to legacy PassTickets and does not affect the replay behavior of enhanced PassTickets.

Bypassing enhanced PassTicket replay protection

(This section is new and not highlighted to improve readability.)

You indicate that replay protection is to be bypassed for enhanced PassTickets for a particular application by setting the SSIGNON segment keyword REPLAY(YES) in the PTKTDATA profile for that application.

Example:

```
RALTER PTKTDATA profile-name SSIGNON(REPLAY(YES))
```

Note:
1. The SSIGNON segment REPLAY keyword replay protection only applies to enhanced PassTickets and does not affect the replay behavior of legacy PassTickets.

Enabling the use of PassTickets

(This section is unchanged and is not included in this document.)

Verifying the PassTicket environment

(This section is unchanged and is not included in this document.)

Migrating from legacy PassTickets to enhanced PassTickets

(This section is new and not highlighted to improve readability.)

Enhanced PassTickets provide the same capabilities as legacy PassTickets but with improved security. Migration from legacy PassTickets to enhanced PassTickets will take planning and effort. RACF allows for an installation to have both legacy PassTickets and enhanced PassTickets configured for the same application in the same PTKTDATA class profile.
When a PTKTDATA class profile contains both a legacy PassTicket key and enhanced PassTicket key:

- PassTicket generation requests though RACF services will result in an enhanced PassTicket
- PassTicket evaluation requests though RACF will evaluate the PassTicket with both the legacy PassTicket algorithm and enhanced PassTicket algorithm.

Installations that wish to migrate from legacy PassTickets to enhanced PassTickets can use the following steps as a guide:

1. **Determine the desired enhanced PassTicket character type:**
   This setting determines the possible characters that represent the enhanced PassTicket. TYPE(UPPER) will only use uppercase A-Z and digits 0-9. TYPE(MIXED) also includes lowercase a-z and the symbols underscore “_” and dash “-“.
   In general, installations that have RACF mixed case password support enabled should

use TYPE(MIXED) and installations that do not have RACF mixed case password support enabled should use TYPE(UPPER). The default value is TYPE(MIXED). The TYPE setting of the enhanced PassTicket evaluator must match the TYPE setting of the PassTicket generator.

2. **Determine the desired enhanced PassTicket REPLAY allowed setting:**
   In some cases an installation may require PassTickets to be able to be replayed for a particular application. To allow replay of enhanced PassTickets for the application set REPLAY(YES) in the PTKTDATA class application profile. The default value is REPLAY(NO). See **"***Bypassing PassTicket replay protection***"** for more information on PassTicket replay considerations.
3. **Define the enhanced PassTicket HMAC key in the ICSF CKDS:**
   The key must be defined with a key label that refers to an ICSF HMAC key with a key algorithm of HMAC, a key type of MAC and the key usage fields must indicate GENERATE. The supported HMAC key size range is from 32 to 256 bytes. The recommended minimum key size is 64 bytes.
   Refer to "Cryptographic Services - Integrated Cryptographic Service Facility - Application Programmer's Guide" for details on managing ICSF keys.
4. **Add the enhanced PassTicket Key Label to the PTKTDATA class profile:**
   Update the PTKTDATA class application profile to add the key label of the HMAC key in ICSF using the SSIGNON segment EPTKEYLABEL keyword.
   When the enhanced PassTicket key label is added to the PTKTDATA class application profile RACF will begin to evaluate user specified passwords with the enhanced PassTicket algorithm.
   Systems which do not share the same RACF database and/or ICSF datasets will need to provision the same enhanced PassTicket HMAC secret key in order to generate and evaluate compatible enhanced PassTickets.
5. **Update applications to generate enhanced PassTickets:**
   - Applications that generate PassTickets on-platform using RACF services will begin to generate enhanced PassTickets when an enhanced PassTicket key is added to the PTKTDATA class application profile.
   - Applications that generate PassTickets outside of RACF with the PassTicket algorithm need to be updated to support the enhanced PassTicket algorithm. Once the application supports generation of enhanced PassTickets, it must be configured with the same HMAC secret key and character set as RACF for evaluation to be successful. Refer to the "z*/OS Security Server RACF Macros and Interfaces"* for details on implementing the enhanced PassTicket algorithm in your own application.
6. **Test enhanced PassTicket generation and evaluation:**
   Use the application to generate an enhanced PassTicket and attempt to use it to authenticate to the configured target application.
7. **Remove the legacy PassTicket key:**
   Once it has been confirmed that enhanced PassTicket evaluation is successful and all applications are no longer generating legacy PassTickets for the target application the legacy PassTicket key should be removed from the PTKTDATA class profile with the NOLEGACYKEY keyword.

Preventing errors

The following checklist describes the errors that might cause a PassTicket to fail. To prevent these errors from occurring:

1. Read the list before you use the PassTicket.
2. Review your process to ensure that you have entered all of the information correctly.
3. Verify the information by using the procedures described in "*Verifying the PassTicket environment*".

Use this checklist to prevent or correct errors:

- The PTKTDATA class is activated.
- You issued the SETROPTS RACLIST(PTKTDATA) command.
- You issued the SETROPTS RACLIST(PTKTDATA) REFRESH command after defining the profile.
- A PTKTDATA class profile exists for the application.
- The application name used by RACROUTE REQUEST=VERIFY during evaluation matches the name in the PTKTDATA profile that you expect to be used. The SMF Type 80 event code 1 record includes relocate section 443, which contains the application name that was used in the evaluation process. If a z/OS application is using the R_Gensec, R_Ticketserv, or RCVTPTGN service to generate or evaluate a PassTicket, and these requests are being logged (see the following topic), SMF Type 80 event code 81 (Evaluate) and event code 82 (Generate) will contain the application name in relocate section 67.
- You issued the RDEFINE command correctly.
- A protected user ID may not be used for PassTicket authentication.
- The PassTicket key must be the same on the system which generated the PassTicket and the system on which the PassTicket is being evaluated.
- The application name used to generate the PassTicket must match the application name used to log on with the PassTicket. Ensure the application name is not altered by a user exit during logon.
- PassTickets can be generated with the legacy PassTicket algorithm or enhanced PassTicket algorithm. Enhanced PassTickets can use either a character set type of UPPER or MIXED. The PassTicket must be evaluated with the same algorithm and character set type as it was generated.

Even if you have followed the proper procedures, it is still possible to receive a message stating that a password is incorrect and be denied access to the application. This can occur if:

- PassTicket replay protection is not being bypassed, and the PassTicket was used previously for this user, application, and time range.
- In this case, RACF generates an SMF record that logs an attempt to replay a PassTicket.
- The GMT clock on the evaluating computer is outside the valid time range for the PassTicket.

  This can be caused by one of the following:
  - The GMT clock on the generating computer and the clock on the evaluating computer are not reasonably synchronized.
  - The PassTicket was not used within approximately 10 minutes of being

generated.
    - The system clock on the evaluating computer might not be set correctly in relation to GMT. See the information about time considerations in "*How RACF processes the PassTicket*".
- An encrypted key is being used, but the key is not preset in the local ICSF CKDS.

PassTicket diagnostic reason codes are provided when generation or evaluation of a PassTicket fails in the following locations:

- SMF Type 80 records
    - The event codes and relocate sections documented above (as containing the application name used) also contain failure return and reason codes
- Service return and reason codes.
    - The services used to generate and evaluate PassTickets (also listed above) can provide useful diagnostic information. Note that for R_Gensec and R_Ticketserv, the application must have requested the additional diagnostics by using the 'extended' versions of the functions. Check if the application provides a trace log or other diagnostic medium containing the return and reason codes from these services.

Auditing the use of PassTickets

Generation and evaluation of PassTickets can be audited. The SETR LOGOPTIONS settings of the PTKTDATA class, as well as the AUDIT and GLOBALAUDIT setting of the PTKTDATA profiles which contain PassTicket keys can be used to determine how the use of PassTickets is audited.

SMF type 80, event code 82 (PassTicket generate) records are created in the following circumstances:

- The RCVTPTGN service is used to generate a PassTicket.
- The R_ticketserv or R_GenSec service is used to generate a PassTicket.
- The Java service, described in *z/OS Security Server RACF Macros and Interfaces* is used to generate a PassTicket.

SMF type 80, event code 81 (PassTicket evaluate) records are created in the following circumstances:

- The R_ticketserv or R_GenSec service is used to evaluate a PassTicket.
- The Java service, described in *z/OS Security Server RACF Macros and Interfaces* is used to evaluate a PassTicket.

When a user provides a PassTicket to a standard z/OS authentication service, logging is performed for the PassTicket evaluation in the SMF type 80 event code 1 record created to record the logon event. This SMF record indicates when a user authenticates with PassTicket and whether the PassTicket was evaluated with the legacy PassTicket algorithm or enhanced PassTicket algorithm.

For more details on PassTicket audit records please refer to *z/OS Security Server RACF Macros and Interfaces*.

3.2 z/OS Security Server RACF Command Language Reference

This information supplements the following chapters and sections:

- Chapter: '*RACF Command Syntax'*
  - *Section:* RDEFINE
  - *Section:* RALTER
  - *Section:* RLIST

3.2.1 RDEFINE

The base segment APPLDATA keyword description is updated to add details for enhanced PassTickets.

Parameters

…

APPLDATA('application-data')

…

- For the PTKTDATA class, the application data field can be used to control the replay protection function of legacy PassTicket support. This setting applies only to legacy PassTickets and does not control the replay behavior of enhanced PassTickets.
  - PassTicket replay protection prevents the use of user IDs to be shared among multiple users. However, in some events it is desirable to bypass this replay protection function.
  - Specifying `no replay protection` in the application data field indicates that replay protection is to be bypassed. For example, the following command would successfully result in replay protection being bypassed.

    ```
    RDEFINE PTKTDATA profile-name
    APPLDATA('NO REPLAY PROTECTION')
    ```

    Note the following:

    - There *must* be a single space between the words no and replay, and between replay and protection. Lack of spaces, or additional spaces or characters, will make the command ineffective. For example, entering the following command would not result in replay protection being bypassed.

      ```
      RDEFINE PTKTDATA profile-name
      APPLDATA('NOREPLAY PROTECTION')
      ```

  - The text string no replay protection will always be translated to uppercase.
  - The text string no replay protection can appear anywhere in the APPLDATA field.
  - See *z/OS Security Server RACF Security Administrator's Guide* for more information on the PassTicket function.

…

The SSIGNON segment is updated to add new fields for enhanced PassTickets.

Syntax

[SSIGNON (
[KEYMASKED(legacy-passticket-key-value)
| KEYENCRYPTED(legacy-passticket-key-value)
| KEYLABEL(legacy-passticket-label-value)]
[EPTKEYLABEL(enhanced-passticket-label-value)]
[TYPE(*UPPER* | ***MIXED*)]**
[TIMEOUT(timeout-seconds)]
[REPLAY(*YES* | ***NO*)]**
)]

SSIGNON

Defines PassTicket keys and associated configuration settings.

RACF PassTickets can be configured with two different algorithms:

- The legacy PassTicket algorithm
- The enhanced PassTicket algorithm

The legacy PassTicket algorithm is the original PassTicket implementation and uses a DES secret key. The enhanced PassTicket algorithm is an updated version of the PassTicket algorithm and uses an HMAC secret key. RACF supports generation and evaluation of PassTickets with either the legacy PassTicket algorithm or the enhanced PassTicket algorithm based on the SSIGNON segment keywords.

The KEYMASKED, KEYENCRYPTED, ENCRYPTKEY and KEYLABEL keywords control the key to be used for the generation and evaluation of legacy PassTickets. These keywords indicate the method you want to use to protect the legacy PassTicket key value within the RACF database. You can mask or encrypt the key. The key-value represents a 64-bit (8-byte) key that must be represented as 16 hexadecimal characters. The valid characters are 0 - 9 and A - F.

The EPTKEYLABEL, TYPE, TIMEOUT and REPLAY keywords control the key and settings to be used for the generation and evaluation of enhanced PassTickets.

Note:

1. Before defining PassTicket keys, please read and understand the PassTicket documentation in the z/OS Security Server RACF Security Administrator's Guide, specifically, the topic Protecting PassTicket keys. That documentation contains important information on setup and authorization issues, especially pertaining to the use of ICSF with encrypted keys.

2. As with RACF passwords, the database unload facility does not unload application keys or PassTicket keys. It will, however, indicate the method of protection of the key, and if the key is encrypted, the key label name.

3. The RLIST command does not list the value of the application key or the PassTicket key. Therefore, when you define the keys, you should note the value and keep it in a secure place. Note that RLIST will, however, indicate the method of protection of the key, and if the key is encrypted, the key label name.

4. The KEYMASKED, KEYENCRYPTED, ENCRYPTKEY and KEYLABEL legacy PassTicket keywords all work against the same field in the RACF database. Use of any of these RALTER keywords replaces the previous legacy PassTicket key (or its label) in the RACF database.

KEYMASKED(legacy-passticket-key-value)

Specifies that you want to mask the legacy PassTicket key value using the masking algorithm.

Note:

1. IBM **STRONGLY** recommends that masked PassTicket keys are not used outside of a test environment.

2. You can specify this operand only once for each application key.

3. If you mask a key, you cannot encrypt it. These are mutually exclusive.

KEYENCRYPTED(legacy-passticket-key-value)

Specifies that you want to encrypt the legacy PassTicket key value.

Note:

1. Before using the KEYENCRYPTED keyword, please read and understand the documentation describing Encrypting the PassTicket key in the z/OS Security Server RACF Security Administrator's Guide.

2. You can specify this operand only once for each application key.

3. If you encrypt a key, you cannot mask it. These are mutually exclusive.

4. ICSF must be installed and active on the system.

You can use the RLIST command to verify that the key is protected.

KEYLABEL(legacy-passticket-label-value)

Specifies the name of an ICSF key label to be used when generating or evaluating a legacy PassTicket.

ICSF must be installed and active, and the key must be defined in the ICSF CKDS at the time of use. However, this is not checked when the KEYLABEL keyword is specified.

When using KEYLABEL, RACF does not make any calls to ICSF. The key label is saved in the RACF database, and it is up to the installation to ensure that the key is added to the ICSF CKDS before any PassTicket operations occur which need it. The key must refer to a DES key with a type of DATA and a length of 8 bytes.

Note:

The KEYLABEL operand cannot be used to override the key label generated by RACF when KEYENCRYPTED or ENCRYPTKEY is specified.

EPTKEYLABEL(enhanced-passticket-*label-value*)

Specifies the name of an ICSF key label to be used when generating or evaluating an enhanced PassTicket.

ICSF must be installed and active, and the key must be defined in the ICSF CKDS at the time of use. However, this is not checked when the EPTKEYLABEL keyword is specified.

When using EPTKEYLABEL, RACF does not make any calls to ICSF. The key label is saved in the RACF database, and it is up to the installation to ensure that the key is added to the ICSF CKDS before any enhanced PassTicket operations occur which need it.

The key label must refer to an ICSF HMAC key with a key algorithm of HMAC, a key type of MAC and the key usage fields must indicate GENERATE. The supported HMAC key size range is from 32 to 256 bytes. The recommended minimum key size is 64 bytes.

The RACF enhanced PassTicket support uses ICSF HMAC keys which require that the ICSF CKDS is defined in either the variable length record format or common record format (KDSR). For more information on ICSF CKDS formats please refer to Chapter 1 of the z/OS: Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide (SC14-7507-09).

The label name cannot exceed 64 characters. The first character must be an alphabetic character or a national character (#, @, or $). Subsequent characters can be a period character (.) or any alphanumeric or national character.

TYPE(*UPPER* | *MIXED*)

Specifies the character set to use for generating and evaluating an enhanced PassTicket.

The type must be one of the following values:

UPPER – The enhanced PassTicket will be generated and evaluated with only uppercase characters A - Z and digits 0 - 9.

MIXED – The enhanced PassTicket will be generated and evaluated with uppercase characters A-Z, lowercase characters a-z, digits 0-9 and the symbols dash (-) and underscore (_).

Using type MIXED is recommended as it provides a larger set of possible PassTicket values and is therefore more secure. Type UPPER may be required when an application does not yet support mixed case passwords.

The default value is MIXED.

TIMEOUT(*timeout-seconds*)

Specifies the number of seconds that the enhanced PassTicket is active.

The value of timeout-seconds can be between 1 and 600 seconds (10 minutes).

The default value is 60 seconds.

REPLAY(*YES* | *NO*)

Specifies whether an enhanced PassTicket is allowed to be replayed within the TIMEOUT value.

The default value is NO.

This setting only applies to enhanced PassTickets and does not apply to legacy PassTickets.

The replay protection setting in the APPLDATA field only applies to legacy PassTickets and does not apply to enhanced PassTickets.

3.2.2 RALTER

The base segment APPLDATA keyword description is updated to add details for enhanced PassTickets.

Parameters

…

APPLDATA('application-data')

…

- For the PTKTDATA class, the application data field can be used to control the replay protection function of legacy PassTicket support. This setting applies only to legacy PassTickets and does not control the replay behavior of enhanced PassTickets.
  - PassTicket replay protection prevents the use of user IDs to be shared among multiple users. However, in some events it is desirable to bypass this replay protection function.
  - Specifying `no replay protection` in the application data field indicates that replay protection is to be bypassed. For example, the following command would successfully result in replay protection being bypassed.


```
RDEFINE PTKTDATA profile-name
APPLDATA('NO REPLAY PROTECTION')
```

Note the following:

- There *must* be a single space between the words no and replay, and between replay and protection. Lack of spaces, or additional spaces or characters, will make the command ineffective. For example, entering the following command would not result in replay protection being bypassed.

```
RDEFINE PTKTDATA profile-name
APPLDATA('NOREPLAY PROTECTION')
```

- The text string no replay protection will always be translated to uppercase.
- The text string no replay protection can appear anywhere in the APPLDATA field.
- See *z/OS Security Server RACF Security Administrator's Guide* for more information on the PassTicket function.

…

The SSIGNON segment is updated to add new fields for enhanced PassTickets.

Syntax

[SSIGNON (
[KEYMASKED(legacy-passticket-key-value)
| KEYENCRYPTED(legacy-passticket-key-value)
| ENCRYPTKEY
| KEYLABEL(legacy-passticket-label-value)
| NOLEGACYKEY]
[EPTKEYLABEL(enhanced-passticket-label-value) | NOEPTKEYLABEL]
[TYPE(*UPPER* | *MIXED*) | NOTYPE]
[TIMEOUT(timeout-seconds) | NOTIMEOUT]
[REPLAY(*YES* | *NO*)]
)
| NOSSIGNON]

SSIGNON | NOSSIGNON

…

SSIGNON

Defines PassTicket keys and associated configuration settings.

RACF PassTickets can be configured with two different algorithms:

- The legacy PassTicket algorithm
- The enhanced PassTicket algorithm

The legacy PassTicket algorithm is the original PassTicket implementation and uses a DES secret key. The enhanced PassTicket algorithm is an updated version of the PassTicket algorithm and uses an HMAC secret key. RACF supports generation and evaluation of PassTickets with either the legacy PassTicket algorithm or the enhanced PassTicket algorithm based on the SSIGNON segment keywords.

The KEYMASKED, KEYENCRYPTED, ENCRYPTKEY and KEYLABEL keywords control the key to be used for the generation and evaluation of legacy PassTickets. These keywords indicate the method you want to use to protect the legacy PassTicket key value within the RACF database. You can mask or encrypt the key. The key-value represents a 64-bit (8-byte) key that must be represented as 16 hexadecimal characters. The valid characters are 0 - 9 and A - F.

The EPTKEYLABEL, TYPE, TIMEOUT and REPLAY keywords control the key and settings to be used for the generation and evaluation of enhanced PassTickets.

Note:

1. Before defining PassTicket keys, please read and understand the PassTicket documentation in the z/OS Security Server RACF Security Administrator's Guide, specifically, the topic Protecting PassTicket keys. That documentation contains important information on setup and authorization issues, especially pertaining to the use of ICSF with encrypted keys.
2. As with RACF passwords, the database unload facility does not unload application keys or PassTicket keys. It will, however, indicate the method of protection of the key, and if the key is encrypted, the key label name.
3. The RLIST command does not list the value of the application key or the PassTicket key. Therefore, when you define the keys, you should note the value and keep it in a secure place. Note that RLIST will, however, indicate the method of protection of the key, and if the key is encrypted, the key label name.
4. The KEYMASKED, KEYENCRYPTED, ENCRYPTKEY and KEYLABEL legacy PassTicket keywords all work against the same field in the RACF database. Use of any of these RALTER keywords replaces the previous legacy PassTicket key (or its label) in the RACF database.

KEYMASKED(legacy-passticket-key-value)

Specifies that you want to mask the legacy PassTicket key value using the masking algorithm.

Note:

1. IBM **STRONGLY** recommends that masked PassTicket keys are not used outside of a test environment.
2. You can specify this operand only once for each application key.
3. If you mask a key, you cannot encrypt it. These are mutually exclusive.

KEYENCRYPTED(legacy-passticket-key-value)

Specifies that you want to encrypt the Legacy PassTicket key value.

Note:

1. Before using the KEYENCRYPTED keyword, please read and understand the documentation describing Encrypting the PassTicket key in the z/OS Security Server RACF Security Administrator's Guide.
2. You can specify this operand only once for each application key.
3. If you encrypt a key, you cannot mask it. These are mutually exclusive.
4. ICSF must be installed and active on the system.

   You can use the RLIST command to verify that the key is protected.

ENCRYPTKEY

Specifies that you want to request conversion of a legacy PassTicket key to a KEYENCRYPTED key with a key label.

If the existing key is KEYMASKED, it is converted to a KEYENCRYPTED key and the data in the RACF database is replaced with the ICSF key label. Knowledge of the existing key value is not necessary.

If the existing key is KEYENCRYPTED in the form of a key token, it is moved into the ICSF CKDS and data in the RACF database is replaced with a key label. Knowledge of the existing key value is not necessary.

If the existing key is KEYENCRYPTED and already referenced by a key label, message IRR52254I is issued and ENCRYPTKEY is ignored.

RACF generates key label names in the form IRR.SSIGNON.sysname.mmddyyyy.hhmmss.nnnnnn. The key label name is not user configurable. RLIST displays the key label name. Sysname indicates the name of the system on which the ENCRYPTKEY operation was performed.

The SEARCH command with the CLIST option provides a way of creating a 'utility' to convert all your PassTicket keys to KEYENCRYPTED in ICSF.

KEYLABEL(legacy-passticket-label-value)

Specifies the name of an ICSF key label to be used when generating or evaluating a legacy PassTicket.

ICSF must be installed and active, and the key must be defined in the ICSF CKDS at

the time of use. However, this is not checked when the KEYLABEL keyword is specified.

When using KEYLABEL, RACF does not make any calls to ICSF. The key label is saved in the RACF database, and it is up to the installation to ensure that the key is added to the ICSF CKDS before any PassTicket operations occur which need it. The key must refer to a DES key with a type of DATA and a length of 8 bytes.

Note:

The KEYLABEL operand cannot be used to override the key label generated by RACF when KEYENCRYPTED or ENCRYPTKEY is specified.

NOLEGACYKEY

Removes an existing legacy PassTicket key from the PTKTDATA profile set by the KEYMASKED, KEYENCRYPTED or KEYLABEL keywords.

EPTKEYLABEL | NOEPTKEYLABEL

EPTKEYLABEL(enhanced-passticket-*label-value*)

Specifies the name of an ICSF key label to be used when generating or evaluating an enhanced PassTicket.

ICSF must be installed and active, and the key must be defined in the ICSF CKDS at the time of use. However, this is not checked when the EPTKEYLABEL keyword is specified.

When using EPTKEYLABEL, RACF does not make any calls to ICSF. The key label is saved in the RACF database, and it is up to the installation to ensure that the key is added to the ICSF CKDS before any enhanced PassTicket operations occur which need it.

The key label must refer to an ICSF HMAC key with a key algorithm of HMAC, a key type of MAC and the key usage fields must indicate GENERATE. The supported HMAC key size range is from 32 to 256 bytes. The recommended minimum key size is 64 bytes.

The RACF enhanced PassTicket support uses ICSF HMAC keys which require that the ICSF CKDS is defined in either the variable length record format or common record format (KDSR). For more information on ICSF CKDS formats please refer to Chapter 1 of the z/OS: Cryptographic Services Integrated Cryptographic Service Facility System Programmer's Guide (SC14-7507-09).

The label name cannot exceed 64 characters. The first character must be an alphabetic character or a national character (#, @, or $). Subsequent characters can be a period character (.) or any alphanumeric or national character.

NOEPTKEYLABEL

Removes the enhanced PassTicket key label.

TYPE | NOTYPE

TYPE(*UPPER* | ***MIXED*)**

Specifies the character set to use for generating and evaluating an enhanced

PassTicket.

The type must be one of the following values:

UPPER – The enhanced PassTicket will be generated and evaluated with only uppercase characters A - Z and digits 0 - 9.

MIXED – The enhanced PassTicket will be generated and evaluated with uppercase characters A - Z, lowercase characters a - z, digits 0 - 9 and the symbols dash (-) and underscore (_).

Using type MIXED is recommended as it provides a larger set of possible PassTicket values and therefore provides more security. Type UPPER may be required when an application does not yet support mixed case passwords.

The default value is MIXED.

NOTYPE

Resets TYPE to the default value of MIXED.

TIMEOUT | NOTIMEOUT

TIMEOUT(*timeout-seconds*)

Specifies the number of seconds that the enhanced PassTicket is active.

The value of timeout-seconds can be between 1 and 600 seconds (10 minutes).

The default value is 60 seconds.

NOTIMEOUT

Resets TIMEOUT to the default value of 60 seconds.

REPLAY(*YES* | *NO*)

Specifies whether an enhanced PassTicket is allowed to be replayed within the TIMEOUT value.

The default value is NO.

This setting only applies to enhanced PassTickets and does not apply to legacy PassTickets.

The replay protection setting in the APPLDATA field only applies to legacy PassTickets and does not apply to enhanced PassTickets.

NOSSIGNON

Specifies that the SSIGNON segment should be deleted.

3.2.3 RLIST

A new SSIGNON segment is updated to add new fields for enhanced PassTickets. RLIST is enhanced to display the new fields.

Syntax

…

[SSIGNON]

SSIGNON

Specifies that you want to display the secured signon information.

Note: Each line of the RLIST SSIGNON segment output is prefixed to indicate that it is legacy PassTicket information or enhanced PassTicket information. RLIST will display the default values for the TIMEOUT and REPLAY keywords even when no enhanced PassTicket key is configured. These prefixes are displayed for the SSIGNON segment fields for all RACF classes even those not necessarily related to PassTicket functions.

Note: The PassTicket key value cannot be displayed. However, information is displayed that describes whether the key value is masked or encrypted, and if encrypted, the ICSF key label name.

When the SSIGNON segment contains a PassTicket key, RLIST displays:

```
SSIGNON INFORMATION
-------------------
```

When a legacy PassTicket masked key exists, the following will be displayed:

```
Legacy   PassTicket: KEYMASKED DATA NOT DISPLAYABLE
```

When a legacy PassTicket key token exists, the following will be displayed:

```
Legacy   PassTicket: KEYTOKEN DATA NOT DISPLAYABLE
```

When a legacy PassTicket key label exists, the following (for example) will be displayed:

```
Legacy   PassTicket: KEYENCRYPTED LABEL:
IRR.SSIGNON.SY1.07192018.185056.915782
```

When an enhanced PassTicket key label exists, the following (for example) will be displayed:

```
Enhanced PassTicket: Key Label = EPTKEY.APPL01
```

EXAMPLES

Example	Activity Label	Description
…		
4	*Operation*	The security administrator wants to display secured signon key information for profile name TSOR001 in the PTKTDATA class to be certain that the application key is masked instead of encrypted.
	Known	ELVIS1 is the user ID of the security administrator and has the SPECIAL attribute. The security administrator wants to issue the command as a RACF TSO command.
	Command	RLIST PTKTDATA TSOR001 SSIGNON
	Defaults	None.
	Output	See Figure 63
5	*Operation*	The security administrator wants to display secured signon key information for profile name TSOR004 in the PTKTDATA class and to be certain that the application key is encrypted instead of masked.
	Known	NONNEL is the user ID of the security administrator and has the SPECIAL attribute. The security administrator wants to issue the command as a RACF operator command, and the RACF subsystem prefix is @.
	Command	@RLIST PTKTDATA TSOR004 SSIGNON
	Defaults	None.
	Output	See Figure 64
…		
18	*Operation*	The security administrator wants to display secured signon key information for profile name APPL01 in the PTKTDATA class and which contains both a legacy PassTicket key label and an enhanced PassTicket key label.
	Known	The security administrator has the SPECIAL attribute.
	Command	RLIST PTKTDATA APPL01 SSIGNON
	Defaults	None.

	Output	See Figure 78

...

```
SSIGNON INFORMATION
-------------------
Legacy   PassTicket: KEYMASKED DATA NOT DISPLAYABLE
Enhanced PassTicket: Timeout = 00000060
Enhanced PassTicket: Replay allowed = NO
```

Figure 63: Output from the RLIST command

```
SSIGNON INFORMATION
-------------------
Legacy   PassTicket: KEYENCRYPTED DATA NOT DISPLAYABLE
Enhanced PassTicket: Timeout = 00000060
Enhanced PassTicket: Replay allowed = NO
```

Figure 64: Output from the RLIST command

...

```
SSIGNON INFORMATION
-------------------
Legacy   PassTicket: KEYENCRYPTED LABEL: IRR.SSIGNON.SY1.07192018.185056.915782
Enhanced PassTicket: Key Label = EPTKEY.APPL01
Enhanced PassTicket: Type = UPPER
Enhanced PassTicket: Timeout = 00000120
Enhanced PassTicket: Replay Allowed = YES
```

Figure 78: Output from the RLIST command

3.3 z/OS Security Server RACF Callable Services

This information supplements the following chapters and sections:

- Chapter: '*Callable services descriptions'*
  - Section: 'R_Admin (IRRSEQ00): RACF administration API'
  - Section: 'R_GenSec (IRRSGS00 or IRRSGS64): Generic security API interface'
  - Section: 'R_ticketserv (IRRSPK00): Parse or extract'

3.3.1 R_Admin (IRRSEQ00): RACF administration API

The R_admin reference appendix is updated to add new fields to the table '*SSIGNON segment fields'*:

SSIGNON segment fields:

Field name	SAF field name	Flag byte value	RDEFINE/RALTER keyword reference	Allowed on add requests	Allowed on alter requests	Returned on extract requests
…						
PTKEYLAB	ptkeylab	'Y'	SSIGNON (EPTKEYLABEL)	Yes	Yes	Yes
		'N'	SSIGNON (NOEPTKEYLABEL)	No	Yes	
PTTYPE	pttype	'Y'	SSIGNON (TYPE(xx))	Yes	Yes	Yes
		'N'	SSIGNON (NOTYPE)	No	Yes	
PTTIMEO	pttimeo	'Y'	SSIGNON (TIMEOUT(xx))	Yes	Yes	Yes
		'N'	SSIGNON (NOTIMEOUT)	No	Yes	
PTREPLAY (boolean)	ptreplay	'Y'	SSIGNON (REPLAY(YES))	Yes	Yes	Yes
		'N'	SSIGNON (REPLAY(NO))	Yes	Yes	

3.3.2 R_GenSec (IRRSGS00 or IRRSGS64): Generic security API interface

The R_GenSec callable service description is updated to indicate that it also supports generation and evaluation of enhanced PassTickets.

…

Subfunction codes	
Value	**Subfunction**
1	Generate PassTicket
2	Evaluate PassTIcket
3	Evaluate PassTicket Extended
4	Generate PassTicket Extended

Generate PassTicket(1 and 4)
This function will generate a legacy PassTicket or enhanced PassTicket for a specified userid and application name. The type of PassTicket returned is based on the keys configured in the associated PTKTDATA class profile:

- An enhanced PassTicket is returned when an enhanced PassTicket key label is configured with the EPTKEYLABEL keyword.
- A legacy PassTicket is returned when a legacy PassTicket key is configured with the KEYMASKED, KEYENCRYPTED or KEYLABEL keywords and no enhanced PassTicket key label is configured.

If option code 4 (PassTicket Generate Extended) is specified, additional reason codes are provided in the event of a PassTicket Generation failure. This is the only difference between the PassTicket Generate and PassTicket Generate Extended options. IBM recommends using the extended option if your application reports SAF return and reason codes in a trace log, or other diagnostic medium.

…

Evaluate PassTicket(2 and 3)
This function will evaluate a legacy PassTicket or enhanced PassTicket for a specified userid and application name. When the associated PTKTDATA class profile contains a legacy PassTicket key the specified PassTicket value is evaluated as a legacy PassTicket. When the PTKTDATA class profile contains an enhanced PassTicket key the specified PassTicket value is evaluated as an enhanced PassTicket. When the PTKTDATA class profile contains both a legacy PassTicket key and enhanced PassTicket key the specified PassTicket value is evaluated as both a legacy PassTicket and enhanced PassTicket.
…

Return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
…			
8	16	X'nnnnnnnn'	PassTicket generation extended failure. X'nnnnnnnn' is the internal reason code for the generation failure.

3.3.3 R_ticketserv (IRRSPK00): Parse or extract

The R_ticketserv callable service description is updated to indicate that it also supports generation and evaluation of enhanced PassTickets and to document the new Generate Extended function.

Function

…

R_ticketserv also allows callers to generate and evaluate PassTickets. Both legacy and enhanced PassTickets are supported.

Parameters

…

Ticket_options

The name of a fullword containing the address of a binary bit string that identifies the ticket-specific processing to be performed. This parameter is unused when a function code of X'0001' is specified.

When function code X'0003' is specified, the bit string is used as an integer to specify which PassTicket operation to perform.

```
X'00000001' - Generate a PassTicket
X'00000002' - Evaluate a PassTicket
X'00000003' - Evaluate a PassTicket Extended
X'00000004' - Generate a PassTicket Extended
```

If option code 3 (PassTicket Evaluate Extended) is specified, additional reason codes are provided in the event of a PassTicket evaluation failure. This is the only difference between the PassTicket Evaluate and PassTicket Evaluate Extended options. IBM recommends using the extended option if your application reports SAF return and reason codes in a trace log, or other diagnostic medium.

If option code 4 (PassTicket Generate Extended) is specified, additional reason codes are provided in the event of a PassTicket generation failure. This is the only difference between the PassTicket Generate and PassTicket Generate Extended options. IBM recommends using the extended option if your application reports SAF return and reason codes in a trace log, or other diagnostic medium.

Return and reason codes

SAF return code	RACF return code	RACF reason code	Explanation
…			
8	16	X'nnnnnnnn'	PassTicket generation extended failure. X'nnnnnnnn' is the internal reason code for the generation failure.

3.4 z/OS Security Server RACF Macros and Interfaces

This information supplements the following chapters and sections:

- Chapter: '*RACF database unload'*
  - Section: 'Record formats produced by the database unload utility'
- Chapter: '*SMF records'*
  - Section: Record type 80: RACF processing record
  - Section: Format of SMF type 80 records
- Chapter: *The format of the unloaded SMF type data*
  - Section: The JOBINIT record extension
- Chapter: '*The RACF PassTicket'*
- Appendix: '*Supplied class descriptor table entries'*
- Appendix: '*RACF database templates*'
  - Section: User template for the RACF database
  - Section: General template for the RACF database

3.4.1 Record formats produced by the database unload utility

The General Resource SSIGNON Data Record (0530) is updated to add new fields for enhanced PassTickets.

Field Name	Type	Start	End	Comments
GRSIGN_RECORD_TYPE	Int	1	4	Record type of the SSIGNON record (0530)
GRSIGN_NAME	Char	6	251	General resource name as taken from the profile name.
GRSIGN_CLASS_NAME	Char	253	260	Name of the class to which the general resource profile belongs.
GRSIGN_PROTECTION	Char	262	325	Method of protection for the legacy PassTicket key. Can contain one of the following values: • *MASKED* - KEYMASKED keyword was used • *KEYTOKEN* - KEYENCRYPTED was used, and the key exists within a key token, perhaps due to an error with ICSF • Key label name – KEYLABEL or KEYENCRYPTED was used, and the key is stored in ICSF with a key label. The output is the label name, which is a 64-character value padded with blanks if

				necessary. • *UNKNOWN* - the format of the data is unrecognized.
GRSIGN _KEY_LABEL	Char	327	390	The enhanced PassTicket ICSF CKDS Key Label name.
GRSIGN _TYPE	Char	392	403	Enhanced PassTicket type.
GRSIGN _TIMEOUT	Int	405	414	Enhanced PassTicket timeout setting.
GRSIGN _REPLAY	Yes/ No	416	419	Indicates whether enhanced PassTicket replays are allowed.

3.4.2 Record type 80: RACF processing record

Type 80 event code 1 (RACROUTE REQ=VERIFY/X) record:

The “Table of extended-length relocate section variable data” is updated to add new enhanced PassTicket information to existing relocate 443 and to add new relocate 67.

Data type (SMF80TP2) dec(hex)	Data length (SMF80DL2)	Format	Audited by event code	Description (SMF80DA2)
…				
67(43)	variable	mixed	81, 82	Byte 1: PassTicket Generation or Evaluation Details **Bit** / **Meaning when set** 0 – Legacy PassTicket 1 – Evaluation: Legacy PassTicket Successful 2 – Enhanced PassTicket Type UPPER 3 – Evaluation: Enhanced PassTicket Type UPPER Successful 4 – Enhanced PassTicket Type

<table>
<tr><td></td><td></td><td></td><td></td><td>
<table>
<tr><td></td><td>MIXED</td></tr>
<tr><td>5</td><td>Evaluation: Enhanced PassTicket Type MIXED Successful</td></tr>
<tr><td>6</td><td>Evaluation: Failure due to PassTicket replay attempt</td></tr>
<tr><td>6-7</td><td>Reserved</td></tr>
</table>

Byte 2: Reserved

Byte 3-6: Return Code

Bytes 7-10: Reason Code

Bytes 11-18: Application Name
</td></tr>
<tr><td>…</td><td></td><td></td><td></td><td></td></tr>
<tr><td>443(1BB)</td><td>variable</td><td>mixed</td><td>1</td><td>Byte 1: Authentication information:
<table>
<tr><th>Bit</th><th>Meaning when set</th></tr>
<tr><td>0</td><td>Authenticated from VLF</td></tr>
<tr><td>1</td><td>User has active MFA factor(s)</td></tr>
<tr><td>2</td><td>MFA user allowed to fall back when no MFA decision can be made</td></tr>
<tr><td>3</td><td>No MFA decision for MFA user</td></tr>
<tr><td>4</td><td>IBMMFA requested that RACROUTE REQUEST=VERIFY return the password-expired return code.</td></tr>
<tr><td>5</td><td>IBM MFA requested that RACROUTE REQUEST=VERIFY return the new-password-invalid return code.</td></tr>
<tr><td>6</td><td>IBM MFA requested that RACROUTE REQUEST=VERIFY return the password-invalid return code, but not to increment the password revoke count (partial success – needs more information).</td></tr>
<tr><td>7</td><td>Relocate 443 is extended.</td></tr>
</table>
</td></tr>
</table>

Byte 2: Authenticator used:

Bit	Meaning when set
0	Password Evaluated
1	Password Successful
2	Password Phrase Evaluated
3	Password Phrase Successful
4	PassTicket Evaluated
5	PassTicket Successful
6	MFA authentication successful
7	MFA authentication unsuccessful

Byte 3-6: MFA Authorization Return Code

Bytes 7-10: MFA Authorization Reason Code

Note: Below fields are only present when relocate 443 is extended.

Byte 11-14: PassTicket Return Code

Bytes 15-18: PassTicket Reason Code

Byte 19: Flag byte 3: Authentication Details

Bit	Meaning when set
0	Password or Password Phrase expired
1	New Password or Password Phrase invalid
2	Identity Token (IDT) Evaluated
3	Identity Token (IDT) Successful
4	IBM MFA requested that RACROUTE REQUEST=VERIFY return the password-invalid return code, but not to increment the password revoke count (reauthentication requested).

				5 Legacy PassTicket Evaluated 6 Legacy PassTicket Successful 7 Enhanced PassTicket Type UPPER Evaluated Byte 20: Flag byte 4: Authentication Details Bit 0 Enhanced PassTicket Type UPPER Successful 1 Enhanced PassTicket Type MIXED Evaluated 2 Enhanced PassTicket Type MIXED Successful 3-7 Reserved Bytes 21-28: Derived Application Name Byte 29-32: IDT Validation Reason Code Byte 33-36: IDT Error Reason Code Byte 37-40: Failing Service ID Byte 41-44: Failing Service Return Code Byte 45-48: Failing Service Reason Code

3.4.3 The format of the unloaded SMF type 80 data

The JOBINIT record extension

The JOBINIT record extension relocate section 443 is updated to reuse former reserved fields as follows:

INIT_RESERVED_01 as INIT_LPT_EVAL
INIT_RESERVED_02 as INIT_LPT_SUCC
INIT_RESERVED_03 as INIT_EPT_UPPER_EVAL
INIT_RESERVED_04 as INIT_EPT_UPPER_SUCC
INIT_RESERVED_05 as INIT_EPT_MIXED_EVAL
INIT_RESERVED_06 as INIT_EPT_MIXED_SUCC

Field Name	Type	Start	End	Comments
INIT_ACEE_VLF	Yes/ No	4540	4543	The ACEE was created from the VLF cache
INIT_MFA_USER	Yes/ No	4545	4548	The user has active MFA factors
INIT_MFA_FALLBACK	Yes/ No	4550	4553	The MFA user is allowed to fall back to password authentication when MFA is unavailable
INIT_MFA_UNAVAIL	Yes/ No	4555	4558	MFA was unavailable to make an authentication decision for the MFA user
INIT_MFA_PWD_EXPIRED	Yes/ No	4560	4563	IBM MFA requested that RACROUTE REQUEST=VERIFY return the password-expired return code
INIT_MFA_NPWD_INV	Yes/ No	4565	4568	IBM MFA requested that RACROUTE REQUEST=VERIFY return the new-password-invalid return code
INIT_MFA_PART_SUCC	Yes/ No	4570	4573	IBM MFA requested that RACROUTE REQUEST=VERIFY return the password-invalid return code, but not to increment the password revoke count (partial success – needs more information).
INIT_RELO443_EXTENDED	Yes/ No	4575	4578	Relocate 443 is extended.
INIT_PASSWORD_EVAL	Yes/ No	4580	4583	The supplied password was evaluated
INIT_PASSWORD_SUCC	Yes/ No	4585	4588	The supplied password was evaluated successfully
INIT_PHRASE_EVAL	Yes/ No	4590	4593	The supplied password phrase was evaluated
INIT_PHRASE_SUCC	Yes/ No	4595	4598	The supplied password phrase was evaluated successfully
INIT_PASSTICKET_EVAL	Yes/ No	4600	4603	The supplied password was evaluated as a PassTicket
INIT_PASSTICKET_SUCC	Yes/ No	4605	4608	The supplied password was evaluated successfully as a PassTicket
INIT_MFA_SUCC	Yes/ No	4610	4613	The supplied password phrase/phrase was evaluated successfully as multifactor data

INIT_MFA_FAIL	Yes/ No	4615	4618	The supplied password/phrase was evaluated unsuccessfully as MFA data
INIT_AUTH_RSN1	Char	4620	4627	MFA Authentication return code. Expressed as hexadecimal number.
INIT_AUTH_RSN2	Char	4629	4636	MFA Authentication reason code. Expressed as hexadecimal number.
INIT_AUTH_RSN3	Char	4638	4645	PassTicket Authentication return code. Expressed as hexadecimal number.
INIT_AUTH_RSN4	Char	4647	4654	PassTicket Authentication reason code. Expressed as hexadecimal number.
INIT_PWD_PHR_EXPIRED	Yes/ No	4656	4659	The supplied password or password phrase was expired.
INIT_NPWD_NPHR_NONVAL	Yes/ No	4661	4664	The supplied new password or new password phrase was not valid.
INIT_IDT_EVAL	Yes/ No	4666	4669	The supplied Identity Token (IDT) was evaluated.
INIT_IDT_SUCC	Yes/ No	4671	4674	The supplied Identity Token (IDT) was evaluated successfully.
INIT_MFA_REAUTHENT	Yes/ No	4676	4679	IBM MFA requested that RACROUTE REQUEST=VERIFY return the password-invalid return code, but not to increment the password revoke count (reauthentication requested).
~~INIT_RESERVED_01~~ INIT_LPT_EVAL	Yes/ No	4681	4684	The supplied Password was evaluated as a legacy PassTicket.
~~INIT_RESERVED_02~~ INIT_LPT_SUCC	Yes/ No	4686	4689	The supplied Password was evaluated successfully as a legacy PassTicket.
~~INIT_RESERVED_03~~ INIT_EPT_UPPER_EVAL	Yes/ No	4691	4694	The supplied Password was evaluated as an enhanced PassTicket type UPPER.
~~INIT_RESERVED_04~~ INIT_ EPT_UPPER_SUCC	Yes/ No	4696	4699	The supplied Password was evaluated successfully as an enhanced PassTicket type UPPER.
~~INIT_RESERVED_05~~ INIT_EPT_MIXED_EVAL	Yes/ No	4701	4704	The supplied Password was evaluated as an enhanced PassTicket type MIXED.
~~INIT_RESERVED_06~~ INIT_EPT_MIXED_SUCC	Yes/ No	4706	4709	The supplied Password was evaluated successfully as an enhanced PassTicket type MIXED.

INIT_RESERVED_07	Yes/ No	4711	4714	Reserved for IBM's use
INIT_RESERVED_08	Yes/ No	4716	4719	Reserved for IBM's use
INIT_RESERVED_09	Yes/ No	4721	4724	Reserved for IBM's use
INIT_RESERVED_10	Yes/ No	4726	4729	Reserved for IBM's use
INIT_RESERVED_11	Yes/ No	4731	4734	Reserved for IBM's use
INIT_DERIVED_APPL_NAM	Char	4736	4743	Derived Application Name
INIT_IDT_VALIDTN_RSNC	Char	4745	4752	IDT Validation Reason Code
INIT_IDT_ERROR_RSNC	Char	4754	4761	IDT Error Reason Code
INIT_SERVICE_CODE	Char	4763	4770	Failing Service Identifier
INIT_SERVICE_RC	Char	4772	4779	Failing Service Return Code
INIT_SERVICE_RSNC	Char	4781	4788	Failing Service Reason Code

Type 80 event code 81 (PassTicket Evaluation) record:

The PTEVAL record extension is updated to add relocate section 67.

Field Name	Type	Start	End	Comments
…				
PTEV_LPT_EVAL	Yes/ No	486	489	The supplied password was evaluated as a legacy PassTicket
PTEV_LPT_SUCC	Yes/ No	491	484	The legacy PassTicket was evaluated successfully.
PTEV_EPT_UPPER_EVAL	Yes/ No	496	499	The supplied Password was evaluated as an enhanced PassTicket type UPPER.
PTEV_EPT_UPPER_SUCC	Yes/ No	501	504	The supplied Password was evaluated successfully as an enhanced PassTicket type UPPER.
PTEV_EPT_MIXED_EVAL	Yes/ No	506	509	The supplied Password was evaluated as an enhanced PassTicket type MIXED.
PTEV_EPT_MIXED_SUCC	Yes/	511	514	The supplied Password was evaluated

	No			successfully as an enhanced PassTicket type MIXED.
PTEV_REPLAY_FAILURE	Yes/ No	516	519	Failure due to replay attempt.
PTEV_RESERVED_08	Yes/ No	521	524	Reserved for IBM's use
PTEV_RESERVED_09	Yes/ No	526	529	Reserved for IBM's use
PTEV_RESERVED_10	Yes/ No	531	534	Reserved for IBM's use
PTEV_RESERVED_11	Yes/ No	536	539	Reserved for IBM's use
PTEV_RESERVED_12	Yes/ No	541	544	Reserved for IBM's use
PTEV_RESERVED_13	Yes/ No	546	549	Reserved for IBM's use
PTEV_RESERVED_14	Yes/ No	551	554	Reserved for IBM's use
PTEV_RESERVED_15	Yes/ No	556	559	Reserved for IBM's use
PTEV_RESERVED_16	Yes/ No	561	564	Reserved for IBM's use
PTEV_APPL_NAME	Char	566	573	Application name used to evaluate the PassTicket.
PTEV_EVAL_RSN1	Char	575	582	Evaluation Return Code. Expressed as hexadecimal number.
PTEV_EVAL_RSN2	Char	584	591	Evaluation Reason Code. Expressed as hexadecimal number.

Type 80 event code 82 (PassTicket Generation) record:

The PTCREATE record extension is updated to add relocate section 67.

Field Name	Type	Start	End	Comments
PTCR_LPT	Yes/ No	486	489	Generate of a legacy PassTicket was attempted
PTCR_RESERVED_02	Yes/ No	491	494	Reserved for IBM's use

PTCR_EPT_UPPER	Yes/ No	496	499	Generate of an enhanced PassTicket type UPPER was attempted
PTCR_RESERVED_04	Yes/ No	501	504	Reserved for IBM's use
PTCR_EPT_MIXED	Yes/ No	506	509	Generate of an enhanced PassTicket type MIXED was attempted
PTCR_RESERVED_06	Yes/ No	511	514	Reserved for IBM's use
PTCR_RESERVED_07	Yes/ No	516	519	Reserved for IBM's use
PTCR_RESERVED_08	Yes/ No	521	524	Reserved for IBM's use
PTCR_RESERVED_09	Yes/ No	526	529	Reserved for IBM's use
PTCR_RESERVED_10	Yes/ No	531	534	Reserved for IBM's use
PTCR_RESERVED_11	Yes/ No	536	539	Reserved for IBM's use
PTCR_RESERVED_12	Yes/ No	541	544	Reserved for IBM's use
PTCR_RESERVED_13	Yes/ No	546	549	Reserved for IBM's use
PTCR_RESERVED_14	Yes/ No	551	554	Reserved for IBM's use
PTCR_RESERVED_15	Yes/ No	556	559	Reserved for IBM's use
PTCR_RESERVED_16	Yes/ No	561	564	Reserved for IBM's use
PTCR_APPL_NAME	Char	566	573	Application Name used to generate the PassTicket.
PTCR_GEN_RSN1	Char	575	582	Generation Return Code. Expressed as hexadecimal number.
PTCR_GEN_RSN2	Char	584	591	Generation Reason Code. Expressed as hexadecimal number.

3.4.4 RACF database templates

The SSIGNON segment is updated in the GENERAL section to add new fields for enhanced PassTickets.

```
$/SEGMENT  004 SSIGNON
SSIGNON  001 00 00 00000000 00 SSIGNON - START OF SEGMENT FIELDS
SSKEY    002 00 00 00000000 00 SSIGNON - SECURE SIGNON KEY
PTKEYLAB 003 00 00 00000000 00 SSIGNON - EPT key label
PTTYPE   004 00 00 00000000 00 SSIGNON - PassTicket Type
PTTIMEO  005 00 00 00000004 00 SSIGNON - PassTicket Timeout
PTREPLAY 006 00 00 00000001 00 SSIGNON - PassTicket Replay
```

The RACF templates version is updated to:

```
VERSION OA59196 00000243.00000050
```

3.4.5 The RACF PassTicket

This chapter is updated to add details about enhanced PassTickets. Updated sections are listed below with additions highlighted.

Introduction:
The RACF PassTicket is a one-time-only password that is generated by a requesting product or function. It is an alternative to the RACF password that removes the need to send RACF passwords across the network in clear text. It makes it possible to move the authentication of a mainframe application user ID from RACF to another authorized function executing on the host system or to the work station local area network (LAN) environment. RACF provides support for the following PassTicket functions:

- Generating a PassTicket.
- Evaluating a PassTicket.

RACF PassTickets can be configured with two different algorithms:

- The legacy PassTicket algorithm
- The enhanced PassTicket algorithm

The legacy PassTicket algorithm is the original PassTicket implementation and uses a DES secret key. The enhanced PassTicket algorithm is an updated version of the PassTicket algorithm and uses an HMAC secret key. RACF supports generation and evaluation of PassTickets with either the legacy PassTicket algorithm or the enhanced PassTicket algorithm based on system configuration. IBM highly recommends using the enhanced PassTicket algorithm as it provides the same capabilities as the legacy PassTicket algorithm but also provides increased security.
For more information on configuring PassTickets see “The RACF PassTicket” in the z/OS Security Server RACF Security Administrator’s Guide.

Generating and evaluating a PassTicket

A product or function that generates a PassTicket must use the RACF legacy PassTicket generator algorithm or enhanced PassTicket generation algorithm. These algorithms require specific information as input data and produces a PassTicket that substitutes for a specific end-user RACF password. RACF uses the PassTicket to authenticate the end-user for a specific application running on a specific system that uses RACF for identification and authentication.

There are four ways to generate and evaluate a PassTicket using the legacy PassTicket algorithm or enhanced PassTicket algorithm:

- If the function using PassTickets is running on a z/OS system, you can use the RACF PassTicket generation service (RCVTPTGN) to generate the PassTicket. The algorithm is already incorporated into the service and allows RACF to generate a PassTicket on the host. An authorized program, such as one authorized by the authorized program facility (APF), can use the service to generate PassTickets. See “Using the RCVTPTGN service to generate a PassTicket” for more information.
- For any function that generates a PassTicket, you can create a program that incorporates the algorithm. See “Incorporating the PassTicket generator algorithm into your program” for more information.
- You can use the R_ticketserv and R_GenSec callable services. This interface supports problem state callers, and both 31-bit and 64-bit callers. For more information about these callable services, see R_ticketserv (IRRSPK00): Parse or extract and R_GenSec (IRRSGS00 or IRRSGS64): Generic security API interface in z/OS Security Server RACF Callable Services.
- Java™ code can use a Java interface that uses a Java Native Interface (JNI) and calls the R_ticketserv and R_GenSec callable services. For information about this interface, see the JavaDoc shipped in the IRRRacfDoc.jar file, which is installed into the directory /usr/include/java_classes. Download the jar file to a workstation, un-jar it, and read it with a Web browser.

Using the RCVTPTGN service to generate a PassTicket

To allow RACF to authenticate a user with a PassTicket instead of a password, the non-RACF function performing the authentication calls the RCVTPTGN service to build a PassTicket.

The RCVTPTGN service:

- Is branch-entered by callers.
- Is not supported in cross-memory mode. Access register (AR) mode must use address space control (ASC).
- Is not supported in SRB mode.
- Requires that the caller be in key zero.
- Is unable to generate PassTickets using the PTKTDATA profiles which are qualified by user id and / or group. It can only generate PassTickets using profiles which match the application name.
- Supports generation of legacy PassTickets or enhanced PassTickets based on RACF configuration.

Before calling the PassTicket-generation service, the application must locate the address of the service. You can find this address from field RCVTPTGN in the RACF communications vector table (RCVT). The ICHPRCVT macro maps the RCVT and field CVTRAC points to it in the MVS communications vector table (CVT).

How the PassTicket-generation service works

The service:

- Uses standard linkage
- Uses the current system time, expressed in Greenwich Mean Time (GMT), [1] as input for the algorithm
- Returns the PassTicket in general purpose register 0 (the leftmost four characters) and general purpose register 1 (the rightmost four characters)
- The type of PassTicket returned is based on the keys configured in the associated PTKTDATA class profile:
  - An enhanced PassTicket is returned when an enhanced PassTicket key label is configured with the EPTKEYLABEL keyword.
  - A legacy PassTicket is returned when a legacy PassTicket key is configured with the KEYMASKED, KEYENCRYPTED or KEYLABEL keywords and no enhanced PassTicket key label is configured.
  - In the case where a PTKTDATA class profile is configured to contain both a legacy PassTicket key and enhanced PassTicket key an enhanced PassTicket is returned.
- Provides return codes
  - If a PassTicket is produced, register 15 contains a return code of 0
  - If a PassTicket is not produced, register 15 contains return code of 8
  - Register 0 contains a reason code. The 1st byte of the reason code indicates the problem, the other 3 bytes may contain additional information:

Value	Meaning	Bytes 2-4
12	ICSF CSNBENC service failed	Byte 2=ICSF RC Byte 3 and 4=ICSF RSN
16	RACROUTE REQUEST=EXTRACT, TYPE=ENCRYPT failed	Byte 2=SAFRC from RACROUTE Bytes 3 and 4=0
20	PTKTDATA class inactive	0
24	No profiles defined to the PTKTDATA class	0
28	Unable to load ICSF CSFACEE or CSFIQF service	Byte 2=Reason code from z/OS LOAD macro
36	PTKTDATA profile representing the APPL not found or the PTKTDATA profile does not have a key saved in the SSIGNON segment	0
52	Caller not in key 0	0
56	ICSF not initialized	Byte 2=ICSF RC

		Byte 3 and 4=ICSF RSN
60	ICSF CSNBHMG service failed.	Byte 2=ICSF RC Byte 3 and 4=ICSF RSN
Other = Internal error		

Notes:
1. Register 13 must point to a standard save area.
2. No additional recovery processing is provided by the PassTicket-generation service beyond what is already in effect within the invoking program.

Invoking the PassTicket-generation service
Following is an example of a generalized programming technique you can use with assembler language to invoke a service. It is not intended to be syntactically correct.

```
L 15,RCVTPTGN
CALL (15),(userid,appname)
```

where:
userid
Is the RACF user ID of the user the PassTicket authenticates. This field is a maximum of 9 bytes. The first byte contains the length of the non-blank portion of the *userid* field that follows. Bytes 2 through 9 contain the user ID and must be in uppercase and left-justified in the field.
appname
Is the application name that the PassTicket-generation service uses to locate the key used in the PassTicket generator algorithm. This field is a maximum of 9 bytes. The first byte is the length of the non-blank portion of the *appname* field that follows. Bytes 2 through 9 contain the application name and must be in uppercase and left-justified in the field.
When the service is invoked, only the *appname* (not the *userid* or *group*) is used to locate the PassTicket key. It is not possible to use the RCVTPTGN service to generate PassTickets using keys which are stored in user id or group id qualified profiles.

Incorporating the PassTicket generator algorithm into your program
To generate a PassTicket without using the RACF service, callable services, or Java interface, you need to incorporate either the RACF legacy PassTicket generator algorithm or enhanced PassTicket generator algorithm into your program.

The RACF PassTicket algorithms each consist of two parts:
- The RACF PassTicket generator
- The RACF PassTicket time-coder

The time-coder is invoked from within the RACF PassTicket generator and returns its results to the generator.

The flowcharts in Figure 6 and Figure 7 and the descriptions that follow show how to implement the RACF legacy PassTicket generator algorithm.

(Figures 6 and 7 are unchanged and not included in this document.)

The flowcharts in Figure 8 and Figure 9 and the descriptions that follow show how to implement the RACF enhanced PassTicket generator algorithm.

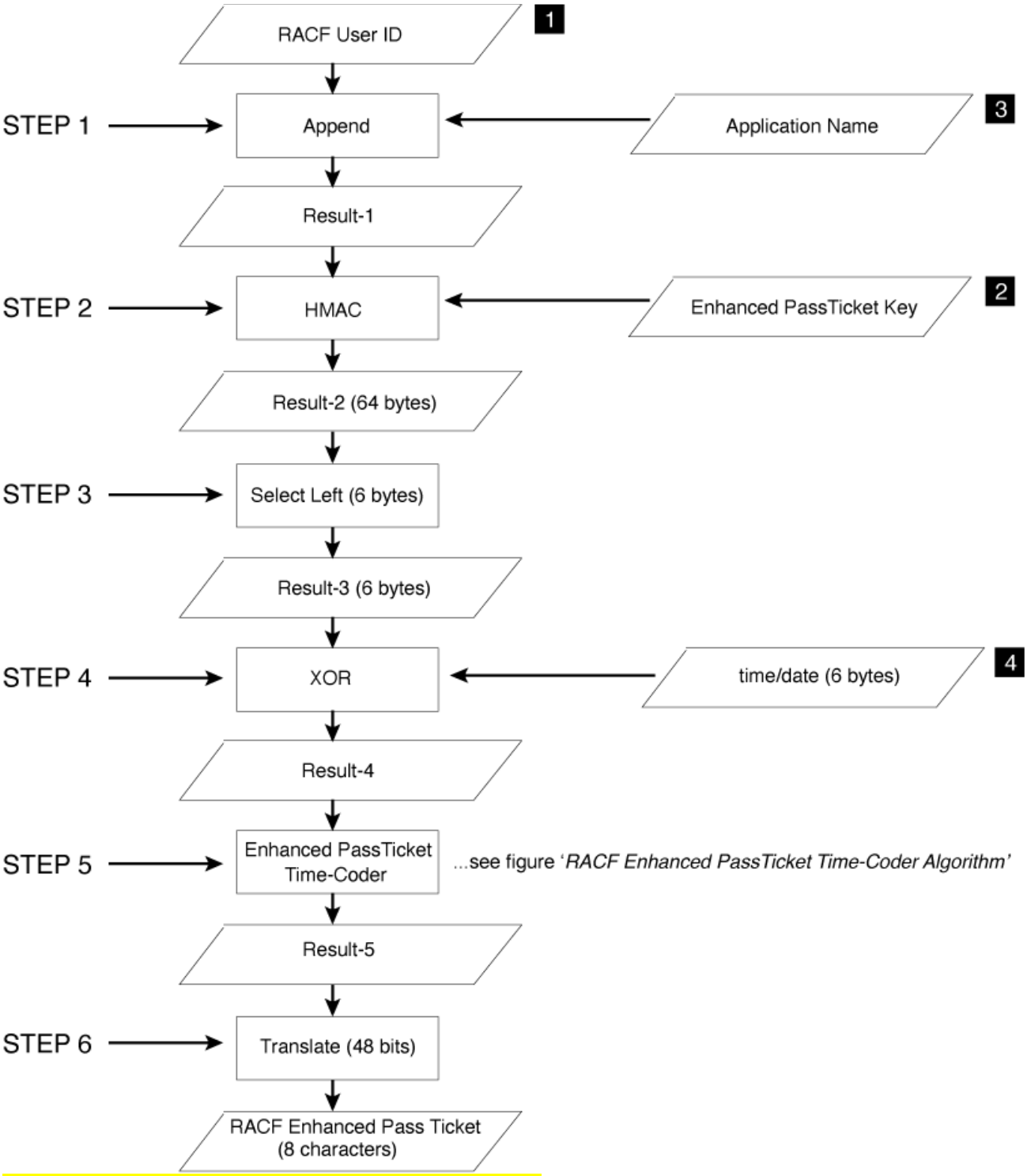

Figure 8. RACF enhanced PassTicket generator

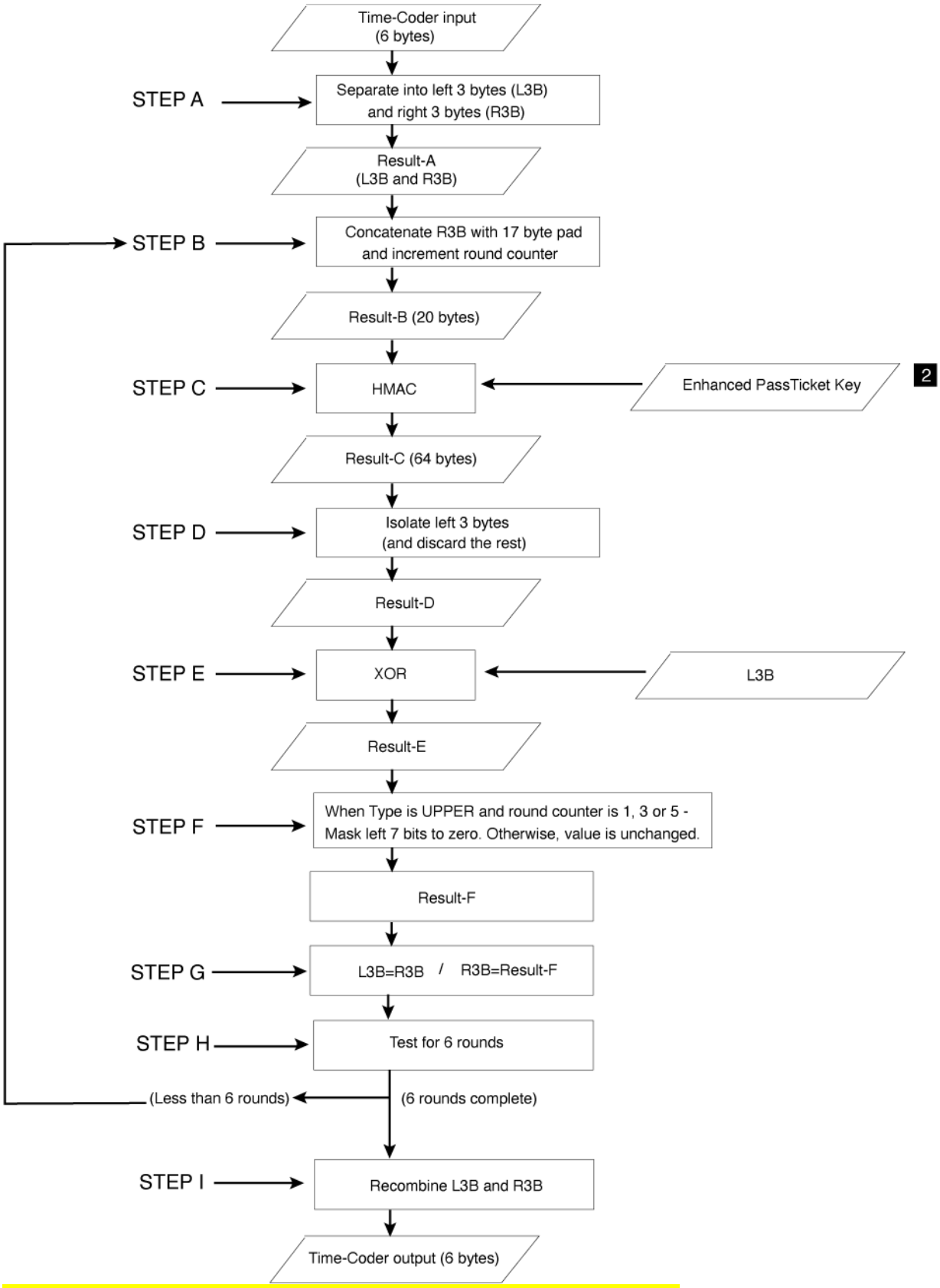

Figure 9. Algorithm for RACF enhanced PassTicket time-coder

Input data for the generator algorithms

To successfully use the PassTicket, the target application using RACF to identify and authenticate a user ID needs to have specific information for processing according to the algorithm. These are:

- A RACF host user ID
- The RACF PassTicket application key
- The application name
- Time and date information
- The PassTicket algorithm type

1. **The RACF user ID:**
   - Identifies the user ID on the system on which the target application runs
   - Is represented in EBCDIC
   - Is left-justified and padded with blanks on the right to a length of 8 bytes
2. **The RACF PassTicket application key:**
   - Must match the key value used when defining the application to the PTKTDATA class to RACF
   - For the legacy PassTicket algorithm:
     - This is a DES secret key.
     - Contains only the characters 0 though 9 and A though F
   - For the enhanced PassTicket algorithm:
     - This is an HMAC secret key.
3. **The application name** as defined for a particular application. You can use it to associate a PassTicket key with a particular host application. See z/OS Security Server RACF Security Administrator's Guide for information about determining application names.

   The name:
   - Is represented in EBCDIC
   - Is left-justified and padded with blanks on the right to a length of 8 bytes
4. **Time and date information**:

   This information:
   - For the legacy PassTicket algorithm:
     - Must be a 4-byte binary number
   - For the enhanced PassTicket algorithm:
     - Must be a 6-byte binary number
   - Shows how many seconds elapsed since January 1, 1970, at 0000 Greenwich Mean Time (GMT)

   Several programming languages support a function for representing time in this way. In C language, for example, you can obtain the time in this way:
   1. Declare the variable ts as **long**.
   2. Invoke the function **time(&ts)**.

   This produces the number of seconds that elapsed since January 1, 1970 at 0000 GMT, expressed as an unsigned long integer.

   Notes:

1. It is likely that the computer that authenticates the PassTicket is not the computer that generated it. To provide for differences in their internal clocks, the algorithms allow the generated time to be different ~~10 minutes on either side of the TOD clock of~~ than the computer that is evaluating the PassTicket. For legacy PassTickets the generated time must be within 10 minutes on either side of the TOD clock. For enhanced PassTickets, the amount of time skew is configurable in the PTKTDATA class profile.
2. For RACF to properly evaluate PassTickets, the TOD clock must be properly set to GMT rather than local time.

5. **The PassTicket algorithm type:**
   - Identifies the type of algorithm used to generate and evaluate the PassTicket.
   - The legacy PassTicket algorithm type is the original PassTicket algorithm and uses a DES secret key.
   - The enhanced PassTicket algorithm type is an improved PassTicket algorithm and uses an HMAC secret key. An enhanced PassTicket can be generated with either a MIXED or UPPER character set.

How the legacy PassTicket generator algorithm works

The RACF legacy PassTicket generator algorithm uses the input information to create a legacy PassTicket. By using cryptographic techniques, the algorithm ensures that each PassTicket is unpredictable.

The legacy PassTicket is an 8-character alphanumeric string that can contain the characters A through Z and 0 through 9. The actual legacy PassTicket depends on the input values.

…

(This rest of this section is unchanged and is not included in this document.)

How the legacy PassTicket time-coder algorithm works

The RACF legacy PassTicket time-coder algorithm uses the result of Step “4” of the legacy PassTicket generator algorithm. It creates the time-coder information and passes it back to step “6” on of that algorithm.

…

(This rest of this section is unchanged and is not included in this document.)

The legacy PassTicket permutation tables

A permutation table exists for each round of permutations that occurs during the legacy PassTicket time-coder process.

…

(This rest of this section is unchanged and is not included in this document.)

The legacy PassTicket translation process

The legacy PassTicket time-coder output produced by the process described in Figure 7 is translated into 8 alphanumeric characters in the following manner:

…

(This rest of this section is unchanged and is not included in this document.)

How the enhanced PassTicket generator algorithm works

(This section is new and not highlighted to improve readability.)

The RACF enhanced PassTicket generator algorithm uses the input information to create an enhanced PassTicket. By using cryptographic techniques, the algorithm ensures that each enhanced PassTicket is unpredictable. The enhanced PassTicket is an 8-character alphanumeric string which has a configurable character set. The PTKTDATA class profile can be configured to indicate the desired character set per application by using the TYPE keyword in the SSIGNON segment. The actual enhanced PassTicket depends on the input values.

The following steps describe this process:

Step 1

The RACF user ID 1 and application name 3 are appended together to produce Result-1.

Step 2

An HMAC with key 2 is performed on Result-1 to produce Result-2.
Note: All enhanced PassTicket cryptographic operations use HMAC with SHA-512 which produces 64 bytes of output.

Step 3

The left 6 bytes from Result-2 are selected as input to the next step as Result-3. The rest are discarded.

Step 4

Result-3 is XORed with the time and date information 4 to produce Result-4.

Step 5

Result-4 is passed to the enhanced PassTicket time-coder routine to produce Result-5.

Step 6

Result-5 from the time-coder routine is converted to an 8-character string called the enhanced PassTicket. Refer to “How the enhanced PassTicket character conversion works”.

How the enhanced PassTicket time-coder algorithm works

(This section is new and not highlighted to improve readability.)

The RACF enhanced PassTicket time-coder algorithm uses the Result-4 from Step “4” of the enhanced PassTicket generator algorithm. It creates the time-coder information Result-5 and passes it back to step “6” of that algorithm.
The following steps, which make up Step “5” of the enhanced PassTicket generator algorithm, describe this process:

Step A

Separate the 6-byte time-coder input (Result-4) into two portions, L3B (the left 3 bytes), and R3B (the right 3 bytes) to produce Result-A.

Step B

Concatenate R3B (the right 3 bytes from Result-A) with 17 bytes of padding bytes to form Result-B. In the resulting 20-byte string, the 3 bytes of R3B occupy the leftmost 3-byte positions.

The padding is a 17-byte string containing three separate fields:

1) The 1-byte round counter
   The round counter starts with the value 1 and is incremented by 1 on each subsequent use.
2) The 8-byte user ID 1
3) The 8-byte application name 3

Step C

An HMAC with key 2 is calculated on Result-B to produce Result-C.

Step D

The left 3 bytes from the Result-C are isolated and the rest of the value is discarded, producing Result-D.

Step E

Result-D is XORed with L3B (from Result-A) to produce Result-E.

Step F

An enhanced PassTicket type MIXED is encoded as a 48-bit value and a type UPPER is encoded as a 41-bit value. This step sets the extraneous leftmost 7 bits of a type UPPER to binary zero.

When the enhanced PassTicket type is UPPER and the round counter is 1, 3 or 5 the following masking operation is performed:

Perform bitwise AND on the leftmost 1 byte of Result-E with ‘01’x to set the leftmost 7 bits to zero to produce Result-F.

When the enhanced PassTicket type is MIXED or the type is UPPER and the round counter is 2, 4 or 6:

Result-E is set as Result-F without any changes.

Step G

The values of L3B and R3B are redefined:

1. L3B is set equal to R3B.
2. R3B is set equal to Result-F.

Step H

This step counts the number of time-coder rounds that have been completed.
If the value is less than 6, the time-coder returns to Step B for another round.
If 6 rounds have been completed, processing continues with the next step.

Step I

L3B (left 3 bytes) and R3B (right 3 bytes) are recombined into a 48-bit string. This completes the time-coder processing and produces Result-5. This result is passed back to the generator algorithm as input to Step “6” for translation.

The enhanced PassTicket translation table:

(This section is new and not highlighted to improve readability.)

The enhanced PassTicket translation table consists of 64 slots. The first ten slots are occupied by the numerics: 0–9. The next 26 slots are occupied by the uppercase letters of the alphabet: A–Z. The next 26 slots are occupied by the lowercase letters of the alphabet:

a–z. The last two slots are occupied by the special characters: dash “-“ and underscore “_”.

Note: An enhanced PassTicket with type UPPER will only use the first 36 slots (0-35) of this translation table.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
0 1 2 3 4 5 6 7 8 9  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P

26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 Q  R  S  T  U  V  W  X  Y  Z  a  b  c  d  e  f  g  h  i  j  k  l

48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
 m  n  o  p  q  r  s  t  u  v  w  x  y  z  -  _
```

How the enhanced PassTicket character conversion works

(This section is new and not highlighted to improve readability.)

The RACF enhanced PassTicket time-coder output is converted to an EBCDIC string value using the following process:

Step A

Copy the 6-byte time-coder output value from Result-5 to the rightmost 6 bytes of a 64-bit binary value to produce Result-A. The leftmost 2 bytes of Result-A are set to binary zero.

Step B

For enhanced PassTicket type UPPER:

- Calculate modulo 36 of Result-A to produce Result-B.

For enhanced PassTicket type MIXED:

- Calculate modulo 64 of Result-A to produce Result-B.

Step C

Translate Result-B from a binary value to an EBCDIC value using the enhanced PassTicket type translation table to produce Result-C.

For example, the binary value 33 is translated to the EBCDIC value ‘X’.

Step D

Result-C is set as an individual character of the EBCDIC enhanced PassTicket value. The characters are concatenated together one at a time starting with the rightmost character and proceeding to the left on each round of conversion.

Step E

When less than 8 characters have been converted:

- For enhanced PassTicket type UPPER:
  - Divide Result-A by 36 to produce Result-E.
- For enhanced PassTicket type MIXED:
  - Divide Result-A by 64 to produce Result-E.

Step F

Replace Result-A with Result-E.

Step G

This step counts the number of characters that have been encoded. When there are less than 8 characters encoded the conversion process returns to **Step B** for another round.

Step H

The final enhanced PassTicket value has been assembled.

Generating a secured signon session key

Note:

1. IBM recommends that the secured signon session key not be used outside of a test environment. It is no longer considered secure. This section is left for reference only.
2. Enhanced PassTickets and enhanced PassTicket keys cannot be used to generate a secured signon session key.
   An attempt to generate a secured signon session key with a specified enhanced PassTicket value may fail with Return Code 4 – “Incorrect PassTicket”
   An attempt to generate a secured signon session key with a PTKTDATA class profile that contains only an enhanced PassTicket key may fail with Return Code 24 – “Error in the session key generator process”.

…

(The remainder of this section is unchanged and not included in this document.)

3.5 z/OS Security Server RACF Data Areas

This information supplements the following chapter and section:

- Chapter: '*RACF Data Areas*'
  - Section: RCVT: RACF Communication Vector Table

3.5.1 RCVT: RACF Communication Vector Table

The RCVT: RACF Communication Vector Table adds a field to indicate that the enhanced PassTicket Functions are available. Other products can check this field to determine if the current version of RACF has enhanced PassTicket support added either in the base OS or via PTF.

Offset (dec)	Offset (Hex)	Type	Len	Name(Dim)	Description
...					
640	280	BITSTRING	1	RCVTFLG4	Function availability bits
...					
	...1			RCVTEPT	Enhanced PassTicket Functions (OA59196) are available.
...					

3.6 z/OS Security Server RACF Messages and Codes

This information supplements the following chapter and section:

- Chapter: 'IRR messages for commands, utilities, and other tasks'
  - Section: 'Dynamic parse (IRRDPI00) messages'

3.6.1 Dynamic parse (IRRDPI00) messages

The message IRR52218I explanation is updated to indicate it can be issued for other segments besides CSDATA.
The message IRR52256I is added.

IRR52218I The value specified for *keyword-name* is not valid. The { maximum value | minimum value | maximum length } allowed is *limit*.
Explanation
The value specified for *keyword-name* ~~in the CSDATA segment~~ does not fall between the minimum and maximum values allowed for the keyword, or has an incorrect length.

For fields in the CSDATA segment the maximum and minimum values allowed, and the maximum length of the value are set using custom field definitions in the CFDEF segment of the CFIELD class. For more information on custom fields, see the *z/OS Security Server RACF Security Administrator's Guide*.
System action
Command processing stops.
User response
You must reissue the command and specify a value that is either less than the maximum value or greater than the minimum value specified by *limit*.
Parent topic: Dynamic parse (IRRDPI00) messages

IRR52256I *keyword-name* is an unsupported keyword. Command processing is terminated

Explanation

The value specified for *keyword-name* is not supported by the command.

System action

Command processing stops.

User response

You must reissue the command with only supported keywords.

Parent topic: Dynamic parse (IRRDPI00) messages

4 Trademarks

IBM®, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.