Introduction and Publication Updates for OA56622: z/OS DFSMS Data Set Encryption for Basic and Large Format Data Sets V2R3 and above

Document Name:	OA56622.pdf
Document Owner:	Cecilia Carranza Lewis (carranc@us.ibm.com)
Version:	V1.1

About this information

This document provides an introduction for z/OS Data Set Encryption for Basic and Large Format Data Sets support available with APAR OA56622, in addition to specific updates for certain publications in the z/OS® product library, as required by the APAR OA56622 package. The information consists of pages excerpted from the respective publications or new information added to the respective publications. The information in this document applies to both V2R3 and V2R4, unless otherwise noted.

Currency of this information: The complete publication updates appear in the next editions of the official publications, which are scheduled to be published in October 2020. Thereafter, the information in the official publications supersedes the publication information in this APAR document. If you are reading this APAR document after October 2020, refer instead to the official publications for the most recent information.

Contents

1 Introduction

This new function enhances the data set encryption support by providing the ability to encrypt sequential basic and large format SMS-managed data sets using BSAM, QSAM and EXCP.

Applications using standard BSAM and QSAM APIs require no, or minimal changes, to access encrypted basic and large format data sets.

Applications using the EXCP, EXCPVR or XDAP macros require changes to access encrypted basic and large format data sets by performing encryption and decryption of data. A new access method encryption macro, IGGENC, is provided to simplify encryption and decryption of data while ensuring compatibility with the access methods. (Note: All references to EXCP apply equally to the EXCPVR and XDAP macros, unless stated otherwise.)

This support provides the ability to define a sequential basic or large format SMS-managed data set as an encrypted data set. (Note that sequential basic and large format data sets are DASD only).

As with encrypted extended format data sets, data set encryption for sequential basic and large format data sets allow you to:

- Supply a key label via the same allocation sources (see below).
- Obtain data set level encryption information from the same system services, displays and interfaces.
- Open an encrypted sequential basic or large format data set for reading or writing while the system requires the user to have SAF authority to both the data set and the key label.
- Have the data remain encrypted during DFSMS backup, migration and replication without allowing the data administrator to have read or write authority to the unencrypted data in the data set.

Also introduced with this support is a new ACS read-only variable, &DSKEYLBL, that can be used within ACS routines to display the key label specified through JCL, dynamic allocation, IDCAMS DEFINE and DFSMSdss/DFSMShsm target data set allocation for sequential extended format, basic format and large format data sets as well as VSAM extended format data sets.

Note: It is important to emphasize that investigation is needed to identify basic and large format data sets that are eligible for data set encryption. Restrictions should be evaluated as well as the applications that access the data sets. EXCP applications require changes in order to support encrypted basic and large format data sets. An attempt to open an encrypted basic or large format data set from an EXCP application that has not been modified will result in an ABEND213-9A and message IEC143I. Refer to 3.5.4 New “Candidates for encrypted basic and large format data sets” Section *on page 8 for information that may help with evaluating candidates. This support provides the interfaces needed by an EXCP application to access encrypted basic and large format data sets. IBM programs and ISV programs that use EXCP and choose to support encrypted basic and large format data sets must modify their EXCP applications accordingly.*

2 Using Encrypted Basic or Large Format Data Sets

2.1 Requirements

Similar to encrypted extended format data sets, exploitation of this function also requires the following minimum hardware:

- IBM® zEnterprise BC12 (zBC12) or later
- Crypto Express3 Coprocessor or later
- Feature 3863, CP Assist for Cryptographic Functions (CPACF)

In addition, ICSF must be installed and configured with a CKDS and AES master key loaded.

Note: The minimum software release that can support encrypted basic and large format data sets is z/OS V2R3 with OA56622. An attempt to access an encrypted basic and large format data set on a z/OS V2R2 system without OA60160 installed or on a lower release will have unpredictable results. With APAR OA60160 installed on a z/OS V2R2, an attempt to open an encrypted basic or large format data set will result in an expected ABEND0C1 with associated message IEC999I.

The DFSMS APAR OA56622 pulls in the pre-req/co-req APAR(s) for this support; however, depending on the level of service already on a system, additional service can be required. SMP/E APPLY or REPORT MISSINGFIX can be used to identify service needed for the fix category IBM.Function.DataSetEncryption. For more information on fix categories, see IBM Fix Category Values and Descriptions. https://www-01.ibm.com/support/docview.wss?uid=isg3T1027683.

Ensure that all systems are at the minimum hardware and software levels before encrypting any basic and large format data sets.

2.2 Enablement

The enablement considerations identified for encrypted extended format data sets apply also to encrypted basic and large format data sets. (Refer to section “Data Set Encryption” in *z/OS DFSMS Using Data Sets* for the enablement considerations.) The following describes an ***additional*** enablement consideration associated with encrypted basic and large format data sets.

To indicate that the specification of a key label should result in the creation of an encrypted basic or large format data set (that is, indicate if the basic and large format data set type should be regarded as a supported data set type for data set encryption), the data set must be SMS-managed and the following resource in the RACF FACILITY class must be defined:

STGADMIN.SMS.ALLOW.DATASET.SEQ.ENCRYPT

The system checks whether this resource is defined when the data set is first allocated (created).

As with other supported data set types for data set encryption, a new basic or large format data set gets the encrypted attribute if a key label is supplied in any of these three manners:

- The DFP segment of the RACF data set profile specifies a key label, or
- the DD statement, dynamic allocation equivalent or IDCAMS DEFINE for a new data set has the DSKEYLBL option(*), or
- the data class has a key label for DASD(*).

(*) Note that in these cases, at least READ authority to the FACILITY class resource **STGADMIN.SMS.ALLOW.DATASET.ENCRYPT** is also required.

The system uses the first key label that is in the above list.

You cannot add the encrypted attribute to an existing data set but you can use IDCAMS REPRO, IEBGENER, ICEGENER or a similar copying program to copy the data set into a new encrypted data set. (Note that DFSMSdss COPY performs physical copies thus cannot be used to copy a non-encrypted data set into a new encrypted data set.)

2.3 Enhanced Physical Format of an Encrypted Sequential Basic or Large Format Data Set

The access method adds an 8-byte prefix to each physical block of the data set on output. Each block prefix will contain a value that uniquely identifies the block. The content of this prefix is relevant only to application programs that use EXCP. The prefix will not be included in the physical block size of the data set stored in the block size fields in the DSCB, DCB, [JFCB], DCOLLECT and SMF records. You must take the length of the block prefixes into consideration when requesting DASD space.

- The prefix will not be allowed on WRITE or PUT requests, nor returned on READ and GET requests.

- The system-determined block size algorithm will take the block prefix length into consideration.

Note: In rare cases, there may be encrypted basic and large format data sets which are created without a block prefix. Such a data set can only be opened for EXCP. A flag in the catalog encryption cell will indicate whether the data set has prefixes. After open, the ISITMGD macro can be used to determine the length of the prefix.

2.4 Processing Considerations Associated with Data Set Encryption for Sequential Basic and Large Format Data Sets

The processing considerations identified for encrypted extended format data sets apply also to encrypted basic and large format data sets. (Refer to the Data Set Encryption section in *z/OS DFSMS Using Data Sets* for the processing considerations.) The following are *additional* processing considerations associated with encrypted basic and large format data sets.

- For applications using BSAM or QSAM APIs, typically no application changes are required to access the data. The data will be encrypted by the access method on output requests and decrypted by the access method on input requests. However, an application that has calculations to derive an optimal block size may want to take into account the 8-byte prefix.
- For applications using channel programs with EXCP, EXCPVR or XDAP, application changes are required to access the data. In this case, the access methods do not get control and therefore cannot process the data on behalf of the caller. To support such applications, EXCP applications use a new access method macro, IGGENC, to perform data encryption and decryption for encrypted basic and large format data sets.

3 Publication Updates

3.1 *z/OS Introduction and Release Guide (GA32-0887)*

The following updates to be added under section “DFSMS data set encryption enhancements for z/OS V2R4”.

Add the following bullet to the list under “z/OS DFSMS data set encryption can be used to encrypt the following types of data sets”:

- Sequential basic and large format data sets that are accessed through BSAM, QSAM and EXCP.

Add an additional **Note:** under Coexistence requirements:

Note: The minimum software release that can support encrypted basic and large format data sets is z/OS V2R3 with OA56622. An attempt to access an encrypted basic and large format data set on a z/OS V2R2 system without OA60160 installed or on a lower release will have unpredictable results. With APAR OA60160 installed on z/OS V2R2, an attempt to open an encrypted basic or large format data set will result in an expected ABEND0C1 with associated message IEC999I.

3.2 *z/OS DFSMS Using New Functions (SC23-6857)*

Refer to the changes under 3.5 *z/OS DFSMS Using Data Sets (SC23-8655)* on page 6 which describe the updates to be duplicated in section “Data set encryption”.

3.3 *z/OS DFSMS Introduction (SC23-6851)*

See Chapter 4, “Managing Data with DFSMSdfp”.

In section “DFSMSdfp Data Set Organizations”, under heading “Large Format Data Sets”, add:

> You can allocate a large format data set as encrypted, which enables you to protect sensitive data. A large format data set is encrypted when it is first allocated with an encryption key label. The access method performs encryption and decryption. The application program does not need any logic to handle it.

In section “DFSMSdfp Data Set Organizations”, under heading “Basic format data sets”, add:

> You can allocate a basic format data set as encrypted, which enables you to protect sensitive data. A basic format data set is encrypted when it is first allocated with an encryption key label. The access method performs encryption and decryption. The application program does not need any logic to handle it.

3.4 *z/OS DFSMSdfp Storage Administration (SC23-6860)*

New FACILITY class resource

In Section “Command and keyword related profiles”, the following new profile will be added to the list of profiles:

> **STGADMIN.SMS.ALLOW.DATASET.SEQ.ENCRYPT**
>
> Controls the ability to allow basic and large format data sets to be treated by the system as a supported data set type for data set encryption.

New Read-only variable

In Section “Read-only variables”, the following is added to Table 24. Read-Only Variables:

Name	Description
&DSKEYLBL	The data set key label whose value is from the following precedence orders: • From RACF DFP data set profile. • From JCL, IDCAMS DEFINE, dynamic allocation • From data class Type: Literal Max Value: 64 characters

3.5 *z/OS DFSMS Using Data Sets (SC23-8655)*

3.5.1 “Data Set Encryption” Section

The section “Data Set Encryption” will update the list of supported data set types to include:

- sequential basic format data sets (SMS managed only)
- sequential large format data sets (SMS managed only)

3.5.2 “Coexistence” Section

The following is added under “Coexistence requirements”:

> On a z/OS V2R3 or z/OS V2R4 system with OA56622 or on a later system, you can create and access encrypted basic and large format data sets.
>
> **Note:** The minimum software release that can support encrypted basic and large format data sets is z/OS V2R3 with OA56622. An attempt to access an encrypted basic or large format data set on a z/OS V2R2 system without OA60160 or a lower system, will have unpredictable results. With OA60160 on a z/OS V2R2 system, an attempt to access an

encrypted basic or large format data set will result in an expected ABEND0C1. Ensure that all systems are at the minimum hardware and software levels before encrypting any data sets.

3.5.3 "Specifying a key label for a non-extended format data set" Section

The following is added under "Specifying a key label for a non-extended format data set"

> If an encryption key label is specified for a DASD data set that is not extended format and the FACILITY class resource STGADMIN.SMS.ALLOW.SEQ.ENCRYPT is not defined, the key label is ignored and the data set is successfully created as non-encrypted non-extended format data set. In addition, SMS message IGD17156I is issued (or if using IDCAMS DEFINE and data set is non-SMS managed, message IDC3040I is issued) indicating that the key label is ignored. Instead to have the system fail the allocation, the user must have at least READ authority to the resource in the FACILITY class: STGADMIN.SMS.FAIL.INVALID.DSNTYPE.ENC
>
> If this facility class resource exists and the user has at least read authority, SMS will fail the allocation and issue message IGD17151I (or if using IDCAMS DEFINE and non-SMS managed data set request, message IDC3039I is issued).

Encrypted data sets must be SMS-managed. Encrypted extended format data sets can also be compressed format.

With z/OS V2R3 and z/OS V2R4 systems (with OA56622) or higher levels, data set encryption supports basic and large format data sets. To indicate that the specification of a key label should result in the creation of an encrypted basic or large format data set (that is, indicate if the basic and large format datasets type should be regarded as a supported data set type for data set encryption), the data set must be SMS-managed and the following resource in the RACF FACILITY class must be defined: STGADMIN.SMS.ALLOW.DATASET.SEQ.ENCRYPT.

Note: Restrictions and considerations must be evaluated before enabling encryption of basic and large format data sets.

Applications using standard BSAM and QSAM APIs require no, or minimal changes, to access encrypted basic and large format data sets. Applications using EXCP require changes to access encrypted basic and large format data sets by performing encryption and decryption of data. The IGGENC macro can be used to simplify encryption and decryption of data while ensuring compatibility with the access methods. (Note: All references to EXCP apply equally to the EXCPVR and XDAP macros, unless otherwise stated.

The restrictions identified under *Restrictions for encrypted data sets* also apply to encrypted basic and large format data sets. The following are ***additional*** restrictions associated with encrypted basic and large format data sets to be added:

- The minimum size of any user block written to an encrypted data set is 16 bytes. Using BSAM or QSAM, an attempt to write a block less than 16 bytes will fail with ABEND 002-F2.
- The minimum DCB LRECL is 16 bytes for RECFM F(B(S)) and 12 bytes for RECFM V(B(S)). An attempt to open a data set with LRECL less than the minimum supported will fail with ABEND 213-89.
- Encrypted basic and large format data sets do not support hardware keys (DCBKEYLE). An attempt to open a data set with a non-zero key length will fail with 213-99. Diagnostic information in the message will clearly identify the reason for this failure.

- BDAM does not support processing encrypted basic and large format data sets. An attempt to open an encrypted basic or large format data set with BDAM will fail with 213-99. Diagnostic information in the message will clearly identify the reason for this failure.
- Checkpoint/Restart does not support open encrypted basic or large format data sets. Also, checkpoint data sets cannot be encrypted. An attempt to take a checkpoint when an encrypted basic or large format data set is open or when the checkpoint data set is encrypted will fail with return code 08 and reason code 127.

3.5.4 New "Candidates for encrypted basic and large format data sets" Section

SMF records can be used to view activity over a period of time. This information can be valuable when evaluating which basic and large format data sets may be eligible for converting to encrypted basic and large format data sets. DCOLLECT records may also be used for some of this determination. This section describes fields within SMF Type 14, SMF Type 15, and SMF Type 42-6 records that can help you identify whether your data sets would be candidates for data set encryption based on the list of restrictions for encrypted basic and large format data sets.

- SMF Type 14 and SMF Type 15 records are mapped by macro IFGSMF14.
- SMF Type 42-6 records are mapped by macro IGWSMF.

First you must determine that the data set is a basic or large format data set. You can do this by testing a field in the JFCB which is found in SMFJFCB1. Test JFCDSORG for bit JFCORGPS being on and test for bit SMF14STR being off. This combination indicates the data set is a sequential data set but not extended format.

To determine the application that accesses the data set, you can use these fields in the SMF 14 and 15 records

- SMF14JBN for the jobname
- SMFJOBID for the job identifier
- SMF14SPN for the step name
- SMF14PGN for the program name

To determine the application that accesses the data set, you can use these fields in the SMF 42-6 records

- S42JDJNM for the jobname

To determine if your data set is processed by EXCP:

- Scan your programs for use of EXCP, EXCPVR or XDAP macros
- Review SMF Type 14 and SMF Type 15 records.
    - You can use SMFDCBMF to identify data sets opened with EXCP. This is a 2 byte field which is a copy of DCBMACRF, where the high order bit indicates MACRF=E. You can find DCBMACRF in the DCBD mapping macro.
    - You can then also use SMF14EXCPBAM bit to identify data sets opened with BSAM or QSAM, but then accessed via EXCP.
- Review SMF 42-6 records.
    - You can use S42DSEXC to identify data sets opened with EXCP.
- Note: Use the z Batch Network Analyzer tool to identify data sets accessed by EXCP, as it uses the SMF records to make this determination.

- To determine if your data set is accessed by an EXCP application that can support encrypted data sets
    - You can use NEW flag SMF14DSENCRYPTOK to identify whether the application program used EXCP and the program is enabled to handle data set encryption for basic and large format data sets. This does not imply that the data set is DASD.
        - Note: The system detects this condition by testing DCBE DSENCRYPT=OK is specified regardless of the data set type.

3.5.5 New "Determine if your data sets meet the restrictions for encrypted basic/large format data sets" Section

To determine if your data set meets the minimum block size requirement

- You can use the SMFJFCB1 field to find the block size of the data set. Using the JFCB mapping macro, IEFJFCBN, test field JFCBLKSI to determine if the block size is at least the minimum required for encryption, which is 16 bytes.
- You can use S42DSBSZ in SMF 42-6 for block size

To determine if your data set meets the minimum record length requirement

- You can use the SMFJFCB1 field to find the record length of the data set. Using the JFCB mapping macro, IEFJFCBN, test field JFCLRECL to determine if the record length is at least the minimum required for encryption, which is 16 bytes for record format F(B(S)) and 12 bytes for record format V(B(S)). Use SMFDCBRF to determine the record format, which is mapped the same as DCBRECFM.
- You can also use the following to view the record format, block size and record length
    - TSO LISTDSI command for REXX
    - IEHLIST LISTVTOC
    - ISPF option 3.2

To determine if your data set uses hardware keys (DCBKEYLE).

- You can use the SMFJFCB1 field to determine if hardware keys are used with the data set. Using the JFCB mapping macro, IEFJFCBN, test field JFCKEYLE for zeros to ensure no hardware keys are in use.

To determine if your data set is accessed by BDAM.

- You can use the SMF14DDA bit to determine if BDAM has been used to access the data set. If the flag is on, this data set is not a candidate for encryption.

The SMF records indicate processing based on a span of time. You can use the IDCAMS DCOLLECT command to create records for data sets that might not have been accessed recently. These records help with analysis of the attributes of data sets, but do not show how the data sets were used. The mapping macro for DCOLLECT records is IDCDOUT.

- To determine the block size, you can use DCDBKLNG
- To determine the record length, you can use DCDLRECL
- To determine the record format, you can use DCDRECRD. This is a copy of DCBRECFM.
- To determine the data set organization, you can use DCDDSORG.

3.6 *z/OS DFSMS Macro Instructions for Data Sets (SC23-6852)*

3.6.1 DCBE Macro

The following new parameter is added to the DCBE macro instruction.

DCBE

…

[,DSENCRYPT={OK|NOTOK}]

…

Description:

DSENCRYPT={OK|NOTOK}

Specifies the support for EXCP access of encrypted data sets.

DSENCRYPT=OK allows you to specify that your EXCP application program supports how to process encrypted data sets according to BSAM and QSAM requirements. The specification of this will resolve to the DCBEDSENCRYPTOK indicator in the DCBE to be set on. This option is needed when processing an encrypted data set in any either of these two situations:

- The DCB indicates MACRF=E (the DCB is only for EXCP). OPEN tests this bit. If you do not code this option, the OPEN function will issue an ABEND.
- The DCB is for BSAM or QSAM but the application program issues EXCP. The system will test this bit for each EXCP. The user program must also set on the IOBSPSVC bit in the IOB. It informs the access method not to examine this IOB. (Note: Refer to IOBSPSVC in publication *DFSMSdfp Advanced Services*.) If you do not code this option, the EXCP will fail.

Note that the DCBEDSENCRYPTOK indicator means that the system acknowledges that you have specified DSENCRYPT=OK. It does not mean that the data set is encrypted or that the application program issues EXCP, EXCPVR or XDAP.

DSENCRYPT=NOTOK indicates that the EXCP application does not support encrypted data sets. The specification of this will resolve to the DCBEDSENCRYPTOK indicator in the DCBE to be set off. This is the default.

3.6.2 ISITMGD Macro

The ISITMGD macro allows you to learn certain attributes of an open data set. The ISITMGD macro sets some bits in the ISITMGD parameter list which you can test to get information about the data set. Existing flag ISMENCRP indicates the data set is an encrypted data set. The description will be modified to indicate that it will also apply for encrypted basic and large format data sets.

The IGWCISM macro maps the ISITMGD parameter list. Currently it is 28 bytes long.

A new field will be defined to indicate the block prefix length, if it exists for the data set.

To accommodate this new field (and allow for future fields), the length of the parameter list is expanded to 40 bytes. For compatibility, you must specify the new VERSION keyword to request the longer length as VERSION=2 in order to obtain the new prefix length field. If you specify VERSION=1 or omit the VERSION keyword, you will get the original 28-byte parameter list.

ISITMGD

…

[,VERSION={1|2}]

…

Description:

VERSION={1|2}

Specifies the version of the parameter list as mapped by IGWCISM.

1 Requests the 28-byte parameter list. This is the default.

2 Requests the 40-byte parameter list.

The new VERSION keyword can be specified on the standard, list and execute forms of the ISITMGD macro.

Use VERSION=2 to obtain the 40-byte parameter list which will contain the following new field to be set by the system and can be tested after the DCB is open.

ISMBPRFX

A one-byte binary field that contains the length of the prefix associated with each physical block in the data set. For encrypted basic and large format sequential data sets, a value of 8 will be returned. However, for special case encrypted basic and large format sequential data sets that do not contain block prefixes, a value of 0 will be returned. For other data set types, a value of 0 will be returned.

3.6.3 IHADCBE

In section **Data control block extension (DCBE),** the following new field is added

Offset	Length	Name	Description
21(15)	1	DCBEFLG5	Flag set by user
	.... .1..	DCBEDSENCRYPTOK	DSENCRYPT=OK specified. Indicates the EXCP application supports data set encryption.

3.7 *z/OS DFSMS Managing Catalogs* (SC23-6853)

3.7.1 ENCNPBLK and ENCRYPTA Fields

Update Chapter 11. Catalog Search Interface User's Guide in the section “Field Name Directory”. See Table 23. Catalog Field Names. Add ENCNPBLK (above ENCRYPTF) and Update ENCRYPTA as follows:

Rep	Type	Length	Name	Description
No	Binary	1	ENCNPBLK	Blocks do not have prefixes. Set only for basic and large format data sets X’00’ – Encrypted with block prefixes X’01’ – Encrypted with no block prefixes X’FF’ – Not encrypted

Rep	Type	Length	Name	Description
No	Mixed	96	ENCRYPTA	All of the encryption fields as one field. It returns 96 bytes of information as formatted in the encryption cell: • 2 bytes for the encryption type • 64 bytes for key label • 8 bytes for first half of the saved ICV (Initial chain vector) • 1 byte for the encryption mode • 16 bytes for a verification value • 5 bytes reserved If the data set is not encrypted, 96 bytes of X'FF' are returned.

3.7.2 IGGCSIF Mapping Macro

This section will be added after the above large table.

A mapping macro, IGGCSIF, is available in SYS1.MACLIB to map the ENCRYPTA field. It has a required positional parameter. You must code "ENCRYPTA" without the quotation marks. Optionally you can enclose it in a pair of parentheses.

All parameters can be in mixed upper and lower case.

IGGCSIF supports two optional keyword parameters:

DSECT={YES|NO}

Whether the macro should generate a DSECT statement before the fields. The default is YES. It can be mixed case. If you code DSECT=NO, you can embed this macro invocation in a CSECT or DSECT.

PREFIX={*xxx*|CSIF}

All symbols defined by the macro expansion will begin with these characters. Do not code delimiters. The default is CSIF. This allows you to call the macro more than once in one program. Although ENCRYPTA is called a "field", it consists of these fields:

Offset	Type	Len.	Name	Description
0(0)	DSECT	(96)	CSIFNALL	Encryption block prefix in basic or large format sequential data set
0(0)	Bits	2	CSIFTYPE	Encryption type
0(0)	Integer	1	CSIFTYPA	Encryption algorithm X'01': AES
1(1)	Integer	1	CSIFTYPKL	Encryption key length code. X'00': 256-bit
2(2)	Character	64	CSIFKEYL	Key label (DSKEYLBL on DD statement)
66(42)	Integer	8	CSIFNICV	High order half of ICV value, initial chain vector
74(4A)	Integer	1	CSIFMODE	Encryption mode (X'02' FOR XTS). It is copied from the BCFZCRYP macro.
75(4B)	Bits	16	CSFIVERIFY	Verification bytes
91(5B)	Bits	1	CSFIFLAG	Verification flags
	1...		CSFIVERSET	Verification bytes set
	.1..		CSFIVERSETV1	Verification bytes are version 1
92(5C)		1	CSFIFLAG2	Verification flags
	1...		CSFIDSENCNP	Blocks do not have prefixes. Set only for basic and large format data sets.

Offset	Type	Len.	Name	Description
93(5D)	Bits	3		Reserved.

3.8 *z/OS DFSMSdss Storage Administration*

In section “Managing availability with DFSMSdss”, the following will be added to the Note:

- Encryption eligible sequential basic and/or large format data sets cannot be used as output of the DUMP command.

In section ‘Securing your tape backups’, the following bullet will be added to the "Observe the following considerations":

- Encryption eligible sequential basic and/or large format data sets cannot be used as output of the DUMP command.

In chapter 18. Syntax – DFSMSdss function commands, the following paragraph will be added to section "COPY Command for DFSMSdss":

> "DFSMSdss COPY will always preserve data set encryption attributes of the source data set. Therefore, new allocations will be defined with the source encryption attribute. If a pre-allocated target data set is encountered, it must also be encrypted to be considered a usable target data set. The usable preallocated target will be overwritten with the source encryption attributes (which includes the key label)."

In chapter 18. Syntax – DFSMSdss function commands, the following paragraph will be added to section

"RESTORE Command for DFSMSdss":

> "DFSMSdss RESTORE will always preserve data set encryption attributes of the source data set. Therefore, new allocations will be defined with the source encryption attribute. If a pre-allocated target data set is encountered, it must also be encrypted to be considered a usable target data set. The usable preallocated target will be overwritten with the source encryption attributes (which includes the key label)."

In chapter 25. Data set attributes, Table 41. Data Set Attributes and How They Are Determined will be updated to include the following information:

- Attribute: zOS data set encryption
- …the source data set? YES
- ...the Pre-Allocated Target? NO - Must be usable target with like encryption eligibility.
- ...a Keyword? NO
- ...another Factor? NO

3.9 *z/OS MVS System Management Facilities (SMF)* (SA38-0667)

Two bits have been defined in the type 14 and 15 record. The system writes a type 14 record when the data set is open for input. The system writes a type 15 record for any other OPEN option. Both record types have the same format. New SMF bit, SMF14DSENCNP, indicates that the encrypted basic or large format data set has no prefixes in its blocks. It is defined as a bit within existing flag byte SMF14DEF.

Offset	Name	Len	Format	Description
4	SMF14DEF	1	Bits	Flag byte
	SMF14DSENCNP		..1.	Blocks do not have prefixes. Set only for basic and large format.

New SMF bit, SMF14DSENCRYPTOK, indicates that the application can support encrypted basic or large format data sets and has specified DCBE DSENCRYPT=OK. This byte is in the "additional data set characteristics" section in the extended information segment.

Offset	Byte	Len	Format	Description
10	SMF14FLG1	1		Meaning when set
	SMF14DSENCRYPTOK		.... 1...	The application program is enabled for basic and large format data set encryption with EXCP. The application coded DSENCRYPT=OK on the DCBE macro.

3.10 z/OS DFSMSdfp Checkpoint/Restart (SC26-6862)

Checkpoint/Restart will not support encrypted basic or large format data sets. Checkpoint will fail if it finds an open encrypted basic or large format data set or if the checkpoint data set is encrypted. Section 'Reason Codes' will be updated to add the following return code and reason code :

- Reason Code: 127
- Return Code: 08
- Meaning: A CHKPT request was made by a task that was open to an encrypted sequential basic or large format data set, or an encrypted sequential basic or large format data set was specified as a checkpoint data set. Programmer Response: Ensure that checkpoint is not issued while this condition exists or correct this condition.

3.11 *z/OS MVS System Messages, Vol 1 (ABA-AOM) (SA38-0668)*

3.11.1 ADR374E

ADR374E (*ttt*)-*mmmmm*(*yy*), UNABLE TO OPEN DDNAME *ddname*, *reason_code return_code*

The following will be added to the list for reason code 14:

- Encryption eligible basic or large format dump data sets are not supported.

3.12 *z/OS MVS System Messages, Vol 7 (IEB-IEE) (SA38-0674)*

3.12.1 IEC036I

IEC036I 002-rc,mod,jjj,sss, ddname[-#],dev,volser,dsname(member)

The following existing reason code introduced with data set encryption for extended format data sets applies also to data set encryption for basic and large format data sets:

F2 On the first write after opening a new encrypted non-compressed extended format sequential data set for output with BSAM, the system detected that the block size derived for the data set by taking the maximum of BLKSIZE at OPEN and BLKSIZE at first WRITE is less than 16 bytes. However, the minimum block size supported for encrypted non-compressed sequential data sets is 16 bytes.

The following new reason code is added for encryption of basic and large format data sets:

46 An error returned from common encryption routine IGGENCAM

47 An error returned from encryption service (BCFXCRYP)

48 An EXCP was issued for an encrypted basic or large format data set which was opened with a BSAM or QSAM DCB but the DCBE macro does not have DSENCRYPT=OK.

3.12.2 IEC143I

IEC143I 213-rc,mod,jjj,sss, ddname[-#],dev,volser,dsname(member)

The following existing reason codes introduced with data set encryption for extended format data sets also apply to data set encryption for basic and large format data sets (descriptions may be slightly modified):

83 During open processing for an encrypted data set, the system detected that either the encryption cell in the catalog does not exist for this data set or the key label in the encryption cell does not exist.

85 During open processing for an encrypted data set, an unexpected error was received from the ICSF service CSNBKRR2 used to process the key label associated with the data set.

86 During open processing for an encrypted data set, on return from the ICSF service used to process the key label associated with the data set, the system detected that the encryption type of the data key associated with the key label was not of a supported encryption type. Only encryption keys of type AES256 are supported for extended format data sets.

87 During open processing for an encrypted data set, an unexpected error was received from the BCFXCRYP service used to prepare for encryption processing for this data set.

89 An attempt was made to open a new encrypted non-compressed extended format sequential data set, or basic or large format data set, for output with QSAM using a BLKSIZE less than 16 bytes. However, the minimum block size supported for encrypted non-compressed extended format sequential data sets, and encrypted basic or large format data sets, is 16 bytes.
This also applies for an attempt to open a new encrypted basic or large format data set for output, where LRECL is specified and LRECL is < 16 bytes with RECFM=F(B(S)) or LRECL is < 12 bytes with RECFM=V(B(S)). This avoids the potential for failures due to an attempt to write a block less than 16 bytes during writing or QSAM buffer flushing at CLOSE.

91 During open processing for an encrypted sequential data set, the system determined that the data key in the CKDS associated with the key label for the data set is not the same data key used to encrypt the data set.

The following new reason codes are added for encryption of basic and large format data sets:

93 During open processing, the access method encryption macro, IGGENC, detected that an input parameter to the service was incorrect. In this case, the message includes additional diagnostic information (return code and reason code) returned from the macro.

95 During open processing, the access method encryption macro, IGGENC, detected that a field in the encryption cell for the data set is not valid. In this case, the message includes additional diagnostic information (return code and reason code) returned from the macro.

96 During open processing, the system was unable to obtain storage required for encryption processing. In this case, the message includes additional diagnostic information (return code and reason code) returned from the service.

99 During open processing, the system was unable to open the data set. See additional diagnostic information at the end of the message (4-byte return code and 8-byte reason code) to identify the reason.

- DIAG=keylength, or RC=8, RSN=1. The key length is non-zero. DIAG has the keylength.
- DIAG=BDAM, RC=8, RSN=2. The program used BDAM to open an encrypted basic or large format data set.

- DIAG=IGGENC, RC=8, RSN= '0000000000001511'X
  An attempt was made to open an encrypted basic or large format data set with an EXCP DCB, however the settings for DCBEDSENCNP and the non-prefixed bit (VVREDSENCNP) in the encryption cell (ENCRYPTA) are not consistent. The DCBEDSENCNP bit means that the application is writing encrypted blocks with no prefix. The VVREDSENCNP bit means that the blocks in the data set do not have prefixes. For other than the first open for output, either both must be off or both must be on. When attempting to access an existing encrypted basic or large format data set which does not contain block prefixes, the non-prefixed blocks setting in the application and the data set encryption cell must both be on. In this case, DCBEDSENCNP bit is on but the non-prefixed bit in the ENCRYPTA is off.
- DIAG=IGGENC, RC=8, RSN='0000000000001521'X
  An attempt was made to open an encrypted basic or large format data set with an EXCP DCB, however the settings for DCBEDSENCNP and the non-prefixed bit (VVREDSENCNP) in the encryption cell (ENCRYPTA) are not consistent. The DCBEDSENCNP bit means that the application is writing encrypted blocks with no prefix. The VVREDSENCNP bit means that the blocks in the data set do not have prefixes. For other than the first open for output, either both must be off or both must be on. When attempting to access an existing encrypted basic or large format data set which does not contain block prefixes, the non-prefixed blocks setting in the application and the data set encryption cell must both be on. In this case, the non-prefixed bit in the ENCRYPTA is on but DCBEDSENCNP bit is off.

  DIAG=IGGENC, RC=X'00000008',RSN=X'00000000 00001531'
  An attempt was made to open an encrypted basic or large format data set with a BSAM or QSAM DCB and the DCBEDSENCNP bit was specified or the non-prefixed bit (VVREDSENCNP) was on in the encryption cell indicating the blocks in the data set do not have prefixes. The DCBEDSENCNP bit means that the application program is writing encrypted blocks without prefixes. BSAM and QSAM access is not supported with encrypted basic and large format data sets that do not have block prefixes.

9A An attempt was made to open an encrypted basic or large format data set with an EXCP DCB and DCBE DSENCRYPT=OK was not specified.

3.12.3 IEC150I

IEC150I 913-*rc*,*mod*,*jjj*,*sss*, *ddname*[-#],*dev*,*ser*,*dsname*(*member*)

Return code 84 has an additional meaning. It used to be only for an extended format data set. Now it also might apply to a basic or large format data set.

3.13 *z/OS MVS System Messages, Vol 8 (IEF-IGD) (SA38-0675)*

3.13.1 IGD17151I

Message IGD17151I will be updated as follows:

IGD17151I ALLOCATION FAILED FOR DATA SET *dsname* BECAUSE A KEY LABEL IS SPECIFIED FOR *rsntxt*.

Explanation: SMS VTOC data set services has detected that a data set key label is specified for an unsupported DASD data set type. For SMS-managed data set types library (PDSE), large format and basic format, in addition to message IGD17151I, message IGD17157I will be issued.

z/OS supports Data set key labels for the following data set types:

- Extended format (VSAM and sequential)
- Library (PDSE) if STGADMIN.SMS.ALLOW.PDSE.ENCRYPT is defined
- Large format if STGADMIN.SMS.ALLOW.DATASET.SEQ.ENCRYPT is defined

- Basic format if STGADMIN.SMS.ALLOW.DATASET.SEQ.ENCRYPT is defined. An exception is that temporary data sets cannot be encrypted. These are data sets with names that begin with an ampersand.

Data set key label can come from:

- DATAKEY(key_label) in DFP segment in RACF data set profile
- JCL and dynamic allocation keyword, DSKEYLBL=key_label
- Parameter, Key Label, in the Data Class

In the message text:
dsname The data set name
rsntxt Can be one of the following:

- AN UNSUPPORTED DATA SET TYPE
- A NON-SMS MANAGED DATA SET
- A DIRECT (BDAM) DATA SET
- A TEMPORARY DATA SET

3.13.2 IGD17157I

IGD17157I DSNTYPE *type* IS NOT A SUPPORTED DATA SET TYPE FOR ENCRYPTION BECAUSE *facility* IS NOT DEFINED

Explanation: An attempt was made to allocate an encrypted "type" data set by supplying a data set key label via JCL DSKEYLBL or SMS data class data set key label. However, SMS detected that the related RACF facility class resource "facility" is not defined to the system; and the user has at least READ access to the resource

STGADMIN.SMS.FAIL.INVALID.DSNTYPE.ENC.

In the message text:

type The data set name type. The possible values are:

- BASIC
- LARGE
- LIBRARY

facility The RACF facility class name. The possible values are:

- STGADMIN.SMS.ALLOW.PDSE.ENCRYPT
- STGADMIN.SMS.ALLOW.DATASET.SEQ.ENCRYPT

3.14 *z/OS DFSMSdfp Advanced Services (SC23-6861)*

IGGENC is a new macro that is part of the support for encryption of basic format and large format data sets. It is designed to be used with programs that use EXCP, EXCPVR or XDAP to read and write these encrypted data sets. Previous data set encryption functions were for extended format data sets and PDSEs. Those two types of data set did not require the IGGENC macro because user programs cannot use EXCP with them. IGGENC is not designed for use with an encrypted extended format data set or PDSE. Some advanced user programs and vendor programs use EXCP, EXCPVR or XDAP for basic and large format data sets. Those users will want to adapt their programs to be able to read and write encrypted data sets. Generally they will want the resulting data sets to be readable and writable with the regular BSAM and QSAM access methods in other programs. This includes backup and restore utility programs.

The IGGENC macro documentation is added to the *z/OS DFSMSdfp Advanced Services* publication in the EXCP chapter, “Executing Your Own Channel Programs”, instead of to a publication with regular access method documentation because it is something that the great majority of application programmers will not be interested in. It is expected that most interest in IGGENC will be for programs that use the EXCP, EXCPVR or XDAP macros.

3.14.1 Existing Organization of the EXCP Chapter (“Executing Your Own Channel Programs”) in *DFSMSdfp* Advanced *Services*

These are the existing level 2 headings in the EXCP chapter (“Executing Your Own Channel Programs”):

Comparing EXCP and EXCPVR
Using EXCP and EXCPVR
Allocating the Data Set or Device
Opening the Data Set
Creating the Channel Program
Creating the EXCP-Related Control Blocks
    Add the new DSENCRYPT option to the list of DCBE options supported by EXCP. Also add BYPASS_AUTH and CONCURRENTRW although they are not related to encryption. They were omitted in prior projects.
Executing the Channel Program
Processing the I/O Completion Status
Handling End of Volume and End-Of-Data-Set Conditions
Closing the Data Set
Control Block Fields
EXCP and EXCPVR Appendages
Converting a Relative Track Address to an Actual Track Address
Converting an Actual Track Address to a Relative Track Address
Using the IECTRKAD Callable Service or the TRKADDR Macro
Obtaining the Sector Number of a Block on an RPS Device

A new level 2 heading will be at the end of this chapter. The heading will be “Encrypting and Decrypting with the IGGENC Macro” as shown below.

3.14.2 New Section: Encrypting and Decrypting with the IGGENC Macro

When you bypass the access methods, you can use the IGGENC macro to encrypt and decrypt data such that the result is compatible with how the access methods encrypt records. This section describes how to use IGGENC with basic format and large format data sets that are SMS-managed. Data set encryption is described further in *z/OS DFSMS Using Data Sets.*

Encryption and decryption require the functions of ICSF, Integrated Cryptographic Service Facility. One of the ICSF functions is to manage access to your encryption key. You can use a cryptographic service such as ICSF to generate a random encryption key but you cannot see your encryption key. You provide a name for the key and you use that name. The name is called a “key label”. The key label is not a secret but its key is a secret even from the owner of the key.

Your program can test the DFASEQENCRYPT bit in the DFA to determine whether the IGGENC function is installed and functional. See the IHADFA mapping macro. If the basic crypto facility is not functional, IGGENC will give a reason code with the two low order bytes as X'0611' or X'081x'. See below.

These are the functions that you can perform with the IGGENC macro:

- Connect to the encryption function.
- Encrypt or decrypt one or more blocks of data. These require that your program either call the connect function first or open an encrypted basic or large format data set.

- Disconnect from the encryption function.

3.14.3 IGGENC Description

3.14.3.1 Programming Environment

The requirements for the caller are:

Environmental factor	Requirement
Authorization	Problem state or supervisor state and any PSW key. Calls after the connect call must be in the same protection key or be in key 0.
Dispatchable unit mode	The connect and disconnect functions require task mode. The encrypt and decrypt functions can be called in task or SRB mode.
Cross memory mode	PASN=HASN=SASN
AMODE	24-bit, 31-bit or 64-bit. If the invocation is in 64-bit, you must precede the invocation with an invocation of the SYSSTATE macro with AMODE64=YES.
ASC mode	Primary
Interrupt Status	Enabled for I/O and external interrupts
Locks	Holding the local lock of the home address space is optional. When the caller holds that lock during encrypt or decrypt calls, ICSF captures statistics but does not write them out until a call without the local lock or when the disconnect function is called. The system calls EXCP appendages while holding the local lock.

3.14.3.2 Programming Requirements

3.14.3.2.1 Encryption and RACF Security

In order to encrypt or decrypt with the key label, your program must have at least RACF read authority to the key label. There are two exceptions to this requirement for authorization to the key label:

- If the system is bypassing security for your job step because the JSCBPASS bit is on. It will be on if your job step program is in the system's program properties table (PPT) and the entry has the NOPASS attribute. The program properties table is defined by the SCHEDxx member of SYS1.PARMLIB.
- If your program has APF authorization, supervisor state or system key and either of these is true:
    - Your program passed a DCB to the encrypt or decrypt function and the DCB points to a DCBE with BYPASS_AUTH=YES coded or
    - Your program called the IGGENC macro for the connect function with an options block that has the IGGENCBypassAuth bit on.

If the ICSF installation options specifies CHECKAUTH(YES), the caller must also have authority to the CSFSERV class with profile CSFKRR2. Note: CHECKAUTH(NO) is the default.

3.14.3.2.2 Testing for Data Set Encryption

3.14.3.2.2.1 Testing Before or After Opening the Data Set

To test whether a data set is encrypted, your program can read the DSCB with the OBTAIN or CVAFDIR macro as described in this book. If the DS1ENCRP bit is on, the data set is encrypted.

- If the DS1PDSE bit also is on, then the data set is a PDSE and cannot be used with EXCP. (If the DS1PDSEX bit also is on, it is an HFS data set and the data set cannot be encrypted.)
- If the DS1STRP bit also is on, then the data set is extended format (possibly also compressed format) and cannot be used with EXCP.
- If neither of the above two bits is on and the DS1DSORG field has the DS1DSGPS bit on or the DCBDSORG field is empty, then it means that the data set is a basic format or large format data set. If the DS1LARGE bit is on, it is a large format data set.

3.14.3.2.2.2 Testing for Encryption

If your program provides a DCBE, your optional DCB open exit routine can test the DCBE to see whether the data set has the encryption attribute. You can do the same test after OPEN.

The bit to test is DCBE_DS_ENCRYPTION. It means that the data set is encrypted. The data set is a PDSE or is basic, large or extended format.

If your program is reading a sequential concatenation, then this bit is for the current data set.

The system sets this bit on or off as appropriate for each data set. If your program turns on the unlike attributes bit, DCBOFPPC, then the system will call your DCB open exit routine at the beginning of each data set. This allows your program to adjust to each data set. If you do not turn DCBOFPPC on, then the system calls your DCB open exit routine only for the first data set. In this case you might want to provide an EOV exit routine. The system will call it at the beginning of each volume, including the first volume of each data set after the first. This also allows your program to adjust to encryption differences.

Even if two data sets have the same key label, they will encrypt differently. If you pass the DCB to IGGENC for encryption and decryption, the system will take care of this difference in encryption. Instead if you call the IGGENC CONNECT function, you must disconnect it at the end of each data set and call IGGENC CONNECT for the next data set.

3.14.3.2.2.3 Testing After You Open the DCB

Learning the maximum block length. For maximum compatibility with the access methods and for maximum reliability in case of a program adding or removing blocks, tracks or volumes, your encrypted basic or large format data sets should have block prefixes. Prefixes are eight bytes long.

The DS1BLKL field in the DSCB contains the maximum block length for the data set. It does not include the length of the block prefix.

Other fields that contain the maximum block size for a data set and do not include the length of the prefix are:

- The JFCBLKSI field in the JFCB.
- The DCBBLKSI field in the DCB for BSAM and QSAM. It is two bytes. This field is not defined in an EXCP DCB.
- The DCBEBLKSI field in the DCBE for any access method. It is four bytes. Your EXCP program can use the field.
- The ARAXLBKS field as mapped by the IHAARA macro as returned by the RDJFCB macro. It is eight bytes.

If you issue the DEVTYPE macro with INFO=AMCAP, it returns the maximum block size for the device. It does not take encryption into account.

Learning whether the data set is encrypted and the block prefix length. In addition to testing the DSCB, there are two ways to learn whether the data set is encrypted:

- In your optional DCB open exit routine, test the DCBE_DS_ENCRYPTION bit in the DCBE. It will remain valid after the DCB is open. In a sequential concatenation, it will remain value (on or off) for each data set.
- Issue the ISITMGD macro after opening the DCB. To learn whether the data set is encrypted, test the ISMENCRP bit.

To learn the length of the block prefix after the DCB is open, code VERSION=2 on the ISITMGD macro and then test the ISMBPRFX byte. For further information about ISITMGD, see *z/OS DFSMS Macro Instructions for Data Sets.*

3.14.3.2.3 Opening an Encrypted Data Set

Opening an encrypted data set with a BSAM or QSAM DCB does not require any source code changes or reassembly. If your program issues the EXCP, EXCPVR or XDAP macro for an encrypted data set, the DCB must point to a DCBE that has the DSENCRYPT=OK option set. It also must meet either of these requirements:

- The DCB has a MACRF value as required for BSAM or QSAM. The system checks the DCBE during each invocation of the EXCP or XDAP macro. (You can use the DCB as a task library but in that case it cannot be encrypted.)
- The DCB has MACRF=E to identify it as a DCB that is only for EXCP, EXCPVR or XDAP. The system checks the DCBE during open.

3.14.3.2.4 Block Prefixes with Data Set Encryption

Statistical analysis to break encryption was discovered in World War 2. Block prefixes in data sets are an important part of preventing statistical analysis.

When the system creates the data set's entry in the catalog, it stores a large random number in the catalog entry. The system concatenates that large random number with the prefix for each disk block to create what is called the tweak value to encrypt that disk block. The content of each prefix should be different. This ensures that each block encrypts differently and each copy of a data set will encrypt differently (unless the copying program bypasses the access method and copies the tracks. DFSMSdss is one such program that copies the tracks.). If you write the same record multiple times, the encrypted version of each copy differs unpredictably.

With EXCP you can choose to write some blocks without encryption where the indicator is stored in the prefix. but those blocks still require prefixes.

To learn whether the data set has prefixes, your program can issue the ISITMGD macro after opening the DCB.

Do not encrypt or decrypt the block prefixes. You do encrypt and decrypt the BDWs, block descriptor words, in variable-length blocks.

You pass the block prefixes to the IGGENC macro with the BLKIDLIST parameter.

Contents of the block prefix for an encrypted data set as mapped by the IGGEBPFX macro

Offset	Type	Length	Name	Description
0	DSECT		EBP	Encryption block prefix in basic or large format sequential data set
	Bits	1	EBPFlag1	Flags.
	1...		EBPEncrypted	This block is encrypted
	.xxx xxxx			Reserved, should be zero
1	Binary	2		Reserved, should be zero
3	Integer	4	EBPTTTT	Relative track number across all volumes, first is zero
7	Integer	1	EBPR	Block number on track, first is 1

3.14.3.2.5 Encrypted Blocks Without Prefixes

Creating data sets without prefixes should be rare and controlled. You can create an encrypted data set without prefixes if all of these are true:

- Programs used to modify the data set can be controlled and are enhanced to understand how to read and write such a data set without block prefixes.
- All blocks in the data set are assumed to be encrypted.
- The relative location of every physical block within the data set can NEVER change.
    - Blocks can NEVER be added or deleted within a track, other than from the end.
    - Tracks can NEVER be added or deleted from the data set, other than from the end of the last volume that contains data.
- You understand that programs that use BSAM or QSAM will not be able to open this type of data set. An attempt to do so will result in an abend 213-99, with diagnostic information at the end of the message to identify the reason.
    - Not allowing access by a BSAM or QSAM application will prevent the potential scenario where standard copying programs copy the data set to an encrypted data set. In this case, the target of the copy would then contain block prefixes.

You can set a DCBE option to inform the system that the data set does not have block prefixes and your program can support it. The DCBE option is the DCBEDSENCNP bit. The DCBE macro does not have a keyword for this. This has an effect when the first OPEN macro option is other than INPUT (thus, includes OUTPUT, OUTIN, INOUT, and UPDAT) and the DCB is for EXCP. The system will save the indicator in the ENCRYPTA portion of the encryption cell in the catalog entry for the data set. This DCBE option is required for both reading and writing with EXCP. An attempt to subsequently open a data set of this type with EXCP for reading or writing without the DCBE DCBEDSENCNP option will result in an abend 213-99, with diagnostic information at the end of the message to identify the reason.

When your program calls the IGGENC macro to encrypt or decrypt, it must create and pass the appropriate block prefixes even though they do not exist on the device.

3.14.3.2.6 Reading and Writing Encrypted Data

When you call the IGGENC macro to encrypt or decrypt, you pass the prefix for each block so that IGGENC can encrypt or decrypt the blocks correctly.

You might choose for your EXCP program to read and write each block in virtual storage areas where the prefix and encrypted data are contiguous with each other or they might be non-contiguous. When you call the IGGENC macro, you still pass the prefix address list separately from the block address list.

If you choose to read and write the prefixes in areas that are not contiguous with the encrypted data, then your channel program must use one of these methods:

- Data chaining in a CCW channel program.
- MIDAWs in a CCW channel program.
- TIDAWs in a transport mode (zHPF) channel program.

3.14.3.3 Restrictions

If your program is running in 64-bit addressing mode, the parameter list and all parameters can reside above the 2 GB bar. If your program is running in 24-bit or 31-bit mode, addresses in the parameter list are treated as 31-bit and addresses in the lists that the parameter list points to are treated as 64-bit.

3.14.3.3.1 Keys

Encrypted data sets cannot have hardware keys. That means that KEYLEN in the DSCB, DCB, DD statement or dynamic allocation must be 0.

3.14.3.3.2 Minimum length of a block

The minimum length of an encrypted block is 16 bytes.

3.14.3.3.3 Reading Part of a Block

It is valid for your program to suppress data transfer with data chaining to skip over part of a block but that technique will not work with an encrypted data set because the encryption algorithm requires data from the beginning of the block.

If you read from the beginning of a block, you can read less than the whole block but there are special restrictions to decrypt it. These principles apply both when you intentionally read less than the whole block and when the device fails to return the whole block. In both cases you might want to pass the data length to IGGENC rounded up to a multiple of 16.

This is because the XTS encryption mode always encrypts groups of 16 bytes. You can encrypt any number of bytes that is at least 16. If you pass a data length that is not a multiple of 16, the algorithm encrypts the last 16 bytes overlapped with previous bytes. Let us say that the actual block length is 80 bytes, not counting the prefix. Let us say that your channel program read only the first 50 bytes. If you pass a data length of 50 to IGGENC, the last two bytes will be returned as garbage and this will not be detected unless you have some way to test validity. This is because the algorithm will incorrectly decrypt the last two bytes. If you want to decrypt those two bytes, you must read at least 64 bytes (the next multiple of 16) and pass those 64 bytes to IGGENC.

In this example if the number of bytes that your program read is 65 to 79, you cannot decrypt the bytes after the first 64 bytes. You can pass those bytes to IGGENC for decryption but those decrypted bytes will contain garbage. In this example you should read the whole 80 bytes.

3.14.3.4 Input Register Information

Before issuing the IGGENC macro, you do not have to place information into any register unless you use it in register notation for a particular parameter or use it as a base register.

You do not have to point register 13 to a register save area before issuing the IGGENC macro.

3.14.3.5 Output Register Information

Reg.	Register Contents
0	Reason code (eight bytes even if your program is not running in 64-bit). Also returned as the third parameter.
1	Used as a work register by the system
2-13	Unchanged
14	Used as a work register by the system
15	Return code (four bytes). Also returned as the second parameter.

3.14.3.6 Syntax of IGGENC Standard Invocation

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT. It can be in mixed case such as Connect.
BLKIDLIST=	*list addr*: A-type address, or register (2) - (12).
BUFCOUNT=	*count addr*: A-type address, or register (2) - (12).
BUFLENLIST=	*list addr*: A-type address, or register (2) - (12).
DCB=	*dcb addr*: A-type address, or register (2) - (12).

Syntax	Description
ENCRYPTA=	*encrypta addr*: A-type address, or register (2) - (12).
INBUFLIST=	*list addr*: A-type address, or register (2) - (12).
OPTIONS=	*area addr*: A-type address, or register (2) - (12).
OUTBUFLIST=	*list addr*: A-type address, or register (2) - (12).
RETCODE=	*return code addr*: A-type address, or register (2) - (12).
RSNCODE=	*reason code addr*: A-type address, or register (2) - (12).
TCBADDR=	*tcb addr*: A-type address, or register (2) - (12).
TOKEN=	*token area:* A-type address, or register (2) - (12).

3.14.3.7 Parameters for the Standard Form of the IGGENC Macro

Table 1 - Required and optional keywords for the IGGENC macro

Keyword	Function		
	CONNECT	**ENCRYPT/ DECRYPT**	**DISCONNECT**
BLKIDLIST=	–	Required	–
BUFCOUNT=	–	Required	–
BUFLENLIST=	–	Required	–
DCB=	–	Optional	–
ENCRYPTA=	Required	–	–
INBUFLIST=	–	Required	–
OPTIONS=	Required	Required	Required
OUTBUFLIST=	–	Optional	–
RETCODE=	Required	Required	Required
RSNCODE=	Required	Required	Required
TCBADDR=	Optional	–	–
TOKEN=	Required	Optional	Required

Function (positional parameter)

Required on standard form. Must be CONNECT, ENCRYPT, DECRYPT or DISCONNECT. Mixed case is allowed.

BLKIDLIST= A-type address, or register (2) - (12).

List of addresses of block prefixes. Each address and each block prefix is eight bytes even if the program is running in 24-bit or 31-bit. See BUFCOUNT.

BUFCOUNT= A-type address, or register (2) - (12).

A halfword containing the number of entries in the input buffer list (INBUFLIST), buffer length list (BUFLENLIST), output buffer list (OUTBUFLIST) and prefix list (BLKIDLIST). The value must be 1 or greater.

BUFLENLIST= A-type address, or register (2) - (12).

List of words that contain the length of each block. See BUFCOUNT.

DCB= A-type address, or register (2) - (12). DCB associated with the encrypted data set. Use DCB only with the ENCRYPT or DECRYPT function. The DCB or TOKEN keyword is required for the ENCRYPT and DECRYPT functions. This keyword is mutually exclusive with the TOKEN keyword.

While the DCB is open, you can pass the DCB to the ENCRYPT and DECRYPT functions under a task that is different from the task that opened the DCB.

ENCRYPTA= A-type address, or register (2) - (12).

96-byte ENCRYPTA cell from the catalog. This keyword is required for the CONNECT function. You can retrieve a copy of this area from the data set catalog by calling IGGCSI00, the catalog search interface. See *z/OS DFSMS Access Method Services Commands.* Also see *DFSMS Using Data Sets* for more detailed information.

INBUFLIST= A-type address, or register (2) - (12).

List of addresses of the input areas. Identifies areas containing data to encrypt or decrypt. Each must be eight bytes and can point above the bar even for a 24-bit or 31-bit caller. See BUFCOUNT.

OPTIONS= A-type address, or register (2) - (12).

Specifies an area containing the eight-byte options block as mapped by the IGGENCOPTS DSECT in the IGGENCPL macro.

Offset	Type	Len.	Name	Description
0	DSECT		IGGENCOpts	IGGENC options block
	Integer	1	IGGENCLength	Length of this block. Must be 8.
1	Integer	1	IGGENCFunction	Function code. Values for the function code:
			IGGENCConnect	1 – Connect
			IGGENCEncrypt	2 – Encrypt
			IGGENCDecrypt	3 – Decrypt
			IGGENCDisconnect	4 – Disconnect
2	Bits	1	IGGENCFlag1	Flags
	1...		IGGENCBypassAuth	Bypass security check. Valid only if connect and caller is APF auth, supervisor state or system key
	.1..		IGGENCENCRYPTA	The fifth parameter is ENCRYPTA and not an open DCB (CONNECT only). If you code the ENCRYPTA or DCB keyword on the execute form and also code the OPTIONS keyword, the macro sets or resets this bit.
	..xx xxxx			Reserved, should be zero
3		5		Reserved, should be zero
8	EQU		IGGENCOptsLen	Length of options block

OUTBUFLIST= A-type address, or register (2) - (12).

Optional list of output areas. Identifies areas to be used as the target of an encrypt or decrypt. The default is to use the input areas. Each must be eight bytes and can point above the bar even for a 24-bit or 31-bit caller. See BUFCOUNT.

RETCODE= A-type address, or register (2) - (12).

Four-byte area for the return code. The IGGENC macro also returns the return code in register 15.

RSNCODE= A-type address, or register (2) - (12).

Eight-byte area for the reason code. The IGGENC macro also returns the reason code in register 0. It is eight bytes even if your program is not running in 64-bit mode.

TCBADDR= A-type address, or register (2) - (12).

Word containing the address of a TCB, task control block. Code this only for the CONNECT function. If the word contents are non-zero, it must point to a TCB in the address space and the calling program must be in supervisor state or a system key or the job step must have APF authorization.

If the address in the parameter list is zero or the word that it points to contains zero, the connect will be for the current task. Therefore a way to denote the current task is to code TCBADDR=0 or to omit it.

TOKEN= A-type address, or register (2) - (12).

Eight-byte area for the token. This is required for the CONNECT and DISCONNECT functions. For CONNECT, the area must contain binary zeroes. It does not have to remain in the same location. You can make a copy to pass to other calls to IGGENC. The token content will remain valid until you call the DISCONNECT function or the task terminates, whichever is first. For DISCONNECT, the token must contain what the CONNECT function stored in it. The DISCONNECT function will clear the passed token to zeroes.

The ENCRYPT and DECRYPT functions require that you code either TOKEN or DCB. They are mutually exclusive. You can call the ENCRYPT and DECRYPT functions under a task that is different from the task that called the CONNECT function but the token becomes invalid when the connecting task does the disconnect or that task terminates.

3.14.3.8 IGGENC Parameter List

You can use the IGGENCPL macro to map the parameter list for the IGGENC macro. There is one form of parameter list for callers in 24-bit or 31-bit and a different form for 64-bit callers. If you wish the IGGENC macro (standard, list, execute or modify form) to expand the 64-bit version, you must precede the macro call by calling the SYSSTATE macro with AMODE64=YES. That macro affects IGGENC on a line-by-line basic without regard to branching.

Three of the parameters are lists of addresses of the input and output areas and the block prefixes. These addresses always are 64-bit (eight bytes each) even if the parameter list is the 31-bit form.

Function of Call	Number of addresses in parameter list
CONNECT	6
ENCRYPT	9
DECRYPT	9
DISCONNECT	4

The parameter list consists solely of a series of addresses. All of the addresses are either four bytes or eight bytes. The first address points to an options block. A byte in the options block specifies the purpose of the call.

The IGGENCPL macro supports these keywords:

DSECT={YES | NO}. Whether to generate a DSECT statement for each area. The default is YES.

PLIST={YES | NO}. Whether to map both forms of the parameter list. The default is YES.

OPTIONS={YES | NO}. Whether to map the options block. The default is YES.

CODES={YES | NO}. Whether to define the return and reason code constants. The default is YES.

The following is the format of the parameter list for 24-bit and 31-bit callers:

Offset	Type	Len	Name	Description
0	DSECT		IGGENCParmL	IGGENC parameter list
0	Address	4	DSEP1Opt	Address of options block
4	Address	4	DSEP2RC	Address of return code (four bytes)
8	Address	4	DSEP3ReaC	Address of reason code (eight bytes)
12	Address	4	DSEP4CT	Address of connect token (eight bytes).
16	Address	4	DSEP5Enc	CONNECT: Address of 96-byte ENCRYPTA area from the catalog entry
	Address	4	DSEP5Blk	ENCRYPT, DECRYPT: Address of list of 64-bit addresses of block prefixes. The number of addresses is pointed to by DSEP8Num.
20	Address	4	DSEP6TCBAddr	CONNECT: Address of a word containing the address of a TCB, task control block. If the word in the parameter list or the word that it points to contains zero, the token will be associated with the current task.
	Address	4	DSEP6InB	ENCRYPT, DECRYPT: Address of list of 64-bit addresses of data areas to encrypt or decrypt. The number of addresses is pointed to by DSEP8Num.
24	Address	4	DSEP7BL	ENCRYPT, DECRYPT: Address of list of data lengths. Each is four bytes. The number of addresses is pointed to by DSEP8Num.
28	Address	4	DSEP8Num	ENCRYPT, DECRYPT: Address of a halfword that contains the number of entries in the lists of input and output (if specified) addresses and the list of area lengths. This also denotes the number of block identifiers in DSEP5Blk
32	Address	4	DSEP9OutB	ENCRYPT, DECRYPT: Address of list of 64-bit addresses of data areas to contain the result of encryption or decryption. The number of addresses is pointed to by DSEP8Num. If this address is zero, the result of the encryption or decryption will be in the input areas.
36	EQU		DSELength	Maximum length of 31-bit parameter list

The following is the format of the parameter list for 64-bit callers:

Offset	Type	Len	Name	Description
0	DSECT		IGGENCParmL	IGGENC parameter list
0	Address	8	DSE64P1Opt	Address of options block
8	Address	8	DSE64P2RC	Address of return code (four bytes)
16	Address	8	DSE64P3ReaC	Address of reason code (eight bytes)
24	Address	8	DSE64P4CT	Address of connect token (eight bytes).
32	Address	8	DSE64P5Enc	CONNECT: Address of 96-byte ENCRYPTA area from the catalog entry
	Address	8	DSE64P5Blk	ENCRYPT, DECRYPT: Address of list of addresses of block prefixes. The number of addresses is pointed to by DSE64P8Num.

Offset	Type	Len	Name	Description
40	Address	8	DSE64P6TCBaddr	CONNECT: Address of a word containing the address of a TCB, task control block. If the doubleword in the parameter list or the word that it points to contains zero, the token will be associated with the current task.
	Address	8	DSE64P6InB	ENCRYPT, DECRYPT: Address of list of addresses of data areas to encrypt or decrypt. The number of addresses is pointed to by DSE64P8Num.
48	Address	8	DSE64P7BL	ENCRYPT, DECRYPT: Address of list of data lengths. Each is four bytes. The number of addresses is pointed to by DSE64P8Num.
56	Address	8	DSE64P8Num	ENCRYPT, DECRYPT: Address of a halfword that contains the number of entries in the lists of input and output (if specified) addresses and the list of area lengths. This also denotes the number of block identifiers in DSE64P5Blk.
64	Address	8	DSE64P9OutB	ENCRYPT, DECRYPT: Address of list of addresses of data areas to contain the result of encryption or decryption. The number of addresses is pointed to by DSE64P8Num. If this address is zero, the result of the encryption or decryption will be in the input areas.
72	EQU		DSELength64	Maximum length of 64-bit parameter list

3.14.3.9 Return and Reason Codes for IGGENC Macro

The return code is four bytes. It is returned in register 15 and in the area provided by the second parameter.

The reason code is eight bytes. It is returned in register 0 and in the area provided by the third parameter.

Table of Return and Reason Codes for the IGGENC macro.

Return Code	Reason Code (Hexadecimal)	Meaning
0	0	The call was successful.

Return Code	Reason Code (Hexadecimal)	Meaning
8		Return code IGGENCRC8. Bad parameters passed in parameter list. The *x* digit identifies the function, where 1: CONNECT, 2: ENCRYPT, 3: DECRYPT and 4: DISCONNECT.
	00000000 0000011*x*	IGGENCReaAddr. A required address is 0.
	00000000 0000012*x*	IGGENCReaFunc. Invalid function code in options. The invalid byte is ORed into the low order byte of the reason code so the high half might not be X'2'.
	00000000 0000013*x*	IGGENCReaLength. Length byte in options is less than eight. The function code is in the *x* digit.
	00000000 00*xx*014*x*	IGGENCReaToken. Bad token. Diagnostic codes for the *xx* byte: IGGENCReaTokenDiag32 X'20' Token not zero for connect IGGENCReaTokenDiag33 X'21' Token is 0 but function not connect
	00000000 00000151	IGGENCReaDEBErr. Invalid DEB.
	00000000 0000016*x*	IGGENCReaLockErr. Local lock not held. Internal system error.
		Bad fields passed in ENCRYPTA area
	00000000 *xx*000211	IGGENCReaType. Bad encryption type in ENCRYPTA. *xx* is the type code.
	00000000 *xx*000221	IGGENCReaKLen. Bad key length code in ENCRYPTA. *xx* is the incorrect length.
	*xxxxxxxx xx*000231	IGGENCReaKLab. Bad key label in ENCRYPTA. The first five bytes are the first five bytes of the incorrect key label
	00000000 *xx*000241	IGGENCReaMode. Bad encryption mode in ENCRYPTA.
		Error when getting or freeing virtual storage.
	00000000 *xx*000311	IGGENCReaGet31. Unable to get storage below the bar. The *xx* is the STORAGE macro return code.
	00000000 *xx*000334	IGGENCReaFree31. Unable to free storage below the bar. The *xx* is the STORAGE macro return code.
	00000000 00000411	IGGENCReaNoEncCell. Encryption cell does not exist.
	00000000 0000051*x*	IGGENCReaSetLock. Error obtaining local lock
	xxyyyyyy 00000611	IGGENCReaICSFErr. Error encountered during CSNEKRR2. The return code is in the *xx* byte and its reason code is in the *yyyyyy* bytes.
	00000000 00000711	IGGENCReaCrypKL. Incorrect crypto key length
	xxyyyyyy 0000081*x*	IGGENCReaBCFErr. Error encountered during BCFXCRYPT. The return code is in the *xx* byte and its reason code is in the *yyyyyy* bytes.
	00000000 00000911	IGGENCReaVerErr. Incorrect verification bytes
	00000000 00000A11	IGGENCReaCatErr. Error occurred during Catalog call
	xxyyyyyy 00000B11	IGGENCReaKeyAcc. Not authorized to read key label. The return code from RACROUTE REQUEST=FASTAUTH is in the *xx* byte and its reason code is in the *yyyyyy* bytes.
	00000000 00000C1*x*	IGGENCReaBuffCnt. Encryption Buffer Count < 1
	00000000 00*xx*0D1*x*	IGGENCReaBuffLen. Encryption Buffer Length < 16 bytes. The number of the buffer is in the *xx* byte.

Return Code	Reason Code (Hexadecimal)	Meaning
	00000000 00*xx*0E1*x*	IGGENCReaBuffAddr. Encryption Buffer Addr is zero. The number of the buffer is in the *xx* byte.
	00000000 00*xx*0F1*x*	IGGENCReaPfxAddr. Block ID Address is Zero. The number of the block is in the *xx* byte.
	00000000 00*xx*101*x*	IGGENCReaBuffInv. Encryption Buffer invalid. The number of the buffer is in the *xx* byte.
	00000000 0000111x	IGGENCReaNonEncDS. Not an encrypted data set.
	00000000 00001211	IGGENCReaEncNotOk. EXCP caller without a DCBE that has DSENCRYPT=OK.
	00000000 00001310	IGGENCReaEncSysErr. Encryption service error
	00000000 00001411	IGGENCReaTCBAddr. Caller unauthorized to specify TCB
	00000000 00001511	IGGENCReaNpDCBE. DCBEDSENCNP bit is on, but the non-prefixed bit in the ENCRYPTA is off
	00000000 00001521	IGGENCReaNpCtlg. Non-prefixed bit in the ENCRYPTA is on, but DCBEDSENCNP bit is off
	00000000 00001531	IGGENCReaNpInvDCB. OPEN issued for encrypted basic/large format data set that has non-prefixed blocks but the DCB is not EXCP.

3.14.3.10 Examples of IGGENC Macro

3.14.3.10.1 Example 1 – Standard forms

```
          ISITMGD DCB=SecretDCB,Version=2
          USING ISM,R1
          LTR   R15,R15             Branch if unable to determine
          JZ    EncFail               encryption status
          TM    ISMOFLG2,ISMENCRP   Branch if the data set is
          JZ    NotEncErr             not encrypted
          SR    R15,R15             Prepare for IC
          IC    R15,ISMBPRFX        Get the length of the
          STH   R15,PrefixLen          encryption prefix
          DROP  R1
          AH    R15,DCBBLKSI-IHADCB+SecretDCB  Add max block length
          GETMAIN R,LV=(R15)        Get prefix and buffer
          STG   R1,PrefixAd         Save prefix address
          AH    R1,PrefixLen        Point to place to read the block
          STG   R1,InBufList        One entry in buffer list
          LH    R15,DCBBLKSI-IHADCB+SecretDCB  Get user data length
          ST    R15,BufferLen       Set length to decrypt
    (Build channel program to read a block prefix and block.)
          EXCP  MyIOB               Read a block and its prefix
          WAIT  ECB=ECB
          IGGENC Decrypt,RetCode=MyRC,RsnCode=MyReason,              *
                BLKIDLIST=PrefixAd,INBUFLIST=InBufList,              *
                BUFLENLIST=BufferLen,BUFCOUNT=BufCount,              *
                DCB=SecretDCB
          LTR   R15,R15
          JNZ   DecryptErr
    (More code.)
R1        EQU   1           Register 1
R15       EQU   15          Register 15
SecretDCB DCB   DSORG=PS,MACRF=R,DDName=SECRET,EXLST=MyList
OptBlock  DC    0XL8
          DC    AL1(L'OptBlock)  Length of options block
          DC    XL7'0'      Rest of options block
MyRC      DC    F'0'        Return code from IGGENC
MyReason  DC    XL8'0'      Reason code from IGGENC (eight bytes)
PrefixAd  DC    AD(0)       List of prefix addresses
InBufList DC    AD(0)       List of block addresses
BufferLen DC    F'0'        Length of block to decrypt
PrefixLen DC    H'0'        Length of prefix
BufCount  DC    H'1'        Number of blocks to decrypt
    (The IOB, channel program and ECB are here.)
```

Before issuing the IGGENC macro to connect to encryption, this program opens the BSAM DCB. It is valid to issue EXCP with a BSAM DCB if the data set is a basic format or large format data set or is a PDS. If the data set is an encrypted basic format or large format data set, you can issue the IGGENC macro for data read or written with EXCP. The READ and WRITE macros take care of encryption and decryption. IGGENC is needed only for data read or written with EXCP, EXCPVR or XDAP.

The DD name identifies a DD statement or dynamically allocated data set that is basic format or large format and has the encrypted attribute. In this case the DD name is SECRET. The data

set name, the key label and the encryption key do not matter to this program. Data set allocation, the OPEN macro and the IGGENC macro take care of them.

Before this program calls the IGGENC macro, it has initialized the IOB and channel program.

3.14.3.10.2 Example 2 – A call from a High Level Language

This is an assembler program that can be called by a program written in any high level language that can define the option block and lists of addresses.

Assembler subroutine callable from a High Level Language

```
IGGENC   TITLE 'IGGENC - Subroutine callable from compiled language'
*   The calling program must build a parameter list consisting only
* of addresses and pass its address in register 1.  The IGGENC macro
* does not use a register save area so this module can use it.
         USING SaveR,R13    Addressability to caller's save area
         ST    R14,SAVGRS14 Save the address of the caller
         IGGENC MF=(E,(1)) Call CONNECT, ENCRYPT, DECRYPT, DISCONNECT
         L     R14,SAVGRS14 Restore address of the caller
         BR    R14          Return to caller with return & reason
R13      EQU   13           Register 13
R14      EQU   14           Register 14
         IGGSAVER ,         Standard register save area
         END
```

3.14.4 IGGENC Macro – List Form

3.14.4.1 Syntax of the List Form

Each parameter that is shown in "Table 1 - Required and optional keywords for the IGGENC macro" on page 24 as required must be coded on the list, modify or execute form. The execute form takes precedence over the modify form. The modify form takes precedence over the list form.

The list form of the IGGENC macro is written as follows:

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT
BLKIDLIST	*list addr*: A-type address
BUFFERCOUNT	*count addr*: A-type address
BUFLENLIST	*list addr*: A-type address
DCB	*dcb addr*: A-type address
ENCRYPTA	*encrypta addr*: A-type address
INBUFLIST	*list addr*: A-type address
MF	L
OPTIONS	*area addr*: A-type address
OUTBUFLIST	*list addr*: A-type address
RETCODE	*return code addr*: A-type address
RSNCODE	*reason code addr*: A-type address
TCBADDR=	*tcb addr*: A-type address
TOKEN=	*token area:* A-type address

3.14.4.2 Parameters for the List Form

The parameters are explained under the standard form of the IGGENC macro, with the following exceptions:

Function (positional parameter)

Required on list form only when no parameter for ENCRYPT or DECRYPT is coded. Must be CONNECT, ENCRYPT, DECRYPT or DISCONNECT. Mixed case is allowed.

MF= L. This identifies the invocation as being the list form of the macro. Normally you code a symbol at the beginning of the IGGENC macro

OPTIONS= A-type address

In a reentrant program the options block can be in read-only storage if you code the OPTIONS keyword on the list form before copying it and you do not code OPTIONS on the execute form. If you do that, the macro does not modify the options block but you risk the possibility that future functions will require the execute form to modify the options block. Therefore it is advisable for the options block to be in storage that the macro can modify. An exception is if you code ENCRYPTA or DCB on the CONNECT call and also code the OPTIONS parameter on the execute call.

3.14.5 IGGENC Macro – Execute Form

3.14.5.1 Syntax of the Execute Form

Each parameter that is shown in "Table 1 - Required and optional keywords for the IGGENC macro" on page 24 as required must be coded on the list, modify or execute form. The execute form takes precedence over the modify form. The modify form takes precedence over the list form.

The execute form of the IGGENC macro is written as follows:

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT
BLKIDLIST	*list addr*: RX-type address, or register (2) - (12).
BUFFERCOUNT	Halfword count of number of entries in the input buffer list (INBUFLIST), buffer length list (BUFLENLIST), output buffer list (OUTBUFLIST) and prefix list (BLKIDLIST).
BUFLENLIST	*list addr*: RX -type address, or register (2) - (12).
DCB	*dcb addr*: RX -type address, or register (2) - (12).
ENCRYPTA	*encrypta addr*: RX -type address, or register (2) - (12).
INBUFLIST	*list addr*: RX -type address, or register (2) - (12).
MF	(E,*xxxx*[,COMPLETE]). The *xxxx* can be RX-type or register (1) – (12).
OPTIONS	*area addr*: RX -type address, or register (2) - (12).
OUTBUFLIST	*list addr*: RX -type address, or register (2) - (12).
RETCODE	*return code addr*: RX -type address, or register (2) - (12).
RSNCODE	*reason code addr*: RX -type address, or register (2) - (12).
TCBADDR=	*tcb addr*: RX-type address, or register (2) - (12).
TOKEN=	*token area:* RX-type address, or register (2) - (12).

3.14.5.2 Parameters for the Execute Form

The parameters are explained under the standard form of the IGGENC macro, with the following exceptions:

Function (positional parameter)

Required on the execute form only when no parameter for ENCRYPT or DECRYPT is coded. Must be CONNECT, ENCRYPT, DECRYPT or DISCONNECT. Mixed case is allowed.

MF= E. This identifies the execute form of the macro.

COMPLETE specifies that the system is to check for required parameters and supply defaults for omitted optional parameters. This sets all fields in the parameter list. If you code COMPLETE, there is no need for your program to set any field in the parameter list before calling the execute form. The macro will initialize the parameter list before storing into it. You must ensure that the area is long enough

3.14.6 IGGENC Macro – Modify Form

Each parameter that is shown in "Table 1 - Required and optional keywords for the IGGENC macro" on page 24 as required must be coded on the list, modify or execute form. The execute form takes precedence over the modify form. The modify form takes precedence over the list form. The purpose of the modify form is to update a parameter list without calling the function. It updates the parameter list with minimal checking. Later your program will call the execute form with the consolidated parameter list

3.14.6.1 Syntax of the Modify Form

The modify form of the IGGENC macro is written as follows:

Syntax	Description
Function (positional parameter)	CONNECT, ENCRYPT, DECRYPT or DISCONNECT
BLKIDLIST	list addr: RX-type address, or register (2) - (12).
BUFFERCOUNT	Halfword count of number of entries in the input buffer list (INBUFLIST), buffer length list (BUFLENLIST), output buffer list (OUTBUFLIST) and prefix list (BLKIDLIST).
BUFLENLIST	list addr: RX -type address, or register (2) - (12).
DCB	dcb addr: RX -type address, or register (2) - (12).
ENCRYPTA	encrypta addr: RX -type address, or register (2) - (12).
INBUFLIST	list addr: RX -type address, or register (2) - (12).
MF	(M,xxxx,[COMPLETE]). The xxxx can be RX-type or register (1) – (12).
OPTIONS	area addr: RX -type address, or register (2) - (12).
OUTBUFLIST	list addr: RX -type address, or register (2) - (12).
RETCODE	return code addr: RX -type address, or register (2) - (12).
RSNCODE	reason code addr: RX -type address, or register (2) - (12).
TCBADDR=	tcb addr: RX-type address, or register (2) - (12).
TOKEN=	token area: RX-type address, or register (2) - (12).

3.14.6.2 Parameters for the Modify Form

The parameters are explained under the standard form of the IGGENC macro, with the following exceptions:

Function (positional parameter)

Not required on the modify form. Mixed case is allowed.

MF= (M,*xxxx*), or (M,*xxxx*,COMPLETE). This identifies the modify form of the macro. The *xxxx* is the name of the parameter list to modify without calling the function.

The *xxxx* is a symbol or an RX form of address.

If you code the ENCRYPTA or DCB keyword on the execute form and also code the OPTIONS keyword, the macro sets or resets the IGGENCENCRYPTA bit.

COMPLETE means that you are coding all required parameters for an invocation. The macro will initialize the parameter list before storing into it.

3.15 *z/OS Summary of Messages and Interface Changes* (SA23-2300)

The following identifies messages which are updated with this function.

z/OS MVS System Messages, Vol 1 (ABA-AOM) (SA38-0668)

- ADR374E. See 3.11.1 ADR374E on page 14.

z/OS MVS System Messages, Vol 7 (IEB-IEE)

The following messages are updated:

- IEC036I (new reason codes added). See 3.12.1 IEC036I on page 14.
- IEC143I (new reason codes added). See 3.12.2 IEC143I on page 15.
- IEC150I (new reason codes added). See 3.12.3 IEC150I on page 16.

z/OS MVS System Messages, Vol 8 (IEF-IGD) (SA38-0675)

IGD17151I. See 3.13.1 IGD17151I on page 16.

IGD17157I. See 3.13.2 IGD17157I on page 17.

End of Document