z/OS

Object Access Method Application Programmer's Reference

SC35-0425-01

z/OS

Object Access Method Application Programmer's Reference

SC35-0425-01

Note
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 71.

Second Edition, March 2002

This edition applies to Version 1 Release 3 of z/OS™ (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC35-0425-00.

Order publications through your IBM® representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
RCF Processing, Department M86/050
5600 Cottle Road
San Jose, CA 95193-0001
United States of America

IBMLINK from US: STARPUBS at SJEVM5
IBMLINK from Canada: STARPUBS at TORIBM
IBM Mail Exchange: USIB3VVD at IBMMAIL
Internet: starpubs@us.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

About This Book

This book describes the programming interface provided by OAM. It is intended to show application programmers how to use the application programming interface to manipulate a special class of data called objects within the OAM system. Using this interface, programmers can store and retrieve specific objects. They can also request information concerning specific objects, change their attributes, and delete them from storage.

Application programmers may also use the information in this book to write custom interfaces that allow their installation's programs to work effectively with OAM.

Major Divisions of this Book

This book contains the following major divisions:

- "Chapter 1. Understanding the Object Access Method" on page 1 provides an overview of concepts relating to objects and the Object Access Method.
- "Chapter 2. Application Program Interface for OAM" on page 7 contains detailed information about the OSREQ macro and how it is used by application programs.
- "Appendix A. Sample Program for Object Storage" on page 41 provides assembler source code for a sample object storage request interface.
- "Appendix B. Reason Codes" on page 53 provides error descriptions and recommended responses for OAM return codes and reason codes.
- "Appendix C. Performance Considerations and Object Data Reblocking" on page 61 presents information about the effect of storage requirements, buffering, and other factors on application performance. This information is provided to help you with tuning. Tuning information should not be used as a programming interface.
- "Appendix D. Using the CBRUXSAE Installation Exit" on page 63 details how this exit is used to provide security checking for the OSREQ macro.
- "Glossary" on page 73 defines acronyms, abbreviations, and terms used in this document.
- "Index" on page 77 provides the page references for information concerning specific terms and concepts discussed in this book.

Required Product Knowledge

To use this book effectively, you should be familiar with:

- DATABASE 2™ (DB2)
- z/OS
- Customer Information Control System (CICS)—optional, depending on your installation
- Information Management System (IMS)—optional, depending on your installation
- Syntax diagrams

Referenced Publications

The following publications are referenced in this book, or are useful in understanding and applying the presented material:

Publication Title	Order Number
CICS Transaction Server for OS/390 Installation Guide	GC34-5697
CICS Messages and Codes	GC33-5716
CICS DB2 Guide	SC34-5707
DB2 Administration Guide	SC26-8957
DB2 Application Programming and SQL Guide	SC26-8958
DB2 Command Reference	SC26-8960
DB2 SQL Reference	SC26-8966
DB2 Utility Guide and Reference	SC26-8967
DB2 What's New	GC26-8971
DB2 Messages and Codes	GC26-8979
DB2 Diagnosis Guide and Reference	LY27-9659
HLASM General Information	GC26-4943
HLASM Language Reference	SC26-4940
HLASM Programmer's Guide	SC26-4941
IMS/ESA Application Programming: Database Manager	SC26-8727
IMS/ESA Application Programming: Design Guide	SC26-8728
IMS/ESA Application Programming: EXEC DLI Commands for CICS and IMS	SC26-8726
z/OS DFSMSdfp Diagnosis Guide	GY27-7617
z/OS DFSMSdfp Diagnosis Reference	GY27-7618
z/OS DFSMSdfp Storage Administration Reference	SC26-7402
z/OS DFSMS: Using the Interactive Storage Management Facility	SC26-7411
z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support	SC35-0426
z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Tape Libraries	SC35-0427
z/OS MVS Initialization and Tuning Guide	SA22-7591

Accessing z/OS DFSMS Books on the Internet

In addition to making softcopy books available on CD-ROM, IBM provides access to unlicensed z/OS softcopy books on the Internet. To find z/OS books on the Internet, first go to the z/OS home page: http://www.ibm.com/servers/eserver/zseries/zos

From this Web site, you can link directly to the z/OS softcopy books by selecting the Library icon. You can also link to IBM Direct to order hardcopy books.

Using LookAt to look up message explanations

LookAt is an online facility that allows you to look up explanations for z/OS messages, system abends, and some codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can access LookAt from the Internet at:

`http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html`

or from anywhere in z/OS where you can access a TSO command line (for example, TSO prompt, ISPF, z/OS UNIX System Services running OMVS).

To find a message explanation on the Internet, go to the LookAt Web site and simply enter the message identifier (for example, `IAT1836` or `IAT*`). You can select a specific release to narrow your search. You can also download code from the *z/OS Collection*, SK3T-4269 and the LookAt Web site so you can access LookAt from a PalmPilot (Palm VIIx suggested).

To use LookAt as a TSO command, you must have LookAt installed on your host system. You can obtain the LookAt code for TSO from a disk on your *z/OS Collection*, SK3T-4269 or from the LookAt Web site. To obtain the code from the LookAt Web site, do the following:

1. Go to http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookat.html.
2. Click the **News** button.
3. Scroll to **Download LookAt Code for TSO and VM.**
4. Click the ftp link, which will take you to a list of operating systems. Select the appropriate operating system. Then select the appropriate release.
5. Find the **lookat.me** file and follow its detailed instructions.

To find a message explanation from a TSO command line, simply enter: **lookat** *message-id*. LookAt will display the message explanation for the message requested.

Note: Some messages have information in more than one book. For example, IEC192I has routing and descriptor codes listed in *z/OS MVS Routing and Descriptor Codes*. For such messages, LookAt prompts you to choose which book to open.

Accessing Licensed Books on the Web

z/OS licensed documentation in PDF format is available on the Internet at the IBM Resource Link Web site at:

`http://www.ibm.com/servers/resourcelink`

Licensed books are available only to customers with a z/OS license. Access to these books requires an IBM Resource Link Web userid and password, and a key code. With your z/OS order you received a memo that includes this key code.

To obtain your IBM Resource Link Web userid and password log on to:

`http://www.ibm.com/servers/resourcelink`

To register for access to the z/OS licensed books:

1. Log on to Resource Link using your Resource Link user ID and password.

2. Click on **User Profiles** located on the left-hand navigation bar.
3. Click on **Access Profile.**
4. Click on **Request Access to Licensed books.**
5. Supply your key code where requested and click on the **Submit** button.

If you supplied the correct key code you will receive confirmation that your request is being processed. After your request is processed, you will receive an e-mail confirmation.

Note: You cannot access the z/OS licensed books unless you have registered for access to them and received an e-mail confirmation informing you that your request has been processed.

To access the licensed books:

1. Log on to Resource Link using your Resource Link userid and password.
2. Click on **Library**.
3. Click on **zSeries**.
4. Click on **Software**.
5. Click on **z/OS**.
6. Access the licensed book by selecting the appropriate element.

How to Send Your Comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other DFSMS documentation, please do one of the following actions:

- Send your comments by e-mail to:
  - IBMLink from US: starpubs@us.ibm.com
  - IBMLink from Canada: STARPUBS at TORIBM
  - IBM Mail Exchange: USIB3VVD at IBMMAIL
  - Internet: starpubs@us.ibm.com

  Be sure to include the name of the book, the part number of the book, version and product name, and if applicable, the specific location of the text you are commenting on (for example, a page number or a table number).
- Fill out one of the forms at the back of this book and return it by mail or by giving it to an IBM representative. If the form has been removed, address your comments to International Business Machines Corporation, Department 61C, 9000 South Rita Road, Tucson, AZ 85744-0002, U.S.A.

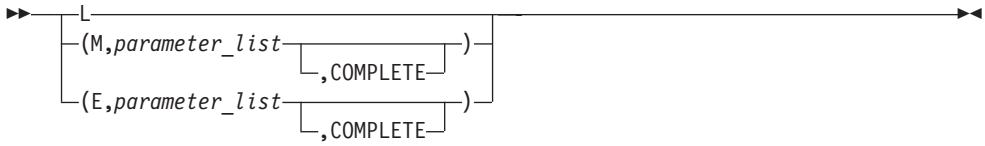
How to Read Syntax Diagrams

There is one basic rule for reading the syntax diagrams: Follow only one line at a time from the beginning to the end and code everything you encounter on that line.

The following rules apply to the conventions used in the syntax diagrams for all the OAM commands:

- Read the syntax diagrams from left to right and from top to bottom.
- Each syntax diagram begins with a double arrowhead (►►) and ends with opposing arrows (►◄).

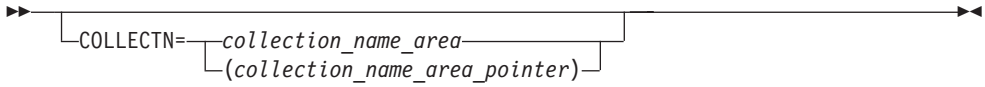
- An arrow (→) at the end of a line indicates that the syntax continues on the next line. A continuation line begins with an arrow (►–).
- Commands, keywords, and macro invocations are shown in uppercase letters.
- Where you can choose from two or more keywords, the choices are stacked one above the other. If one choice within the stack lies on the main path, a keyword is required, and you must choose one. In the following example you must choose either **L**, **M**, or **E**.



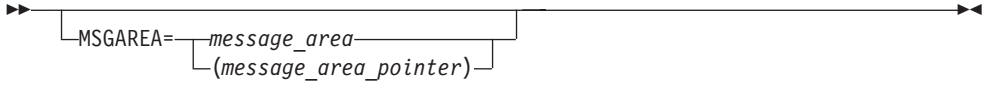
- If a stack is placed below the main path, a keyword is optional, and you can choose one or none. In the following example, **TOKEN**, **COLLECTN**, and **NAME** are optional keywords. You can choose any one of the three.

- Where you can choose from two or more keywords and one of the keywords appears above the main path, that keyword is the default. You may choose one or the other of the keywords, but if none is entered, the default keyword is automatically selected. In the following example you may choose either **PRIMARY**, **BACKUP**, or **BACKUP2**. If none is chosen, **PRIMARY** is automatically selected.

- Words or names in italicized, lowercase letters represent information you supply. The values of these variables may change depending on the items to which they refer. For example, in the syntax diagram below, *collection_name_area* refers to the name of a collection, while *collection_name_area_pointer* refers to the pointer for the collection name.

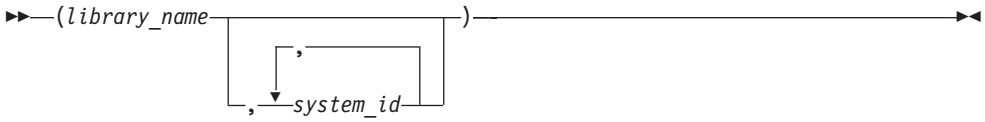


- You must provide all items enclosed in parentheses ( ). You must include the parentheses. In the following example, you must supply the volume serial number (*message_area_pointer*) and it must be enclosed in parentheses.



- The repeat symbol shown below indicates that you can specify keywords and variables more than once. The repeat symbol appears above the keywords and variables that can be repeated. For example, when a comma appears in the repeat symbol, you must separate repeated keywords or variables with a comma.

  In the following example, you may specify the *library_name* and one or more system identification numbers (*system_id*) that are separated by commas. You must enclose the name of the library and all of the system IDs in parentheses.



  You would code this as follows:

  (*library_name*, *system_id*, *system_id*, *system_id*)

  The variable *library_name* is the name of the library you are working with, and *system_id* names three different instances of system identification numbers.

Summary of Changes

This book contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by vertical lines to the left of the changes.

You may notice changes in the style and structure of some content in this book—for example, headings that use uppercase for the first letter of initial words only or procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in the z/OS DFSMS books.

Summary of Changes for SC35-0425-01 z/OS Version 1 Release 3

This book contains information previously presented in *z/OS Version 1 Release 1 Object Access Method Application Programmer's Reference* (SC35-0425-00).

The following sections summarize the changes to that information.

New Information

This edition includes the following new information:

- The BACKUP2 optional keyword for the VIEW parameter of the OSREQ RETRIEVE function will be modified for this new support. PRIMARY | BACKUP | BACKUP2 can be used with the VIEW parameter to indicate whether the primary copy of the object, the first backup copy of the object, or the second backup copy of the object should be retrieved.
- A new retrieval order key (QELQB2OK) on the CBRIQEL macro allows the RETRIEVE and QUERY functions to include information on a secondary backup of an object, if it exists.
- Two new return codes have been added to support the second backup copy of an object function.

Changed Information

The following information was changed in this edition:

- The section "Establishing a Storage Management Policy" on page 2 has been rewritten for greater clarity and to provide more specific information on setting up a storage management policy.
- The descriptions in "OSREQ Keyword Parameter Descriptions" on page 23 have been broken into two lines each to make them easier to read.
- The section "How to Read Syntax Diagrams" on page x has been enhanced.
- "Appendix C. Performance Considerations and Object Data Reblocking" on page 61 has been rewritten for greater clarity.

Moved Information

The following information was moved within this edition:

- The information on management policies in the section "Coordinating DB2, OAM, and Your Application" on page 5 was moved to the section "Establishing a Storage Management Policy" on page 2.
- The information on the data class ACS routine in the section "Coordinating DB2, OAM, and Your Application" on page 5 was moved to the section "Establishing a Storage Management Policy" on page 2.

- The information on how to read syntax diagrams in “Chapter 2. Application Program Interface for OAM” on page 7 was moved to the section “How to Read Syntax Diagrams” on page x.

Chapter 1. Understanding the Object Access Method

The Object Access Method (OAM) is a component of DFSMSdfp, the base for the z/OS product. OAM uses the concepts of system-managed storage, introduced by z/OS, which provide functions for data and space management. z/OS offers the following advantages to its users:

- Facilitates the management of storage growth
- Improves the use of storage space
- Reduces the effort of device conversion and coexistence
- Provides centralized control of external storage
- Exploits the capabilities of available hardware

OAM supports a class of data referred to as objects. An *object* is a named stream of bytes. The content, format, and structure of that byte stream are unknown to OAM. For example, an object can be a compressed scanned image or coded data. Objects are different from data sets handled by existing access methods. The characteristics that distinguish them from traditional data sets include:

Lack of record orientation
: There is no concept of individual records within an object.

Broad range of size
: An object may contain less than one kilobyte or many megabytes of data.

Volume
: Objects are usually much smaller than data sets; however, they are more numerous and consume vast amounts of external storage.

Varying access-time requirements
: Reference patterns for objects change over time or cyclically, allowing less critical objects to be placed on lower-cost, slower devices or media.

z/OS includes the definition of a storage hierarchy for objects and the parameters for managing those objects. OAM uses the z/OS-supplied hierarchy definition and management parameters to place user-accessible objects anywhere in the storage hierarchy.

The location of an object in the hierarchy is unknown to the user. Device-dependent information is not required of the user; for example, there are no JCL DD statements and no considerations for device geometry, such as track size.

OAM provides an application programming interface known as the object storage request (OSREQ) macro to store, retrieve, delete, query, and change information about an object. OAM includes the functions necessary to manage the objects after storing them.

OAM stores objects in collections. A *collection* is a group of objects that typically have similar performance characteristics:

CHARACTERISTIC	DESCRIPTION
Availability	The degree to which a resource is ready when needed.
Backup	A copy of the information that is kept in case the original is changed, lost or destroyed.
Retention	The default lifetime of an object.

Class transition An event that can cause the assignment of a new management class, storage class, or both.

A collection is used to catalog a large number of objects, which, if cataloged separately, require an extremely large catalog. Every object must be assigned to a collection. Object names within a collection must be unique; however, the same object name can be used in multiple collections. A collection can belong to only one storage group; however, a storage group can have many collections associated with it.

Understanding OAM Components

The functions of OAM are carried out by its three components:

- The **Object Storage and Retrieval Function (OSR)** stores, retrieves, and deletes objects. Applications operating in the CICS®, IMS™, TSO, and MVS/ESA™ environments use this application programming interface to store, retrieve, and delete objects, and to modify information about objects. Object Storage and Retrieval stores the objects in the storage hierarchy and maintains the information about these objects in DB2® databases.
- The **Library Control System (LCS)** writes and reads objects on tape volumes or optical disk storage, and it manipulates the volumes on which the objects reside. The LCS controls the hardware resources attached to the system.
- The **OAM Storage Management Component (OSMC)** determines where the objects should be stored, manages object movement within the object storage hierarchy, and manages expiration attributes based on the installation storage management policy defined through z/OS.

Establishing a Storage Management Policy

Each installation defines a storage management policy that allows effective object storage management without requiring user intervention. Through the use of Interactive Storage Management Facility (ISMF), the storage administrator and system programmer define an installation storage management policy in an Storage Management Subsystem (SMS) configuration. OAM then manages object storage according to the currently active policy.

OAM defines the management policy parameters in the SMS constructs of management class, storage class, storage group, and data class. The constructs include the following specifications:

- Object retention rates
- Media on which OAM stores object collections
- Legal requirements for object retention
- Retrieval response time
- Location of object collections in the storage hierarchy
- How long OAM should hold the object collection at that level in the hierarchy
- Whether you need one or two backup copies of an object
- Media type to which OAM should direct backup copies of objects
- Affiliation of libraries with relevant storage groups

Refer to *z/OS DFSMS: Using the Interactive Storage Management Facility* for general information on using ISMF. Refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support* and *z/OS DFSMS*

OAM Planning, Installation, and Storage Administration Guide for Tape Libraries for specifics of using ISMF within tape and optical storage environments to set up the management policy parameters.

Objects in OAM reside in a storage hierarchy that can include direct access storage devices (DASD), optical volumes, and tape volumes. Optical and tape volumes can be library-resident or shelf-resident. The primary copies of objects can be stored to DASD, optical volumes, or tape volumes; while backup copies of objects can only be stored to optical or tape volumes. OAM manages the storage hierarchy at the system level by using SMS management class, storage class, storage group, and data class constructs. The constructs specify the management policy parameters that define the performance, retention, and backup requirements. OAM associates these parameters with every object that it stores. The storage administrator defines the associations through automatic class selection (ACS) routines. The constructs are as follows:

Management Class
: Defines backup, retention, and class transition characteristics for objects. A management class contains parameters that define the need for making one or two backup copies of the object. They also determine the default lifetime of an object, and an event that can cause the assignment of a new management class, storage class, or both. OAM uses these parameters to create one or two backup copies of an object, to delete an object automatically, and to invoke an automatic class selection (ACS) routine when the specified transition event occurs. An ACS routine defines the management policy for a collection based on a combination of these constructs.

Storage Class
: Defines the level of service for an object, which is independent of the physical device or medium that contains the object. A storage class contains parameters that define performance characteristics and availability requirements for an object. OAM uses these parameters to determine where to place objects in the storage hierarchy (DASD, optical, or tape).

Storage Group
: Allows the user to define a storage hierarchy and to manage that hierarchy as if it were one large storage area. You may assign a first and a second Object Backup storage group to a specific Object storage group, or to all Object storage groups, by including SETOSMC statements in the CBROAM*xx* member of PARMLIB. For more information on multiple object backup specification and the SETOSMC command, refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*.

Data Class
: Defines tape-related information for scratch tape volumes that are allocated for OAM objects. The information defined by the data class includes the retention period, tape expiration date, tape compaction, recording technology, and media type.

  Note: You must update the data class's ACS routine to ensure that OAM does not assign a DATACLASS parameter to the OAM object-to-tape data sets. These data sets are named OAM.PRIMARY.DATA, OAM.BACKUP.DATA, or OAM.BACKUP2.DATA. You may associate a DATACLASS with a scratch tape volume through the SETOAM command of the CBROAM*xx* PARMLIB member when the scratch tape volume is allocated. Allowing the data class's ACS routine to

override or change the DATACLASS value provided by the SETOAM command can cause unexpected results. This may interfere with the storage management expectations for the installation. For more information on object-to-tape support and the SETOAM command, refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*. You should consider how your application affects the administration of the objects it stores.

To control the management of an object, assign it to a collection whose management policy is the same as that required by the new object. There is no explicit way to tell OAM where to store a particular object.

For more information on z/OS constructs, refer to the *z/OS DFSMSdfp Storage Administration Reference* manual.

Understanding the OAM Application Programming Interface

Typically, you want to do more with your files than store, retrieve, and delete them. You might write application programs to do things like update databases, pass data between workstations, communicate with peripheral devices, and other similar functions. See Figure 1 on page 5 for an example of the devices that may be used. OAM is designed to work with your application programs in the following environments:

- CICS
- IMS
- TSO
- MVS™ batch

For your applications to work well with OAM, you must consider OAM data types, partial object retrieval, DB2, OAM's object identification, management policy defaults, separating objects, and deletion of objects.

"Appendix A. Sample Program for Object Storage" on page 41 contains a sample program that uses the OSREQ macro for object storage and manipulation.

Choosing Data Types That Work Well with OAM

OAM is designed to work primarily with object data, although it is not restricted to that type of data. If your data is of the nontraditional type, is composed of many dissimilar records, is subject to infrequent updates, and is expected to be stored for long periods of time, then OAM is a good choice. On the other hand, if your data is of the traditional data set type, is composed of many similar records, and is subject to frequent updates, perhaps a different access method such as the ICF catalog or another currently supported access method is a better choice.

Retrieving a Partial Object

Although OAM does not support a record interface, if you need to store an object as a single entity and that object contains more than one logical entity, use the OAM partial object retrieve function to obtain those logical entities. For example, a drawing is composed of many subassemblies. Storing the subassemblies separately would take too much DASD space for OAM directory information, so they are stored as one object. The object is stored with control information (including subassembly identifiers, byte offsets, and lengths) that indicates where a subassembly is located within the object. Partial object retrieval allows you to read that control information and to use it to formulate an OAM request to retrieve a specific subassembly from within the object.

Coordinating DB2, OAM, and Your Application

OAM uses DB2 databases to contain descriptive information about every object that is stored. OAM does not commit the descriptive information written to that DB2 database; the application using OAM must perform that function. This allows the transaction to correlate and synchronize OAM's activity with other activity in the application (for example, synchronization of an application's and OAM's permanent database changes, or alternatively, synchronization of backing out of those changes).

Another example is an application transaction to perform an object update, something OAM does not support. That is, an object can be retrieved using OAM, updated by the application, original version deleted by OAM, new version stored by OAM with the original name, then committed as a permanent change by the application when it is satisfied with the results. If the application is not satisfied with the results, it has the option of preserving the original object by backing out all of the changes made by OAM up to that point.

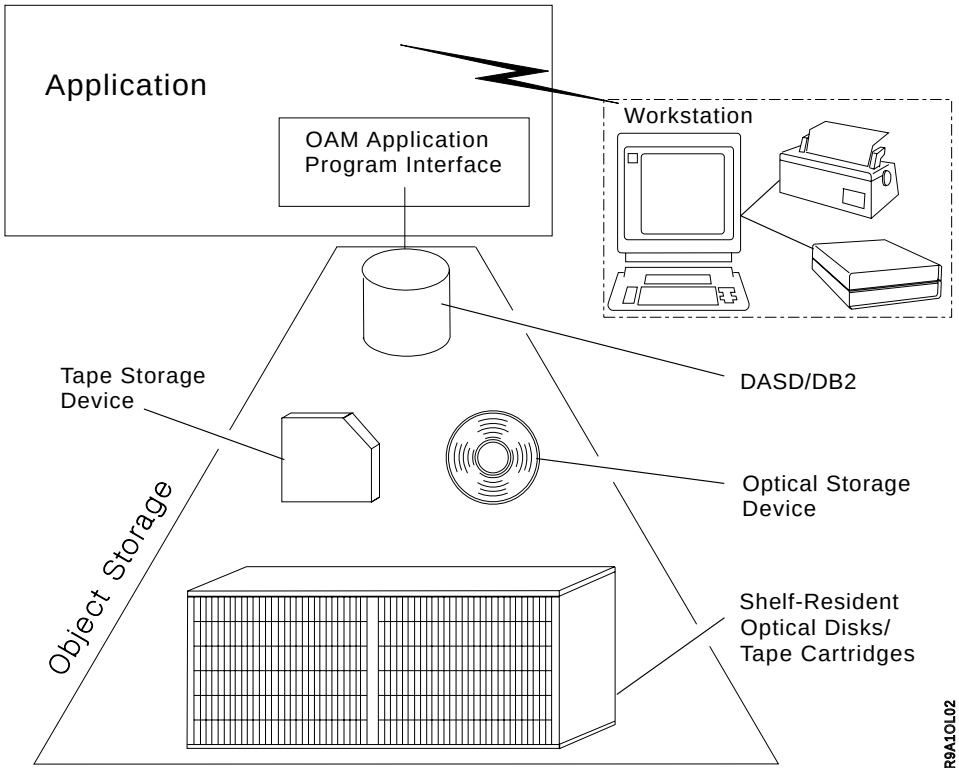


Figure 1. Application Illustration

Coordinating Your Application with OAM's Object Identification

OAM uses two-level naming: an object name and a collection name. Once you define a collection, give it a name, and establish its management policy, you can add objects to the collection by using the collection name as part of the object name, thus assigning the management policy to the new object.

The names you choose for collections and objects are important because normally objects associated with a particular collection are managed by the management policies for that collection. If you choose to store an object into a collection that has been previously established, the object will be managed according to the collection's management policies unless you specifically override those policies for

the object. Likewise, if you choose an object name that assigns the new object to a previously defined collection, the new object is managed according to the previously defined collection's management policy. Before coding an application, you should consult your installation's storage administrator for a naming convention for your application.

Overriding Management Policy Defaults

You will probably be storing several types of data that have different performance objectives and different management criteria. Some of your stored objects may need faster access time than others, and some may need backup copies, but others may not. Place objects that have differing characteristics in different collections. If the number of objects that differ is small, instead of creating a new collection, consider overriding the defaults by using explicit class names on the interface to OAM. Refer to "Processing a STORE to an Existing Collection" on page 22.

Separating Objects

OAM records descriptive information about each object that is stored. If your application stores a large number of objects, the amount of descriptive information can become excessive, causing performance degradation. OAM does not separate any descriptive information for objects in the same collection. It may separate descriptive information for objects in different collections, making it possible to improve performance by reducing the size of the accumulated descriptive information.

If you decide to separate one set of objects from another set, place them in different collections within the storage group. To ensure that collections remain separate, assign them to separate storage groups. System variables, including ACS routines, determine physical separation of objects. The number of objects your application stores may lead to your decision to separate objects by collections.

Deleting Objects

Your application design need not include explicit deletion of objects. The management class associated with an object can specify that the object is to be deleted after some time has elapsed. If your application keeps information about objects (for example, their names) in a repository, you should consider synchronizing the maintenance of that information with the automatic deletion of objects. For more information on the Auto Delete installation exit for deleting objects, refer to the *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*.

Chapter 2. Application Program Interface for OAM

The Object Access Method provides the object storage request macro (OSREQ) as an application program interface for storing and retrieving objects. Object storage requests can also return information (attributes) about specific objects, change attributes of specific objects, and delete objects from storage.

Using the OSREQ Macro

The OSREQ macro is the application program interface to OAM and is located in the SYS1.MACLIB macro library. Assembler H Version 2 is required to assemble this macro. For a list of books that contain more information about Assembler H, see “About This Book” on page vii.

See “Appendix C. Performance Considerations and Object Data Reblocking” on page 61 for performance considerations to take into account when writing your application program that interfaces with the OSREQ macro.

See “Appendix D. Using the CBRUXSAE Installation Exit” on page 63 for information on and a sample of the CBRUXSAE security authorization installation exit that is used at the OSREQ macro level.

Here is What You Can Do with OSREQ

The OSREQ macro permits the caller to request the following OAM functions:

FUNCTION	DESCRIPTION
Access	Establishes resources common to a set of OAM requests. Returns a token that must be specified with all other requests associated with this set.
Change	Changes an object’s directory entry reference to management class, storage class, and/or the expiration date, subject to the approval of the ACS routines.
Delete	Removes an object’s directory information and frees all reusable resources allocated to the object.
Query	Interrogates the object directory and returns information describing objects within the storage system. Specific and generic (wild card) queries are permitted.
Retrieve	Locates the requested object and returns the entire object or the specified portion of it in the virtual storage buffer provided by the caller.
Store	Records an object’s management criteria, object storage location, and other information in an object directory. Places the new object into the object storage hierarchy at a specific hierarchy level based on the storage class.
Unaccess	Frees the resources obtained with an OSREQ ACCESS request. The token cannot be used after the UNACCESS invocation.

“Implementing the Functions” on page 9 contains detailed descriptions of the functions and their corresponding syntax diagrams.

Choosing the Form

OSREQ is available in three forms, summarized in the following list:

MACRO FORM	DESCRIPTION
List (MF=L)	Generates a parameter list that can be used with the other forms of the macro.
Modify (MF=M)	Updates the parameter list with new parameters (specified when the modify form is invoked).
Execute (MF=E)	Initiates execution of the actual object request; also updates the parameter list if new parameters are specified when the execute form is invoked.

Each form supports a variety of functions. These functions are described in “Here is What You Can Do with OSREQ” on page 7. Subsequent sections present detailed information about coding and invoking the macro to perform these functions. Use of the OSREQ macro must take into consideration both the programming language techniques and the environment in which the program executes. These issues are discussed in “Usage Considerations” on page 28.

Getting the Code Right

The following list summarizes general guidelines for coding the OSREQ macro:

- The OSREQ macro uses only one positional parameter: function. This parameter is always required.
- To invoke OAM functions, the OSREQ macro execute form is always necessary. It must be coded in one of the following ways:
  MF=(E,*parameter_list*)
  MF=(E,*parameter_list*,COMPLETE)

  where *parameter_list* identifies a parameter list area generated using the list form of the OSREQ macro. That area may have been modified previously by the modify form of the OSREQ macro (MF=(M,*parameter_list*)).

  Note: Use either the actual generated list or a copy of it.

  The execute form updates the parameter list area with any parameter values supplied and calls OAM.

  When you specify COMPLETE, the parameter list is zeroed, and nonzero defaults are set before any supplied parameter values are applied.
- Some parameters must be supplied from one or more of the following sources:
  List form
  Modify form
  Execute form

  Parameters must be encoded at least once and must be provided for every invocation of the macro; however, it may not be necessary to explicitly code each parameter for each invocation within an application.
- The following keyword parameters are optional for all OSREQ macro functions, but if specified, are used by all functions:
  MSGAREA
  RETCODE
  REACODE

- The object name that is specified in the name keywords must be fully qualified. Fully qualified names are described in the explanations of the COLLECTN and NAME parameters. See “OSREQ Keyword Parameter Descriptions” on page 23 for descriptions of these and all other OSREQ function parameters.

  Note: The name parameter does not have to be fully qualified when it is used with the QUERY function. Generic names in which the lowest level qualifier of the object name may end in an asterisk are also acceptable.
- Keyword parameters that are not specified in the syntax diagram for a function may be included with that function. The keyword value pointers are established or updated, but the keyword values that are not related to the function are ignored.

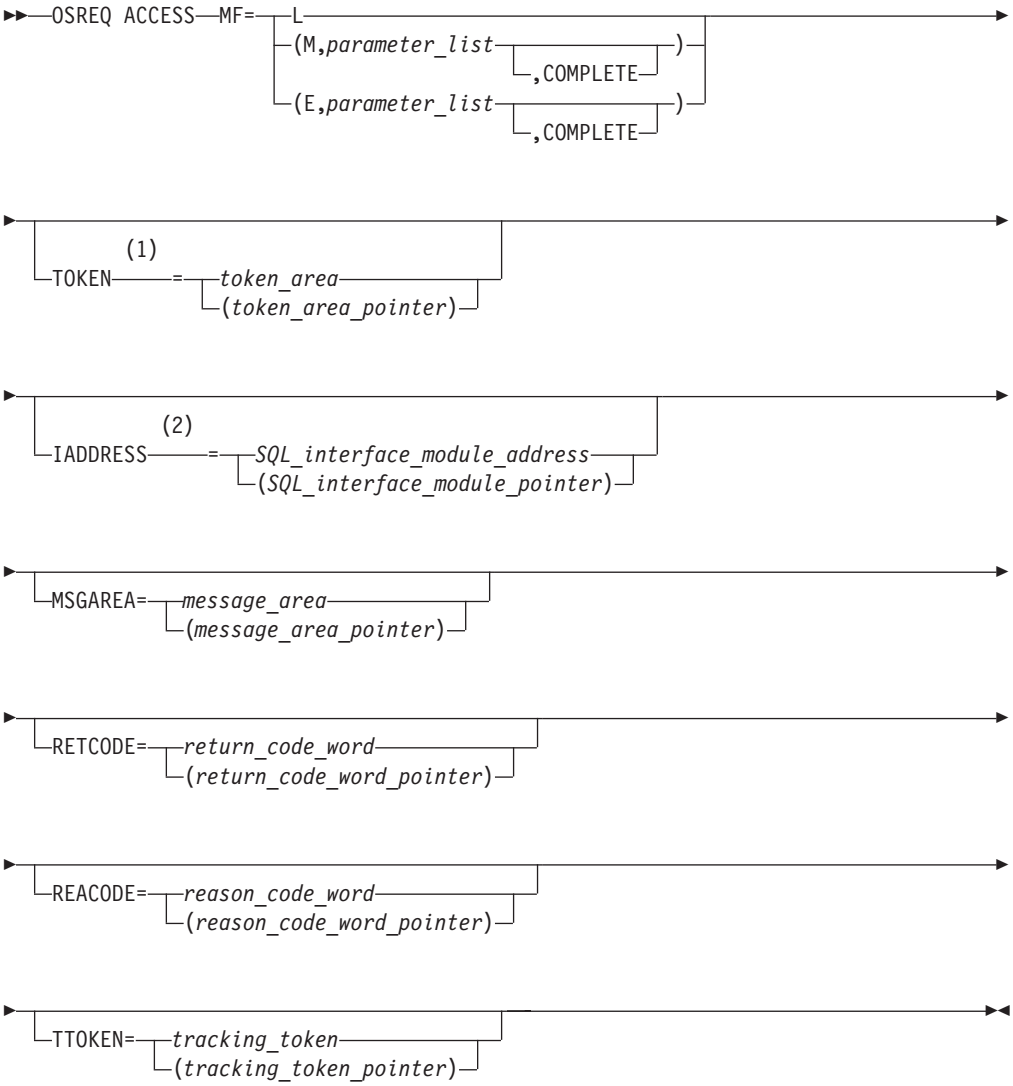
Implementing the Functions

The following alphabetical listing includes the functions that you can perform with the OSREQ macro and instructions for implementing them. A syntax diagram is included with each function. For instructions on reading the syntax diagrams, see “How to Read Syntax Diagrams” on page x. For an explanation of the keyword parameters used in the syntax diagrams, see “OSREQ Keyword Parameter Descriptions” on page 23.

ACCESS—Initializing the OSREQ Interface

The ACCESS function establishes a connection between the caller and OAM. The caller supplies an eight-byte area identified by the TOKEN parameter. ACCESS stores a token into this area. The token set by ACCESS must be specified on all other OSREQ calls. A successful OSREQ ACCESS request must precede any other type of OSREQ request. The syntax diagram for the OSREQ ACCESS function follows.

Syntax for OSREQ ACCESS



Notes:

1 This keyword must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

2 This keyword indicates that a connection to DB2 already exists.

The OSREQ ACCESS function establishes the environmentally-dependent resources needed for other OSREQ function processing in the address space. In environments other than CICS or under the DSN command processor, the DB2 call attachment facility (CAF) is used to establish a connection and open thread between the application unit of work (task) and DB2. This allows for efficient database processing and synchronization of database activities by the application. An exception to this DB2 connection is when the IADDRESS parameter is specified, which is further described below.

In the CICS and DSN command processor environments, the ACCESS function assumes a connection and open thread to DB2 already exists, so CAF services are not needed.

In environments where a connection and open thread to DB2 already exist, but the ACCESS function cannot detect this condition (for example, IMS), the IADDRESS= keyword must be used to specify the structured query language (SQL) interface module entry point address. This address will be used for all SQL processing in the other OSREQ functions. See Figure 2 for the effects of the IADDRESS parameter when used in various processing environments.

PROCESSING ENVIRONMENT	**IADDRESS PARAMETER**	
	SPECIFIED	**NOT SPECIFIED**
IMS	USED	CAF ERROR
MVS BATCH	USED*	CAF SUCCESS
CICS	IGNORED	N/A
DSN Command Processor	IGNORED	N/A
TSO	USED*	CAF SUCCESS
Note: *If the DB2 CONNECT is not done by the application, a DB2 CONNECT and COMMIT will be done for each SQL CALL.		

Figure 2. IADDRESS Parameter Effects in Various Processing Environments

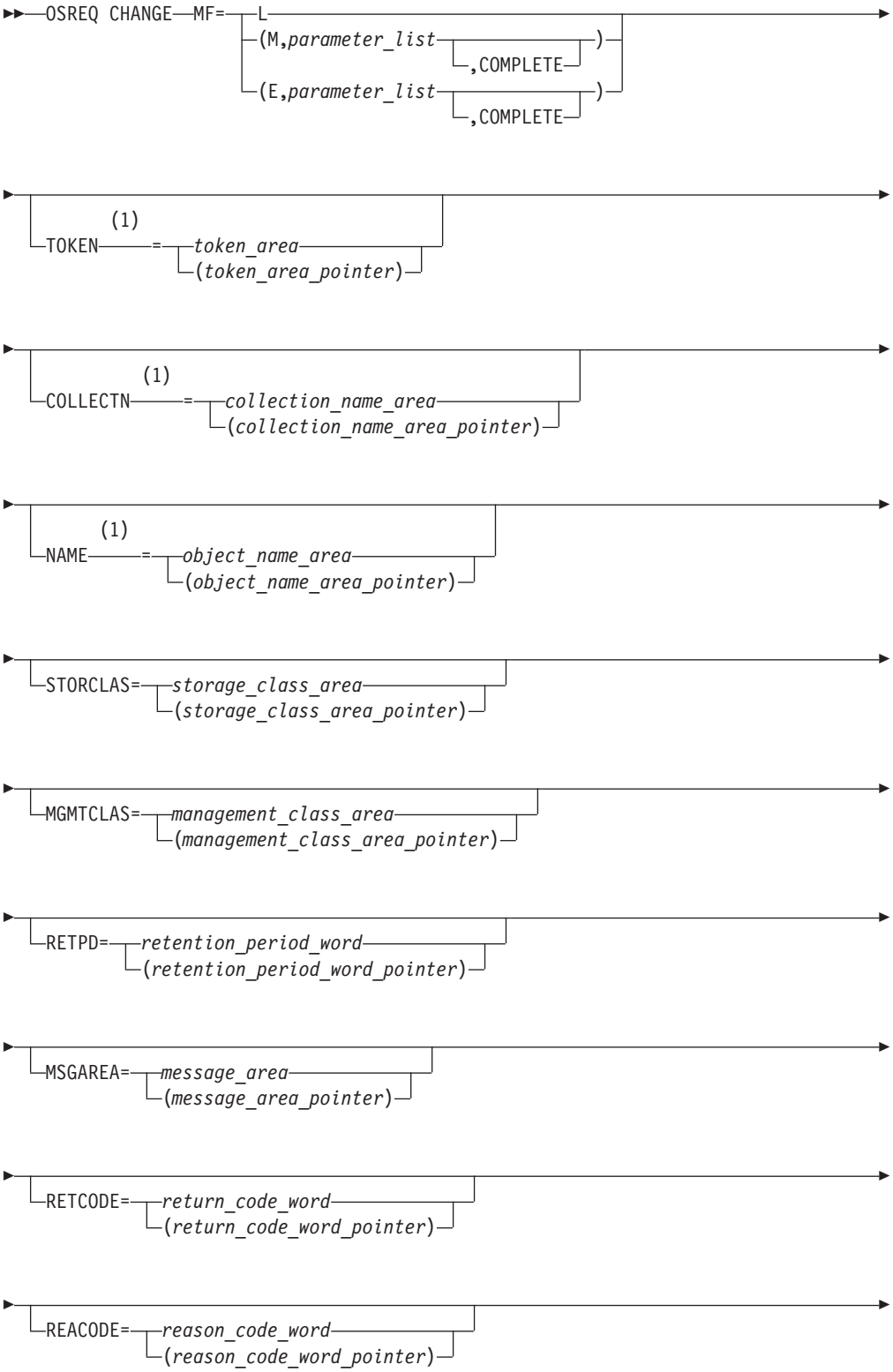
To limit the scope of database activities synchronized by the application, each application should issue its own ACCESS. The application must observe the DB2 restrictions regarding multiple threads from a single task as described in the *DB2 Application Programming and SQL Guide*.

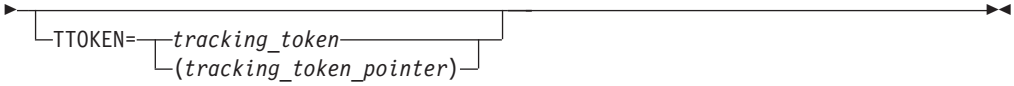
When the calling program no longer requires OSREQ services, it issues the OSREQ UNACCESS request. This clears the token contents. The token cannot be used after OSREQ UNACCESS is issued.

CHANGE—Changing an Object's Management Characteristics

The CHANGE function is used to alter the storage class, management class, or retention period for previously stored objects. A new storage class name, a new management class name, or a new retention period can be specified. Any combination is valid. The specified change is made to the object directory table immediately. The syntax diagram for the OSREQ CHANGE function follows.

Syntax for OSREQ CHANGE





Notes:

1 These keyword parameters must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

As a result of an OSREQ CHANGE, the last referenced date and pending action date of an object are updated to the current date. Because the pending action date is updated, changed objects are scheduled for action during the next storage management cycle. During that cycle, an object may be placed in a different level of the object storage hierarchy to meet a new performance objective. Thus, a new storage class assignment becomes effective during that storage management cycle.

If storage class is specified without management class, the ACS routines either confirm or override the requested storage class assignment. The resulting storage class assignment may be the previously assigned storage class, the requested storage class, or another storage class as determined by the ACS routines. After determining the storage class, the ACS routines determine whether a change in management class is also needed.

If storage class and management class are both specified, first the ACS routines either confirm or override the requested storage class assignment as above and then process the management class. In a method similar to storage class processing, the ACS routines either confirm or override the requested management class assignment. The resulting management class assignment may be the previously assigned management class, the requested management class, or another management class determined by the ACS routines.

If management class is specified without storage class, the ACS routines either confirm or override the requested management class assignment, resulting in assignment of the previous management class, the requested management class, or another management class. The storage class is not affected.

The new management class values obtained through ACS routine processing become the basis for retention period processing.

If the RETPD parameter is specified, a new expiration date is calculated as follows:

- If the object's management class retention limit is zero, the expiration date is not changed unless RETPD was set to −1, in which case the expiration date is set to the reserved value '0001-01-01'. The expiration date for the object is then based solely on the object's management class expiration attributes.
- If RETPD is specified but it is greater than the object's management class retention limit, the expiration date is set to the creation date of the object plus the object's management class retention limit.
- If a RETPD of X'7FFFFFFF' (2 147 483 647) is specified (requesting that the object never expire) and the management class retention limit is NOLIMIT, the expiration date is set to '9999-12-31'.
- If RETPD is specified, the RETPD value is in the range of 1 to 32 767, and none of the above conditions applies, expiration date is set to the creation date of the object plus the number of days specified in the RETPD.

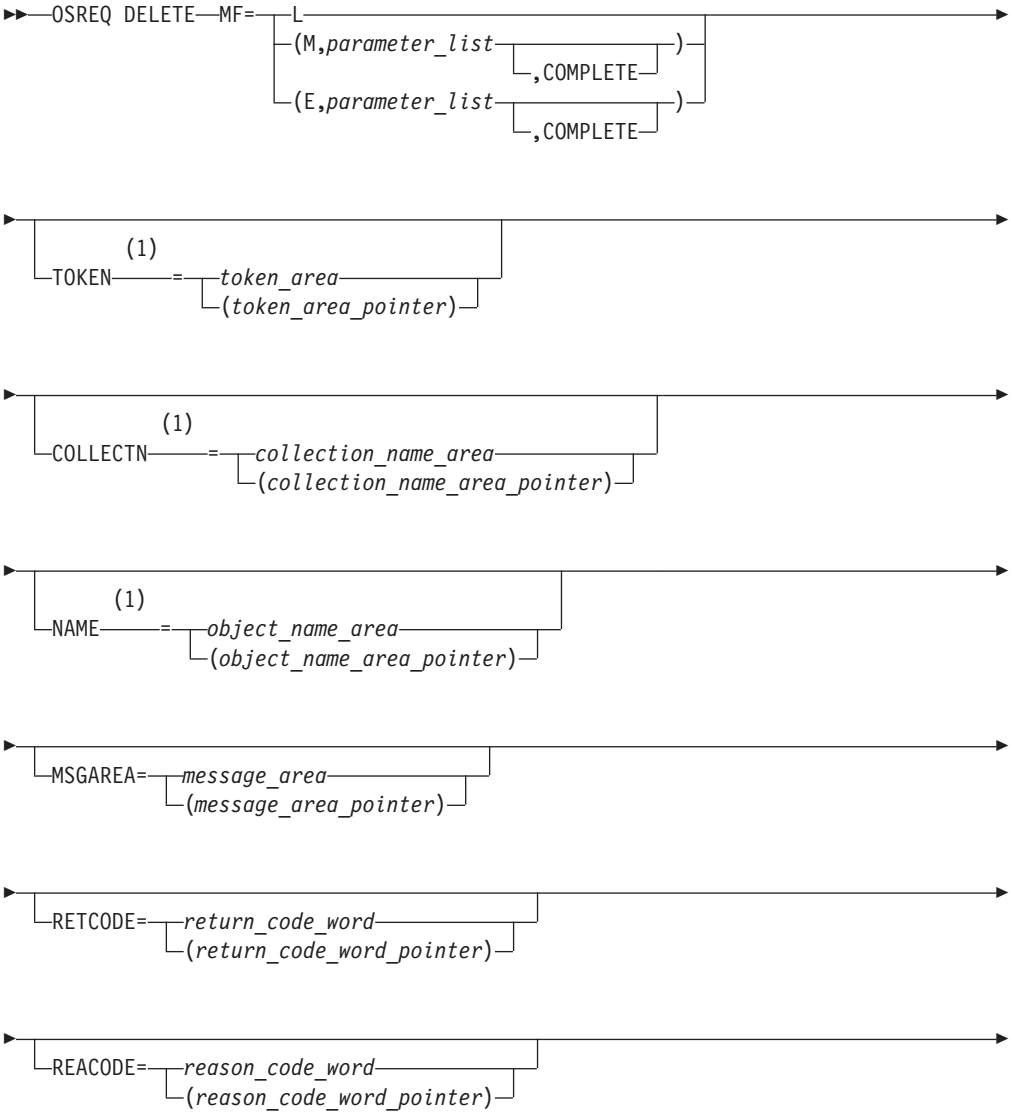
- If RETPD is not specified or is specified as 0 on the OSREQ invocation, then the expiration date is not changed (see Figure 3 on page 32).

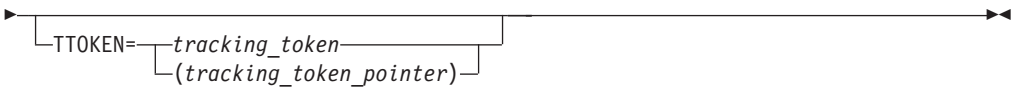
See “Expiration Date Processing” on page 31 for more information.

DELETE—Deleting an Existing Object

The DELETE function removes an object as identified by the COLLECTN and NAME parameters from the object storage hierarchy. The directory information for the object is deleted and all DASD storage used for the object data is released. Primary object data stored on optical, tape, or DASD and backup copies of data stored on optical or tape storage can no longer be referenced. The syntax diagram for the OSREQ DELETE function follows. For further information on the OSMC DASD space management process, refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*.

Syntax for OSREQ DELETE





Notes:

1 These keywords must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

QUERY—Obtaining Object Characteristics

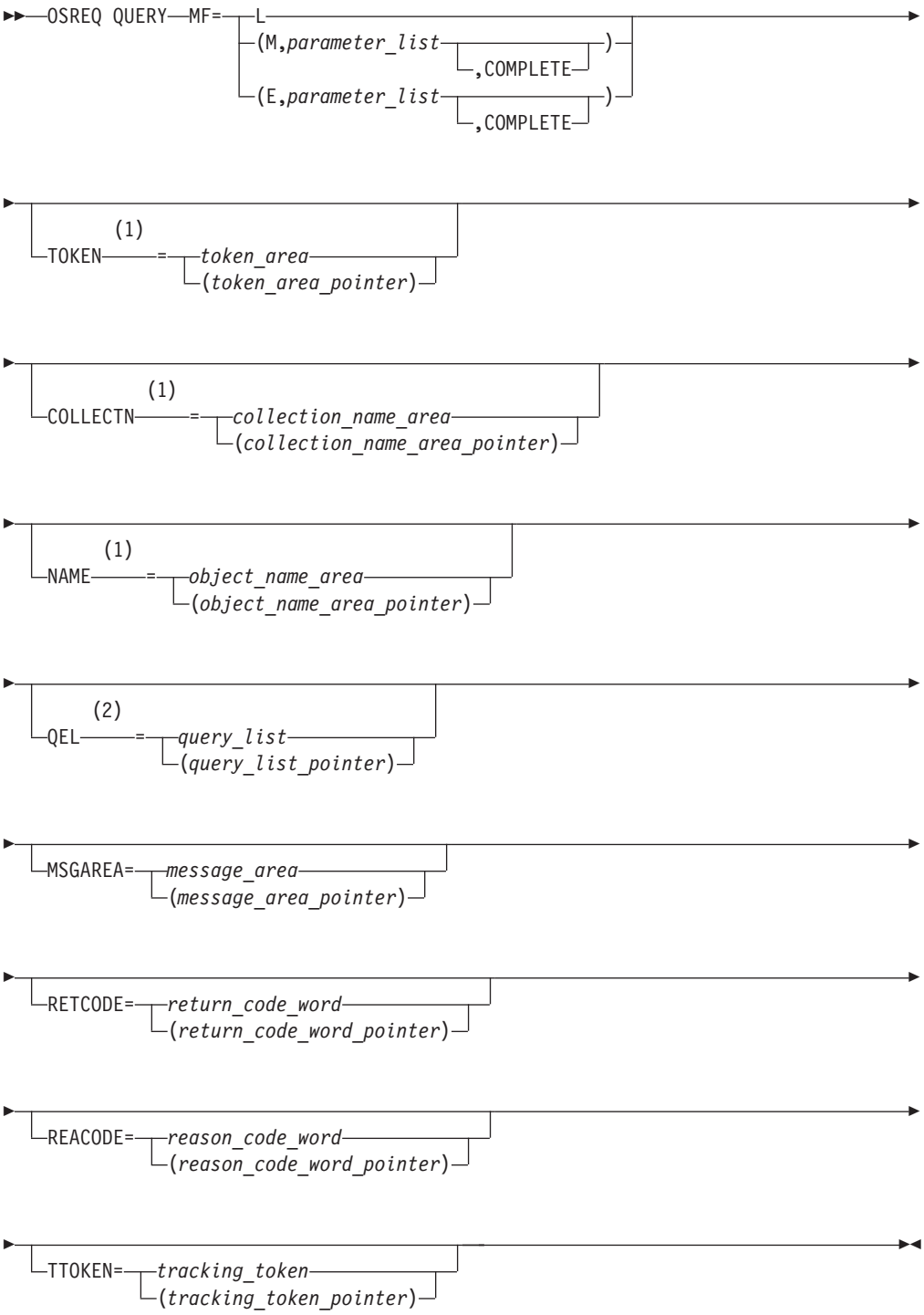
The QUERY function obtains descriptive information about an object within a collection. The object information is presented in query element (QEL) format. The QEL format is described in section “CBRIQEL Macro” on page 35.

QUERY searches the directory containing the objects that belong to the collection name specified in the COLLECTN keyword parameter for a match on the fully qualified object name specified in the NAME keyword parameter, and returns a single query element (QE). QUERY also supports a generic search that returns a QE for each object whose name matches the partially qualified name specified in the NAME keyword.

Request a generic search by substituting an asterisk (*) for the rightmost part of the name (rightmost qualification level). This indicates that the search request applies to all objects whose names match the characters to the left of the asterisk. For instance, MIKES.MAIL.IN is a fully qualified name and results in a single QE when a match is found. The names MIKES.MAIL.* and MIKES.MAIL.PEL* are generic forms and can return multiple QEs when multiple objects exist that match the parts of the names specified. When multiple objects are returned, no ordering can be assumed.

The syntax diagram for the OSREQ QUERY function follows.

Syntax for OSREQ QUERY



Notes:

1 These keywords must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

2 These keywords must be specified on at least one of the forms if the MF=E

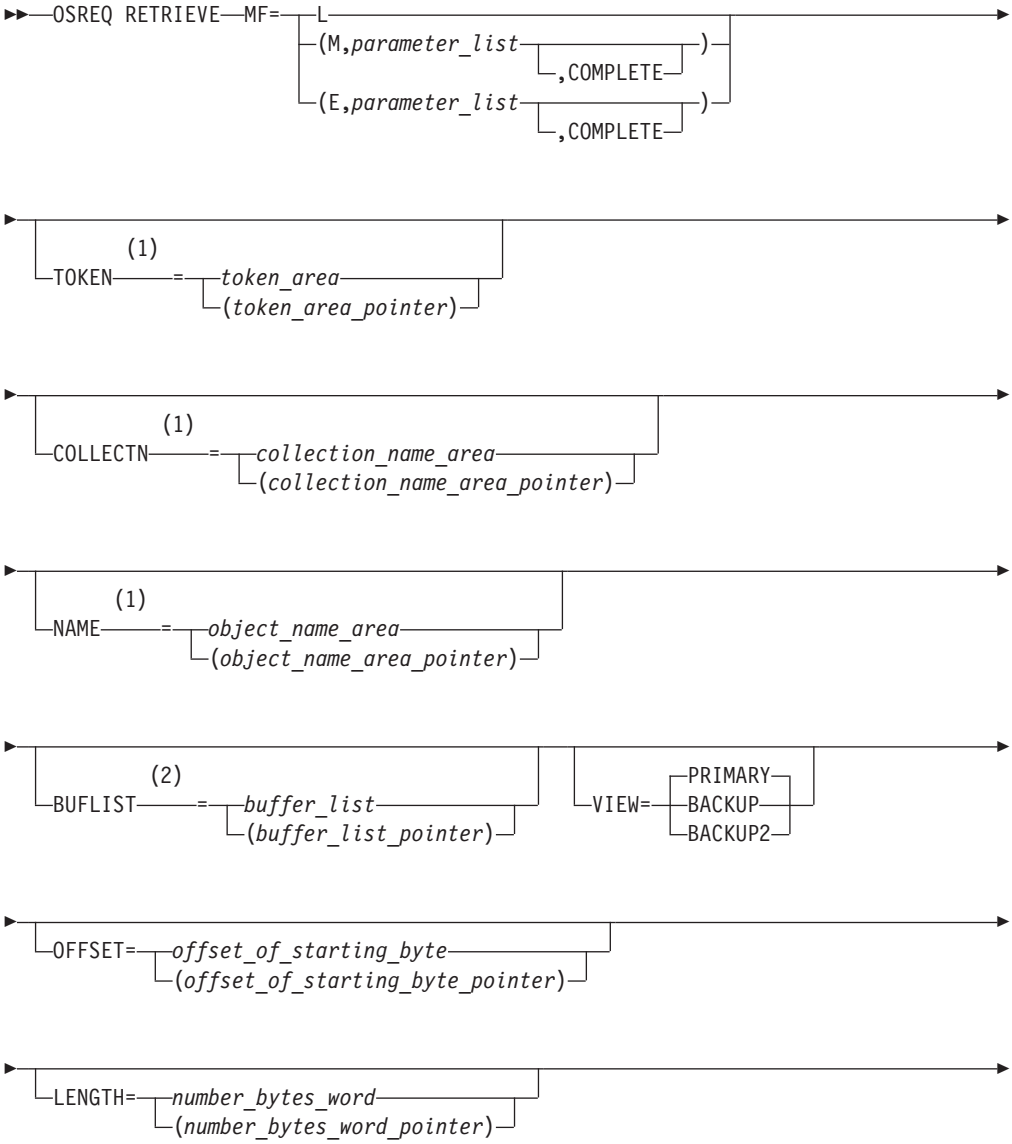
does not indicate COMPLETE. For each buffer specified in *query_list*, the length of the buffer must be specified. The variable *query_list* is described in Figure 6 on page 36.

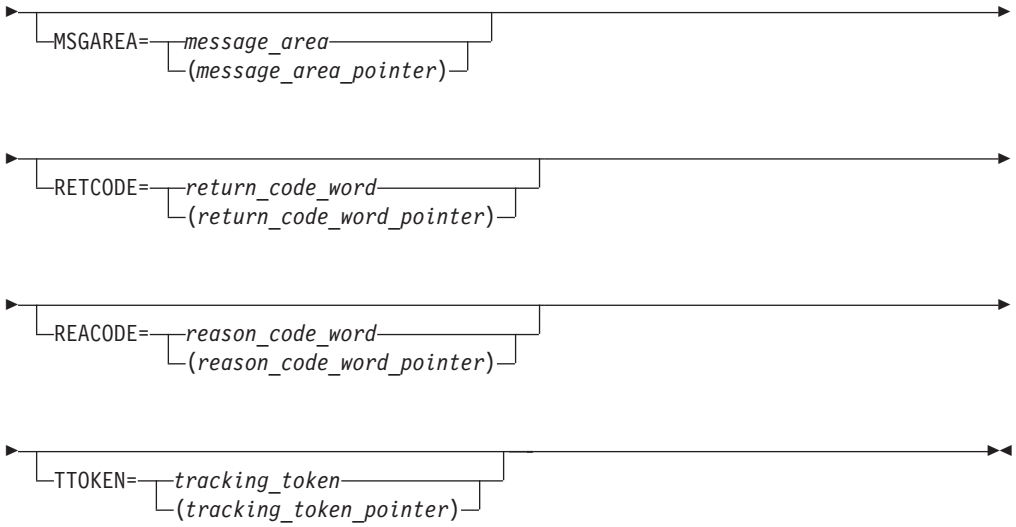
The output of a QUERY request can be used as input to a RETRIEVE request (see "RETRIEVE—Retrieving an Existing Object").

RETRIEVE—Retrieving an Existing Object

The RETRIEVE function locates the primary or backup copy of an object as specified by the COLLECTN, NAME, and VIEW keywords, and returns all or a specified portion of the object to the caller. The syntax diagram for the OSREQ RETRIEVE function follows.

Syntax for OSREQ RETRIEVE





Notes:

1 These keywords must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

2 These keywords must be specified on at least one of the forms if the MF=E does not indicate COMPLETE. For each buffer specified in *buffer_list*, the length of the buffer must be specified. The variable *buffer_list* is described in Figure 4 on page 33.

If the VIEW=PRIMARY function is requested, the object is copied from its place in the object storage hierarchy to the requester's virtual storage buffers that are specified in the BUFLIST keyword. When VIEW=BACKUP is specified, OAM attempts to retrieve the first backup copy of the object from backup optical or tape. When VIEW=BACKUP2 is specified, OAM attempts to retrieve the second backup copy of the object from backup optical or tape. If the specified VIEW function is requested but no object exists, return and reason codes reflect the error (see "Appendix B. Reason Codes" on page 53) and no data is retrieved into the user's buffers.

You may retrieve a copy of the entire object (PRIMARY, BACKUP, or BACKUP2). Alternatively, you may retrieve a specified portion of the object, as defined by the OFFSET and LENGTH keywords. With adequate buffer space supplied by the application, RETRIEVE returns the entire object (or requested portion). If any errors occur during RETRIEVE processing, the buffer contents are invalid.

The RETRIEVE function can use the output from a successful OSREQ QUERY request by using the collection name length field (QELQECNL) as the parameter for the COLLECTN keyword, the object name length field (QELQEONL) as the parameter for the NAME keyword, and by supplying an input buffer of the size noted by object size (QELQEOS).

If you do not specify UPD=N on the CBRINIT statement in the IEFSSN*xx* member of PARMLIB that is used during IPL, the last referenced date and pending action date of a retrieved object are updated to the current date. This schedules the retrieved objects for action during the next storage management cycle. During that

cycle, objects may be placed in a different level in the storage hierarchy to meet new performance objectives, or the objects may not need any processing other than resetting their pending action dates.

If OAM cannot successfully retrieve the object and one or more backup copies exist, the application can use OSREQ RETRIEVE with VIEW=BACKUP or VIEW=BACKUP2 to retrieve the appropriate backup copy. The storage administrator may activate the automatic access backup function to obtain a backup copy of an object when the primary copy of the object is resident on removable media that is unreadable due to disaster or damage. See the *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support* for more information on automatic access backup.

Upon successful completion of object recovery, you can use OSREQ RETRIEVE to retrieve the primary copy of the object.

STORE—Adding an Object

The STORE function adds a complete and unique object to the object storage hierarchy. The application may specify a storage class name, management class name, and retention period, and must specify a collection name and object name. The syntax diagram for the OSREQ STORE function follows.

Objects are stored on an object storage device based on storage class. For more information concerning the selection of media for object storage, refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support.*

The number of bytes specified in the SIZE parameter are written to an object storage device from the buffers specified in the BUFLIST parameter. Objects are removed from the object storage hierarchy based on management class expiration attributes or after their expiration date.

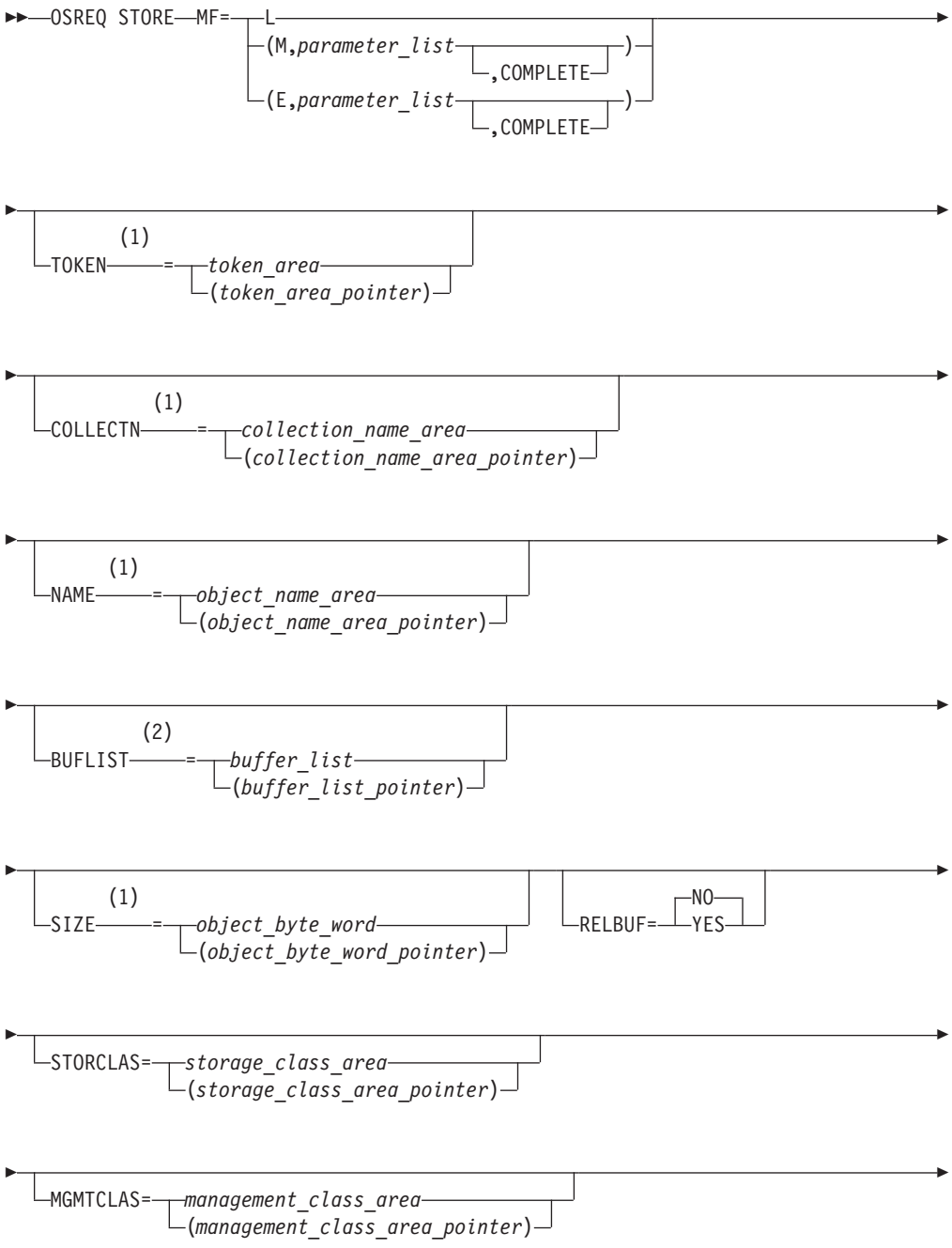
When an object is stored, OAM sets the following date-related fields in the directory entry:

- Set the date last referenced in the object directory to ‘0001-01-01’, which is a reserved value that means that the object has not been referenced yet.
- Set the expiration date:
  - If RETPD is not specified on the OSREQ request, the expiration date is set to the reserved value ‘0001-01-01’. The expiration date for the object is then based solely on the object’s management class expiration attributes.
  - If the object’s management class retention limit is zero or if the retention period is 0 or −1, the expiration date is set to the reserved value ‘0001-01-01’ (see Figure 3 on page 32 for more information).
  - If RETPD is specified but it is greater than the object’s management class retention limit, the expiration date is set to the creation date of the object plus the object’s management class retention limit.
  - If a RETPD of X'7FFFFFFF' (2 147 483 647) is specified (requesting that the object never expire) and the management class retention limit is NOLIMIT, the expiration date is set to ‘9999-12-31’.
  - If RETPD is specified, the RETPD value is in the range of 1 to 32 767, and none of the above conditions apply, expiration date is set to the creation date of the object plus the number of days specified in the RETPD.
  - Set the creation timestamp to the current date/timestamp.

– Set the pending action date to the current date so that the object is selected for processing during the next storage management cycle.
– Set the management class assignment date to the current date.

See “Expiration Date Processing” on page 31 for more information.

Syntax for OSREQ STORE



Notes:

1 These keywords must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

2 These keywords must be specified on at least one of the forms if the MF=E does not indicate COMPLETE. For each buffer specified in *buffer_list*, the length of the buffer must be specified. The *buffer_list* variable is described in Figure 4 on page 33.

Processing a STORE to a New Collection

If the OSREQ STORE request specifies a new collection name, an MVS catalog entry is created for the collection. The MVS catalog entry contains the names of the management class and storage class to be used as default assignments for objects added to the collection. The management class and storage class names are determined by the ACS routines as follows:

- If storage class and management class names are not specified in the OSREQ STORE request, the ACS routines determine the storage class and management class names to be used as the default assignments for the collection.
- If storage class and management class are specified in the OSREQ STORE request, the names are provided to the ACS routines, which either confirms or overrides the assignments as the default storage class and management class assignments for the collection.
- If storage class is specified without management class, the storage class name is provided to the ACS routines, which either confirms or overrides the assignment, and then determines the default management class assignment for the collection.
- If management class is specified without storage class, the ACS routines determines the default storage class assignment. The management class name is provided to the ACS routines, which either confirms or overrides the management class assignment.

Processing a STORE to an Existing Collection

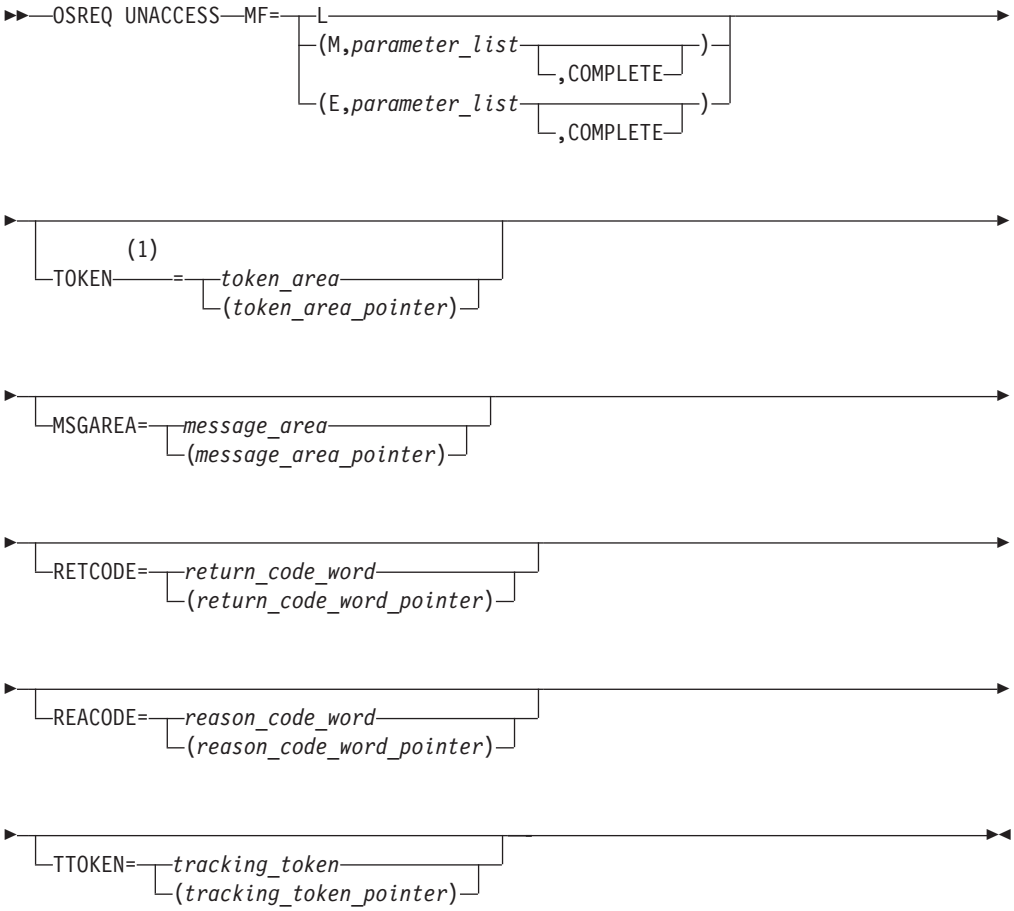
If the STORE function is requested for an existing collection name or is requested after the new collection name MVS catalog entry has been defined, the actual storing of the object is completed. The initial storage class and management class assignments are stored in the directory entry created for the object. The initial class assignments are determined as follows:

- If the management class and storage class are not specified on the OSREQ STORE request, the default assignments contained in the MVS catalog entry for the collection are used as the assignments for the object.
- If management class and storage class are specified in the OSREQ STORE request, the names are provided to the ACS routines, which either confirm or override the assignments as the initial storage class and management class assignments for the object.
- If storage class is specified without management class, the storage class name is provided to the ACS routines, which either confirms or overrides the assignment, and then determines the initial management class assignment for the object.
- If management class is specified without storage class, the ACS routines determine the initial storage class assignment. The management class name is provided to the ACS routines, which either confirms or overrides the management class assignment.

UNACCESS—Ending the OSREQ Interface

The UNACCESS function ends the connection between the application program and OAM. When the calling program no longer requires OSREQ services, it must issue OSREQ UNACCESS. When invoking UNACCESS, the caller supplies an eight-byte token that has been set by a successful issuance of OSREQ ACCESS. UNACCESS should not be requested unless the corresponding ACCESS was successful. An initialized token is required by all OSREQ calls, except ACCESS. The syntax diagram for the OSREQ UNACCESS function follows.

Syntax for OSREQ UNACCESS



Notes:

1 This keyword must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

OSREQ UNACCESS does not attempt to end any active requests that are using the same token, but returns control to the UNACCESS caller with a warning return code and reason code. When each of the outstanding requests completes, any further OSREQ requests using that token receive return and reason codes indicating that the token is no longer valid.

OSREQ Keyword Parameter Descriptions

This section describes the OSREQ macro keyword parameters as they generally pertain to all operations. The values in parentheses identify a register that contains the address of the parameter (not applicable when using the OSREQ macro list form). Restrictions and limitations may apply for some operations, and they are explained separately under each operation. The keywords are listed alphabetically.

BUFLIST=*buffer_list*

BUFLIST=*(buffer_list_pointer)*

buffer_list specifies the name of a variable or expression defining an area that has the format described by the CBRIBUFL macro. See “CBRIBUFL Macro” on page 33.

COLLECTN=*collection_name_area*

COLLECTN=*(collection_name_area_pointer)*

collection_name_area specifies a variable-length field. This area contains a fully qualified collection name. The first two bytes specify the number of characters that follow; the maximum value is the maximum length of a standard MVS data set name. A name consists of one to 21 parts. Each part is separated from the next part by a period (X'4B'). Each part must start with an uppercase alphabetic, #, $, or @ character. Each part can contain one to eight uppercase alphanumeric, #, $, or @ characters. Each part of the name after the first period is often referred to as a qualification level. Any disallowed character causes a parameter error return code (except for blanks to the right of the name).

IADDRESS=*SQL_interface_module_address*

IADDRESS=*(SQL_interface_module_address_pointer)*

SQL_interface_module_address specifies the entry point of the address of the DB2 (or equivalent) SQL interface module (for example, the DFSLI000 and the DSNALI interface modules). For details on the use of these modules, refer to the *DB2 Administration Guide*. This parameter must directly identify the entry point address instead of acting as a pointer to a fullword that contains the entry point address. The use of the IADDRESS keyword implies to the OSREQ interface that the environment is not CICS nor DSN and that the DB2 connection and thread are controlled by the application or by the environment in which the application is running.

LENGTH=*number_bytes_word*

LENGTH=*(number_bytes_word_pointer)*

number_bytes_word specifies a fullword that indicates how many bytes of the object are retrieved. It is used with the OFFSET keyword to retrieve part of an object. The LENGTH keyword is an optional parameter, which is used only on a RETRIEVE request. It is ignored on all other requests.

If a LENGTH value of zero is specified, or if the LENGTH parameter is omitted on a RETRIEVE request, the length defaults to the remaining portion of the object (that is, from the OFFSET to the end of the object). If the length specified is negative or is greater than the remaining portion of the object, a return code and a reason code indicating the error are returned; the object is not retrieved.

MF

The MF (macro form) keyword parameter uses several operands to indicate which form of the macro is to be invoked. The forms and their associated operands are as follows:

- **MF=L**

  The list macro form generates a parameter list suitable for use with the MF keyword on the execute and modify forms of the macro. The label position of the

list form of the macro becomes the label of the generated parameter list. The parameter list is a modifiable area of storage in the caller's key, 96 bytes in length.

- **MF=(M,parameter_list[,COMPLETE])**

  The modify macro form updates *parameter_list* with the other parameters specified on the macro statement.

- **MF=(E,parameter_list[,COMPLETE])**

  The execute macro form updates *parameter_list* with the other parameters specified on the macro statement and initiates execution of the request.

When you specify COMPLETE, the parameter list is zeroed, and nonzero defaults are set before any supplied parameter values are applied. In this case, required parameters that are not specified for the requested function on the MF=E form of the macro are flagged as errors during assembly of the macro.

MGMTCLAS=*management_class_area*

MGMTCLAS=*(management_class_area_pointer)*

management_class_area specifies a variable-length field containing a two-byte length field, followed by a variable-length name field containing a name identified to z/OS as a management class name. The first two bytes specify the number of characters that follow, not including the length field itself. The length-field value can be from zero to the maximum length allowed for z/OS management class names. The name must be left-justified in the name field and can be padded on the right with blanks. If the length includes trailing blanks, only the name characters up to the trailing blanks are used. Specifying a length value of zero or filling the name field with blanks is equivalent to omitting this parameter.

MSGAREA=*message_area*

MSGAREA=*(message_area_pointer)*

message_area specifies an optional variable-length message area that contains a length field followed by a message data area. This message data area is used for message data that may accompany return codes from DB2. Message data is placed in the message data area, and any message data that exceeds the available space is truncated.

The first two bytes of the message area contain a length value equal to the length of the message data area immediately following the first two bytes, but not including the length field itself. The second two-byte field (first two bytes of the message data area) contains the length of the message data returned, including the two bytes for the second length field. A suggested initial message area length is 1024 bytes. The minimum value for the message area length is 244 bytes.

Note: Not all errors have corresponding message data.

NAME=*object_name_area*

NAME=*(object_name_area_pointer)*

object_name_area specifies a variable-length field. This area contains a fully qualified object name (except when used in conjunction with the OSREQ QUERY function which allows the use of generic names). The first two bytes specify the

number of characters that follow; the maximum value is the maximum length of a standard MVS data set name. A name consists of 1 to 21 parts. Each part is separated from the next part by a period (X'4B'). Each part must start with an uppercase alphabetic, #, $, or @ character. Each part can contain one to eight uppercase alphanumeric, #, $, or @ characters. Each part of the name after the first period is often referred to as a qualification level. Any disallowed character causes a parameter error return code (except for blanks to the right of the name). For an OSREQ QUERY, one asterisk (X'5C') can be substituted for the rightmost characters of the rightmost part of the name (rightmost qualification level) to indicate that the search request applies to all objects whose names match the characters to the left of the asterisk.

OFFSET=*offset_of_starting_byte*

OFFSET=*(offset_of_starting_byte_pointer)*

offset_of_starting_byte is a fullword that specifies the offset of the first byte to be retrieved. The first byte of the object has an offset of zero, the second byte has an offset of one, and so on. The OFFSET keyword is only used by a RETRIEVE request and is ignored on all other requests.

If the OFFSET parameter is omitted on a RETRIEVE request, the offset defaults to the beginning of the object (that is, OFFSET=0). If the offset specified is negative or past the end of object, a return code and a reason code are returned, indicating the error; the object is not retrieved.

QEL=*query_list*

QEL=*(query_list_pointer)*

query_list specifies the name of a variable or an expression defining an area that has the format described by the CBRIQEL macro. See “CBRIQEL Macro” on page 35.

REACODE=*reason_code_word*

REACODE=*(reason_code_word_pointer)*

reason_code_word specifies an optional area into which the reason code value is to be copied. The reason code value is always in register 0. In order to determine the success or failure of an OSREQ request, the programmer should check the reason code in register 0.

Note: There are conditions under which the *reason_code_word* is not set, such as the *reason_code_word* area is invalid or a major error occurs before the *reason_code_word* area has been validated. The reason code value is always returned to register 0.

RELBUF=YES

RELBUF=NO

The RELBUF keyword indicates the disposition of the data in the buffers that are specified for a STORE operation. RELBUF=NO indicates that the data in the buffers will be retained by the system. After the data is stored on the requested media, RELBUF=YES indicates that the pages containing the data in the buffers may be

discarded by the system and not restored when the respective pages are later referenced. This use of RELBUF often improves performance by saving I/O operations for paging data. RELBUF=NO is the default.

Attention: RELBUF=YES may release pages that contain data that has not been committed to the database.

RETCODE=*return_code_word*

RETCODE=*(return_code_word_pointer)*

return_code_word is an area into which the return code value is copied. The return code value is always in register 15. In order to determine the success or failure of an OSREQ request, the programmer should check the return code in register 15.

Note: There are conditions under which the *return_code_word* is not set, such as the *return_code_word* area is invalid or a major error occurs before the *return_code_word* area has been validated. The return code value will always be returned to register 15.

RETPD=*retention_period_word*

RETPD=*(retention_period_word_pointer)*

retention_period_word specifies a fullword or an expression that contains the override retention period. See Figure 3 on page 32 for valid retention periods.

SIZE=*object_byte_word*

SIZE=*(object_byte_word_pointer)*

object_byte_word specifies a fullword that contains the total object length in bytes. The maximum size is 50 megabytes (52 428 800 bytes).

STORCLAS=*storage_class_area*

STORCLAS=*(storage_class_area_pointer)*

storage_class_area specifies a variable-length field containing a two-byte length field, followed by a variable-length name field containing a name identified to z/OS as a storage class name. The first two bytes specify the number of characters that follow, not including the length field itself. The length-field value can be from zero to the maximum length allowed for z/OS storage class names. The name must be left-justified in the name field and can be padded on the right with blanks. If the length includes trailing blanks, only the name characters up to the trailing blanks are used. Specifying a length value of zero or filling the name field with blanks is equivalent to omitting this parameter.

TOKEN=*token_area*

TOKEN=*(token_area_pointer)*

token_area specifies an eight-byte area on a word boundary into which OSREQ ACCESS stores a value. Token_area must be specified on all other issuances of OSREQ. The token becomes invalid after OSREQ UNACCESS is issued.

TTOKEN=*tracking_token*

TTOKEN=*(tracking_token_pointer)*

tracking_token specifies a 16-byte area containing a tracking token. The contents of the tracking token may be any user-supplied information. The tracking token supplied on the OSREQ macro with the TTOKEN keyword will be placed in the OAM System Management Facility (SMF) record, in the ST1TTOK field for record subtypes 1 through 7. If no tracking token is supplied on the OSREQ macro, the ST1TTOK field in record subtypes 1 through 7 will contain binary zeros. For information concerning SMF recording, refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*.

VIEW=PRIMARY

VIEW=BACKUP

VIEW=BACKUP2

The VIEW parameter specifies which copy of an object is obtained during a RETRIEVE. If VIEW=PRIMARY, OAM retrieves the primary copy of the object. If VIEW=BACKUP, OAM retrieves the backup copy. If VIEW=BACKUP2, OAM retrieves the second backup copy. If the specified copy of the object does not exist, return and reason codes reflect this error (see “Appendix B. Reason Codes” on page 53); no data is returned. The VIEW keyword is only applicable to RETRIEVE requests and is ignored on all other requests. VIEW=PRIMARY is the default.

Usage Considerations

Use of the OSREQ macro must take into consideration both the programming language techniques and the environment in which the program executes. The following list summarizes those considerations:

- Any or all parameters can be supplied on any form of the OSREQ macro (MF=L, MF=M, or MF=E). When you specify a parameter, a pointer to that parameter is placed in the parameter list. This does not mean that the parameter pointer or the parameter value is validity-checked for all requested functions. Only parameters required by the specific function are checked for validity.
- Because parameters not relevant to the current function are ignored, parameters specified on the MF=L form of the OSREQ macro can remain set for all following OSREQ macro functions that use the same parameter list, unless the COMPLETE operand is specified. In this way, parameter values can be altered as needed, but parameter pointers do not need to be updated by subsequent forms of the OSREQ macro. This can reduce some of the inline code created by the macro.
- When you use the COMPLETE operand on the MF=M or MF=E forms of the OSREQ macro, the entire parameter list is cleared and initialized; then, specified parameter pointers are placed in the parameter list. The only way for the OSREQ macro to verify that all required parameters are supplied is to use the MF=(E,parameter_list,COMPLETE) form; however, additional inline code is generated by using the COMPLETE operand.
- The TOKEN parameter of the OSREQ macro must be supplied by the MF=E form or one of the previous invocations of the MF=L or MF=M forms. If the TOKEN parameter is not specified or if an invalid token-area address is specified, the MF=E form of the OSREQ macro specifying any function other

than ACCESS produces unpredictable results (generally abnormal termination). ACCESS identifies an invalid token area with appropriate return codes and reason codes.

- The IADDRESS is an optional parameter that is valid only for an OSREQ ACCESS function. The IADDRESS=*keyword* parameter is ignored for all other OSREQ functions. If the application does not specify IADDRESS with an ACCESS function, then OAM determines the execution environment. OAM uses the appropriate DB2 language interface module consistent with the execution environment when performing DB2 functions on behalf of the application.
- The OSREQ macro uses several literal values. It may be necessary to insert a LTORG in the assembly code so that the created literals are addressable at the point where the OSREQ macro is used.
- The user of the OSREQ macro must request the ACCESS function before any other functions are requested. The user must request the UNACCESS function when OAM processing is complete.
- When you are using the OSREQ macro in environments similar to CICS, where all processing is done under one task control block (TCB), it is permissible for one subroutine (or transaction) to request the ACCESS function and to pass a pointer to the token to other subroutines (or transactions) that will need that token for other functions. Passing a copy of the token itself from one subroutine (or transaction) to another can produce unpredictable results.

  Note: All processing *must* be done under the same TCB that issued the ACCESS. The token cannot be used by more than one task.
- When the OSREQ macro is used in multitasking environments, each task must request its own OSREQ ACCESS, and all functions within that task must use the same token, not separate copies of the token.

Usage Requirements

The following requirements must be met in order to use the OSREQ macro successfully:

- The caller must be in task mode, 31-bit addressing mode, primary addressing mode, problem or supervisor state, and any storage protect key. (Callers may not be in cross-memory mode.)
- The calling program cannot hold any MVS locks.
- All input and output parameters must be contained within the home address space and must be accessible in primary addressing mode.
- The DB2 subsystem must be running and, if CICS is used, it must be connected to DB2. The installation is responsible for starting the DB2 subsystem and establishing the connection.
- The call attachment facility is used by OAM in the MVS batch environment to connect to DB2 during the ACCESS call to OAM. After the connection is made to DB2, a thread is established (via OPEN) to plan CBRIDBS. The call to ACCESS should be invoked prior to any application DB2 activities occurring to allow synchronization with the OAM database activities. Synchronization is the responsibility of the application and is in the form of CLOSE, then OPEN, as described in the *DB2 CAF User's Guide and Reference* manual.
- In the CICS, DSN Command Processor, and IMS environments, it is assumed that the connection to DB2 has already been made. Synchronization in CICS is accomplished through the use of the SYNCPOINT function (refer to the *DB2 Application Programming and SQL Guide*). In the TSO environment, synchronization is accomplished through the use of COMMIT and ROLLBACK

functions, as described in the *DB2 SQL Reference*. In the IMS environment, synchronization is accomplished through the use of COMMIT and ROLLBACK functions (see the *DB2 SQL Reference* manual), or by the use of SYNC and ROLL/B call to IMS.

- If you use JOBLIB or STEPLIB JCL statements in your application that include DB2 load modules, then the entire JOBLIB or STEPLIB concatenation must be assigned to authorized libraries. Because the OSREQ application programming interface runs in an authorized state, it must load the DB2 modules at the time the ACCESS function is invoked. MVS requires that all libraries in a concatenation must be authorized when the loading program is authorized.

Restrictions and Limitations

OAM supports a maximum object size of 50 megabytes (52 428 800 bytes). The minimum message area size is 244 bytes.

Programming Notes

The programming notes that follow may be relevant as you code your application interface:

- Optional input parameters on the OSREQ macro may be omitted. OAM processing identifies omitted optional input parameters as follows:
  - If the optional input parameter has not been specified on any of the OSREQ macro forms (MF=L, MF=M, or MF=E), the parameter pointer is zero.
  - If the optional input parameter is specified on one of the OSREQ macro forms but the value identified by the parameter is null, then the parameter has the appropriate null value. The concept of null is different for different parameters. A null RETPD parameter value is zero. A null STORCLAS parameter value is indicated by either a length value of zero or the entire name containing blanks.
  - If the optional input parameters MGMTCLAS and STORCLAS are omitted, these parameter values are supplied by the ACS routines, as described in “OSREQ Keyword Parameter Descriptions” on page 23.
- If you do not specify a collection name on any function other than ACCESS or UNACCESS, a return code and a reason code are generated, and the requested function is not performed. The collection name is required if the function is to be completed. If a specified collection name does not exist in the catalog for any function other than STORE, ACCESS, or UNACCESS, a return code and a reason code are generated.
- When an MVS catalog entry is created for a new collection on a STORE function or the specified storage class or management class is overridden by the ACS routines, a warning return code of 4 and a reason code with the fourth byte indicating the processing status are generated. The conditions are possible in all combinations. The processing status in the fourth byte of the reason code contains individual bits that indicate the presence or absence of each of the conditions.
- The caller must establish synchronization points for DB2 inserts, updates, and deletes for the OSREQ functions STORE, DELETE, CHANGE, and RETRIEVE as soon as possible (to minimize DB2 timeouts or deadlocks), depending on return code.
- In order to allow your application to establish synchronization points in DB2, the DBRM from your application program must be bound in the CBRIDBS plan. The SAMPLIB job CBRIBIND (or CBRIBIND for DASD-only users) is used to create the CBRIDBS plan in DB2. For more information on the CBRIBIND job and

CBRIDBS plan, refer to the *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*.

If your application uses the IADDRESS keyword, the application connection to DB2 must be established and have an open thread. The plan identified for the open thread can include any DBRMs or packages that are needed by the application. However, it must also contain the DB2 packages created by the CBRIBIND job for the CBRIDBS plan. For more information on the bind jobs or on the DB2 plans, refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support*.

- If the OSREQ macro is invoked and either the OSREQ parameter list or the token area is in nonaddressable storage, a program check occurs within the executable OSREQ macro code. For diagnostic purposes, the potential reason code for the specific error is preloaded into register 0 before storage is accessed. The register 0 contents in the abend summary should contain a reason code that indicates the parameter or storage problem. This also applies if the token contents have been corrupted before invoking the OSREQ macro.
- If the return code word or reason code word are not located in addressable storage, the return and reason codes are only found in general registers 15 and 0, respectively, upon return from OSREQ.

Register Use

When the OSREQ macro is invoked, register 13 must contain the address of a standard 18-word save area.

Registers 0, 1, 14, and 15 are used by the OSREQ macro. At exit, the contents of the registers are as follows:

0 Reason code

1 Unpredictable

2–13 Unchanged

14 Unpredictable, except for ACCESS and UNACCESS, when it remains unchanged

15 Return code

Expiration Date Processing

The expiration date is the date on which OAM can delete objects automatically. The expiration date is based on the retention period (RETPD) specified on OSREQ STORE or CHANGE or on the object's management class retention limit. The expiration date in the object's directory entry is set to the reserved value of '0001-01-01' when the object has no explicit expiration date. In this case, the expiration of the object is based on the object's management class expiration attributes. The object's management class referred to in this section is the actual management class for the object after review and possible override by the automatic class selection routine, which could be different from the management class specified on the OSREQ macro.

Figure 3 on page 32 shows the processing of the values that may be specified on the RETPD parameter and the resulting expiration date. RETPD values in the range of 1 to 32 767 and the special value X'7FFFFFFF' (2 147 483 647) may be overridden. If the RETPD parameter value exceeds the management class retention limit, the management class retention limit is used to determine the expiration date.

For the special parameter value X'7FFFFFFF' (2 147 483 647) to be effective, the management class retention limit must be set to NOLIMIT.

Specified RETPD Parameter Value	Requested Expiration Date STORE	Requested Expiration Date CHANGE
0 or retention period parameter not specified (Null)	Set expiration date to 0001-01-01 and use management class attributes to determine expiration date.	Use existing expiration information for this object.
−1	Set expiration date to 0001-01-01 and use management class attributes to determine expiration date.	Reset expiration date to 0001-01-01 and use management class attributes to determine expiration date.
1 to 32 767	Expiration date is set to the sum of the object creation date and RETPD parameter value.	Expiration date is set to the sum of the object creation date and RETPD parameter value.
X'7FFFFFFF' (2 147 483 647)	9999-12-31	9999-12-31
Any other value	These values are invalid. Return and reason codes are returned to the caller.	These values are invalid. Return and reason codes are returned to the caller.

Figure 3. Valid Retention Periods for Expiration Date Processing

Messages and Codes

OAM generates return codes and reason codes in response to errors detected during the processing of OSREQ requests. While operating under control of the calling transaction, OAM does not generate any messages to the operator, system programmer, or storage administrator.

Return Codes and Reason Codes

OAM issues return codes 0, 4, 8, C, and 10 (hexadecimal). These return codes are accompanied by reason codes that define the error encountered. See “Appendix B. Reason Codes” on page 53 for a table of return codes and their associated reason codes.

The return codes are defined as follows:

0 The requested function was successfully completed. Recommended program action: None required.

4 The requested function was completed with a warning condition. Recommended program action: Correct program, if necessary.

8 The requested function was not completed due to an application programming error. Recommended program action: Write an error message to the operator (system console, CICS, or IMS master terminal) that includes the return code and reason code.

C The requested function was not completed due to an environmental error. Recommended program action: Write an error message to the operator (system console, CICS, or IMS master terminal) that includes the return code and reason code.

10 The requested function was not completed due to an OAM programming error. Recommended program action: Write an error message to the operator (system console, CICS, or IMS master terminal) that includes the return code and reason code.

CBRIBUFL Macro

The CBRIBUFL macro describes the area to which the BUFLIST keyword on the OSREQ macro points. The area contains a header and a list of buffer descriptors. Each buffer descriptor describes one data buffer, giving the address of the buffer, the length of the buffer, and the amount of data in the buffer. The data buffer contains the data for the object to be stored or provides the buffer space for the object to be retrieved.

The CBRIBUFL macro is a mapping macro consisting of three DSECTs. The first two DSECTs are used to describe the buffer list. The third DSECT maps the data buffer pointed to by the buffer list. Figure 4 and Figure 5 on page 34 describe the contents of the DSECTs.

```
     OBL       DSECT      Data buffer list control block
               DS   0F
+0   OBLID     DS   CL4    Control block identifier ('OBL ')
+4   OBLLSTL   DS   F      Length of buffer list cb in bytes
                          including buffer descriptors
+8   OBLVERSN  DS   XL1    Buffer list version (X'02')
+9             DS   XL3    Reserved, must be zero
+12            DS   F      Reserved, must be zero
+16  OBLNUMBF  DS   F      Number of data buffer descriptors that
                          follow
+20  OBLBUFL   DS   0F     Beginning of data buffer descriptor list,
                          mapped by OBLBDESC
```

The following buffer descriptor is repeated for each data buffer:

```
     OBLBDESC  DSECT      Data buffer descriptor
+0   OBLBUFP   DS   A      Address of buffer
+4   OBLBBLTH  DS   F      Length of buffer
+8   OBLBUSED  DS   F      Length of data in buffer
+12            DS   F      Reserved, must be zero
```

Each data buffer is described as follows:

```
     OBLB      DSECT      Data buffer
               DS   0F
+0   OBLBDATA  DS   0X     Object data area
```

Figure 4. Fields Described by CBRIBUFL

Figure 5 on page 34 is a structure diagram of the data buffer list (CBRIBUFL) pointed to by the BUFLIST keyword on an OSREQ STORE or OSREQ RETRIEVE macro.

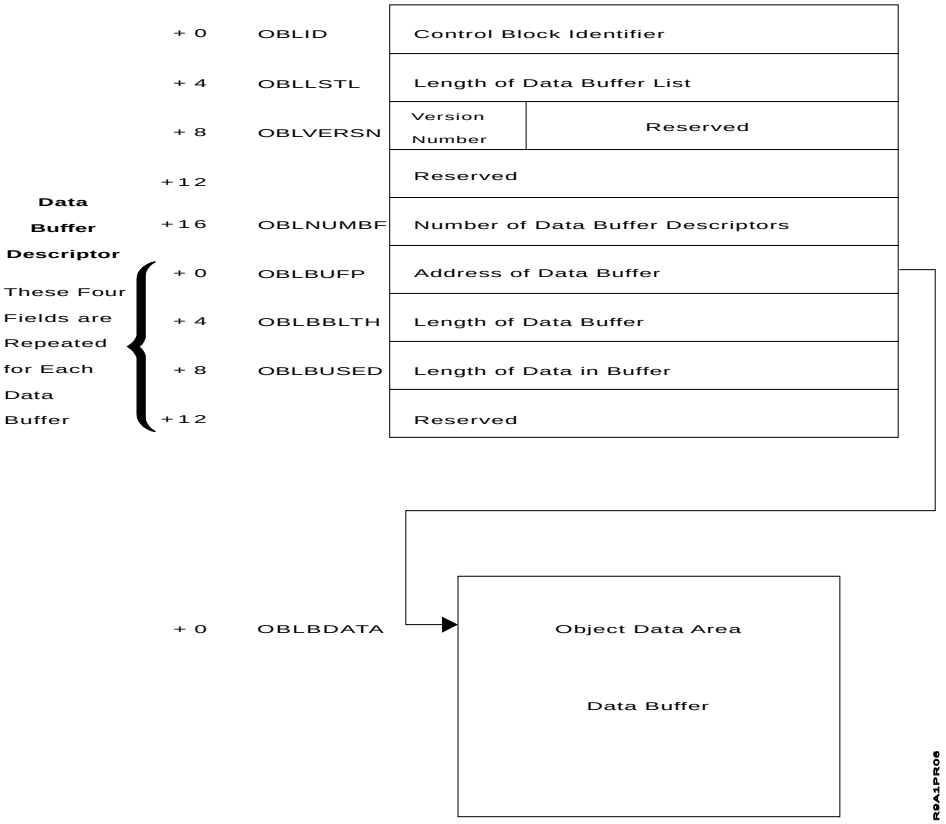


Figure 5. Data Buffer List Structure Diagram

The caller uses the buffer descriptor for each buffer to provide buffer location, buffer size, and data length to the system; it is then used by the system to return data length information to the caller. The OBLBBLTH field indicates the buffer length. The contents of this field must be set by the caller. The OBLBUSED field will indicate the number of bytes used in the buffer. For a STORE request, the value in this field is supplied by the caller; for a RETRIEVE request, this field is zeroed by OAM and updated when information is loaded in the data area.

Part of an object may occupy space in an individual buffer; therefore, an object may span several buffers. For a RETRIEVE request, the entire object (or requested portion) is stored in the buffer space provided. If an error occurs during a RETRIEVE request, the buffer data is invalid. Given adequate buffer space, RETRIEVE will fill the first buffer with data, then the second, and so forth until the total number of bytes filled in the buffers is equal to the size of the object (or the requested portion of the object). For a STORE request, if the object data is in a contiguous area of storage immediately following the last (or only) buffer descriptor, the object data is stored directly from the data buffers; otherwise, object data is reblocked from the data buffers into a temporary storage buffer and stored from the temporary buffer.

CBRIQEL Macro

The CBRIQEL macro describes the area to which the QEL keyword on the OSREQ macro points. The area contains a header and a list of buffer descriptors. Each buffer descriptor points to and describes one query buffer. A query buffer contains query elements. A query element describes the information retrieved by the OSREQ QUERY function for an object. Each query buffer must be large enough to contain at least one query element.

A series of query buffers can be specified in the buffer list so that information about a large number of objects can be returned without requiring a large contiguous area in virtual storage.

The CBRIQEL macro is a mapping macro that consists of four DSECTs. The QEL DSECT describes the entire buffer list. The QELBDESC DSECT is used in conjunction with the QEL DSECT to map one query buffer descriptor in the buffer list.

The QELB DSECT describes a query buffer. The QELQ DSECT is used in conjunction with the QELB DSECT to map one query element in the query buffer. Figure 6 on page 36 and Figure 7 on page 38 describe the contents of the DSECTs.

The OSREQ QUERY command returns three order retrieval keys. The primary retrieval order key field (QELQPROK), the backup retrieval order key field (QELQBROK), and the secondary backup retrieval order key field (QELQB2OK) are 10-byte fields that allow OAM to retrieve a large number of objects within a limited amount of time. It is important that OAM retrieve these objects in an order that minimizes the mounting of the media. This utilizes process time efficiently when the objects reside on removable media.

The OSREQ QUERY command returns, in addition to the primary retrieval order key and the backup retrieval order key, a second backup retrieval order key. To retrieve objects the most efficiently, you may use the QELQB2OK field on the CBRIQEL mapping macro, which sorts objects prior to their retrieval. This retrieval method uses less time to position and mount media and is therefore more efficient. If no second backup copy of the object exists, this field contains binary zeroes.

These order retrieval keys are important when you use the output that is created by the OSREQ QUERY request to retrieve a large number of objects. Use the primary retrieval order key, the backup retrieval order key, or the secondary backup retrieval order key for each object to sort the list of objects that is indicated on the OSREQ QUERY request output for retrieval. Using these keys minimizes the number of mount requests for each piece of removable media that contains the objects that are being retrieved.

```
     QEL       DSECT             Query buffer list control block
               DS   0F
 +0  QELID     DS   CL4           Control block identifier ('QEL ')
 +4  QELLSTL   DS   F            Length of query buffer list in bytes
                                 including buffer descriptors
 +8  QELVERSN  DS   XL1           Query buffer list version
 +9  QELRSVD1  DS   XL3           Reserved, must be zero
 +12 QELRSVD2  DS   F             Reserved, must be zero
 +16 QELNUMBF  DS   F             Number of query buffer descriptors
 +20 QELBUFL   DS   0F            Beginning of query buffer descriptor
                                 list, mapped by QELBDESC
```

The following query buffer descriptor is repeated for each query buffer:

```
     QELBDESC  DSECT             Query buffer descriptor
 +0  QELBUFP   DS   A             Address of query buffer
 +4  QELBBLTH  DS   F             Length of query buffer
 +8  QELBUSED  DS   F             Number of bytes returned in query
                                 buffer
 +12 QELBRSV1  DS   F             Reserved, must be zero
```

Each query buffer is described as follows:

```
     QELB      DSECT             Query buffer
               DS   0F
 +0  QELBDATA  DS   0X            Object data area
```

Each query element is described by the following:

```
     QELQ      DSECT             Query element
+0   QELQELE   DS   H             QE length including this field
+2   QELQECD   DS   CL10          Creation date (yyyy-mm-dd)
+12  QELQEDH   DS   CL1           Set to '-'
+13  QELQECT   DS   CL15          Creation time (hh.mm.ss.nnnnnn)
+28  QELQELD   DS   CL10          Last referenced date (yyyy-mm-dd)
+38  QELQEED   DS   CL10          Expiration date (yyyy-mm-dd)
+48  QELQESC   DS   XL2,CL8       Storage class length and name
+48  QELQESCL  EQU  QELQESC,2     Storage class length
+50  QELQESCN  EQU  QELEQSCL+2,8  Storage class name
+58            DS   CL22          Reserved
+80  QELQEMC   DS   XL2,CL8       Management class length and name
+80  QELQEMCL  EQU  QELQEMC,2     Management class length
+82  QELQEMCN  EQU  QELQEMCL+2,8  Management class name
+90            DS   CL22          Reserved
+112 QELQEOS   DS   F             Object size
+116 QELQECN   DS   XL2,CL44      Collection name length and name
+116 QELQECNL  EQU  QELQECN,2     Collection name length
+118 QELQECNN  EQU  QELQECNL+2,44 Collection name
+162 QELQEON   DS   XL2,CL44      Object name length and name
+162 QELQEONL  EQU  QELQEON,2     Object name length
+164 QELQEONN  EQU  QELQEON+2,44  Object name
+208 QELQERRT  DS   F             Estimated retrieval response time
                                 (milliseconds). Value of -1 means
                                 this information is not available.
+212 QELQPROK  DS   CL10          Primary retrieval order key
+222 QELQBROK  DS   CL10          Backup retrieval order key
+232 QELQB2OK  DS   CL10          Secondary backup retrieval order key
```

Figure 6. Fields Described by CBRIQEL

The QELVERSN and QELQELE fields must be set by the user, as indicated below. The QELQELE field should be adjusted to reflect the inclusion or exclusion of the QELQPROK, QELQBROK, and QELQB2OK fields in the total length of the QUERY element.

- If QELVERSN>=5, then the query buffer (QELQ) contains the QELQPROK, QELQBROK, and QELQB2OK fields. These backup retrieval order key fields contain binary zeroes if none of the backup copies exists.
- If QELVERSN=4, then the query buffer (QELQ) contains the QELQPROK and QELQBROK fields. The backup retrieval order key fields contain binary zeroes if none of the backup copies exists.
- If QELVERSN<4, then none of the fields (QELQPROK, QELQBROK, and QELQB2OK) is included in the query buffer (QELQ).

The estimated retrieval response time field (QELQERRT) does not take current system workload into consideration. The following values are returned to indicate object location, thereby determining an estimated retrieval response time.

−1	Object location cannot be determined currently.
300	Object resides on DASD.
12 000	Object resides in an optical library.
60 000	Object resides on a tape volume inside an automated tape library.
120 000	Object resides on an optical volume on the shelf.
240 000	Object resides on a tape volume outside an automated tape library.

The estimated minimum retrieval response time field (QELQERRT) contains the estimated time (in milliseconds) needed to retrieve the object. It is the total estimated time, from the initiation of the RETRIEVE request until control is returned to the caller with the object. This time is based on the physical device characteristics of the hierarchy level on which the object is stored. It is an optimum time and does not consider delays due to queue lengths, system load, or any other dynamic situation. The time returned is a representative time to retrieve an object from the device on which the object resides. The estimated time does not consider the size or location of the specific object. If the retrieval response time cannot be determined, QELQERRT is set to the reserved value of −1 (X'FFFFFFFF').

Figure 7 on page 38 is a structure diagram of the query buffer list (CBRIQEL) pointed to by the QEL keyword on an OSREQ QUERY macro:

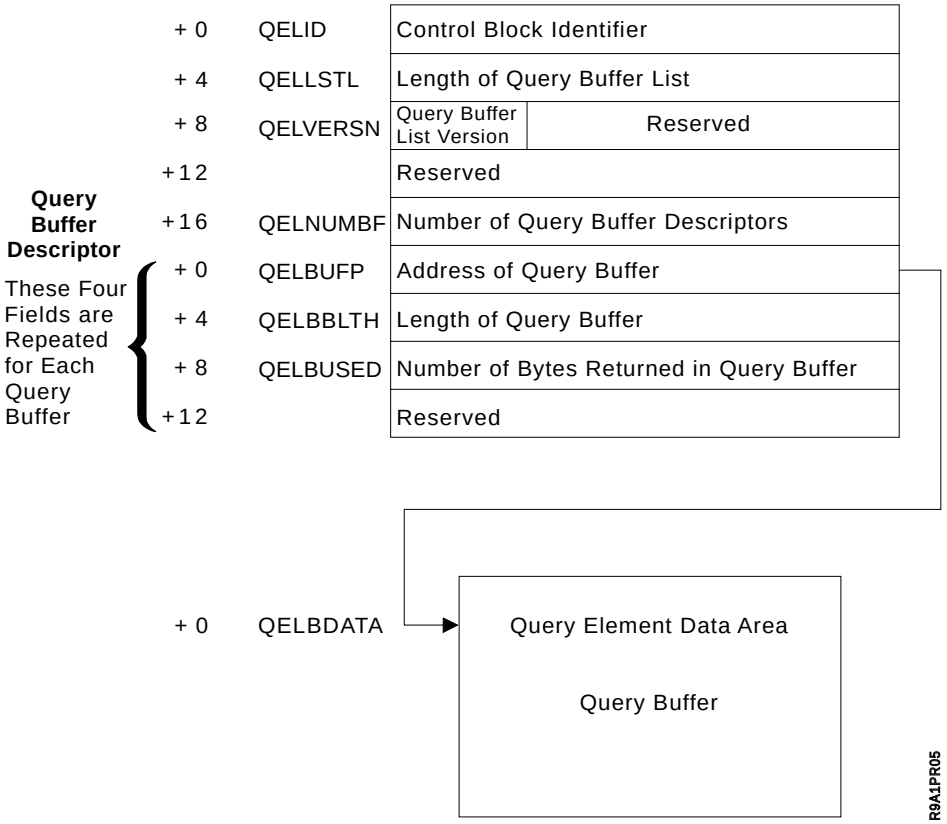


Figure 7. Query Buffer List Structure Diagram

The caller uses the buffer descriptor for each buffer to provide buffer location, buffer size, and data length to the system; it is then used by the system to return data length information to the caller. The QELBBLTH field indicates the length of the query buffer. The content of this field must be set by the caller (the query buffer must be at least long enough to hold one query element). The QELBUSED field indicates the number of bytes used in the query buffer. This field is zeroed by OAM and updated when information is stored in the query buffer.

Information about multiple objects (that is, multiple query elements) may occupy space in one query buffer; however, no query element (QE) spans query buffers. The first query buffer is filled until additional complete query elements no longer fit, then the second buffer is filled, and so forth. The QELBUSED field indicates the number of bytes used in each query buffer. Unused query buffers have the QELBUSED field set to zero. The first zero QELBUSED field indicates the end of a list of query elements. When the buffer space provided (QEL) is inadequate for the number of query elements retrieved, a warning return code is provided to the caller, and the number of query elements that fit in the available space is placed in the query buffers.

The QE length field contains the size of the individual query element. The date fields are in ISO format: yyyy-mm-dd. This format is different from the format of the four-byte date stored in the object directory, which is a compressed form of this information. An expiration date of “0001-01-01” indicates that no expiration date has been specified, and therefore the management class is used to determine the expiration date. If the object has not been retreived or changed, or if the UPD=N parameter was specified on the CBRINIT statement of the IEFSSN*xx* member of PARMLIB that was used during IPL, the last date referenced is “0001-01-01”. A last

date referenced of “0001-01-01” indicates that the last referenced date and pending action date are not to be updated when an object is retrieved.

The object name field contains the length of the name and the object name. When the object name is less than 44 characters, it is left-justified in the field adjacent to the length, which is the first byte of the field. The unused characters in this field are blanks.

Appendix A. Sample Program for Object Storage

This appendix contains the source listing of a sample program that uses the OSREQ macro for object manipulation. See Figure 8 on page 42 for the sample program. This program is available as member CBROSREQ in SAMPLIB.

You can use member CBROSREQ in a number of ways depending on your application:

- You can generate the IADDRESS parameter in the OSREQ ACCESS function by specifying IADD as the SYSPARM value in the PARM field of the EXEC JCL statement. For example:

  ```
  //ASSEMBLE EXEC PGM=IEV90,PARM='LOAD,DESK,SYSPARM(IADD)'
  ```

- You can link-edit member CBROSREQ as part of the application load module. You do not need to issue LOAD request before using the OSREQ calls.
- You can use member CBROSREQ without modification to support application programs written in PL/1 or COBOL.
- You can modify member CBROSREQ as necessary to support applications written in high-level languages other than PL/1 or COBOL.
- You must run the DB2 pre-compiler due to the EXEC SQL statements in the code.

```
*********************************************************************
*
* DESCRIPTIVE NAME: Object Storage Request Sample interface
*
* FUNCTION: Provides a generalized interface for the Object Storage
*           Request (OSREQ) macro.
*
* OPERATION: This routine is called with a parameter area that
*            defines the function and pointers necessary to invoke
*            the OSREQ macro and/or synchronize the databases that
*            are connected to the current DB2 thread.
*            If it is determined that an OSREQ function is requested,
*            then the OSREQ parameter list is filled in with an
*            MF=M form of the macro. The function is executed via an
*            MF=E form.
*            A call is made to an internal routine which will
*            determine the need to synchronize the databases.
*            If sync has been requested and the value in the
*            field pointed to by the RETURN_CODE_PTR
*            field is 0 or 4, then DB2 will be notified
*            to commit all changes made to the databases
*            since the last synchronization point.
*            If sync has been requested and the value in the
*            field pointed to by the RETURN_CODE_PTR
*            field is greater than 4, DB2 will be
*            notified to rollback all changes made to the data
*            bases since the last synchronization point.
*
* NOTE: To generate the IADDRESS keyword in the OSREQ ACCESS function,
*       specify the SYSPARM value as IADD in the PARM field of
*       the EXEC JCL statement. For example:
*
*       //ASSEMBLE  EXEC PGM=IEV90,PARM='LOAD,DECK,SYSPARM(IADD)'
* INPUT: Register 1 must point to a 4-byte field that contains
*                   an address of an area that is described by
*                   the dsect named DATAAREA in this program.
*                   The DATAAREA must be filled in to indicate
*                   the function requested and provide the proper
*                   data for execution of the OSREQ macro.
*        Register 13 must point to a 72-byte area into which this
*                   routine will save the registers at entry and
*                   from which registers will be restored at exit.
*        Register 14 must point to the instruction address to which
*                   this routine will return.
*        Register 15 must point to the entry point address of this
*                   routine.
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 1 of 10)

```
* OUTPUT: Register 15 will contain the return code received from
*                     the syncpoint processing.
*         Fields pointed to by REASON_CODE_PTR and RETURN_CODE_PTR
*                will contain the reason and return codes returned
*                from OAM.
*         Areas defined by the CBRIBUFL (for retrieve) and CBRIQEL
*               (for query) will be filled in when the respective
*               function is requested.
*
*********************************************************************
OSRSAMPL CSECT ,
OSRSAMPL AMODE 31
OSRSAMPL RMODE ANY
         STM   R14,R12,12(R13)
*
* Register 12 is the base for the code
*
         LR    R12,R15
         USING OSRSAMPL,R12
*
* Register 11 is the base for the data area which is passed to this
* routine as a parameter.
*
         L     R11,0(R1)
         USING DATAAREA,R11
         LA    R15,SAVE_AREA
         ST    R15,8(R13)
         ST    R13,SAVE_AREA+4
         LR    R13,R15
*
* The static OSREQ parameter list is copied into the work area
*
         MVC   PARM_LIST,STATIC_PARM_LIST
*
* The parameter list is now modified to establish all of the basic
* parameters of all of the OSREQ functions.
* A pointer with a value of zero is equivalent to an omitted parameter.
*
         L     R0,MESSAGE_AREA_PTR
         L     R2,OBJECT_SIZE_PTR
         L     R3,STORAGE_CLASS_PTR
         L     R4,MANAGEMENT_CLASS_PTR
         L     R5,RETENTION_PERIOD_PTR
         L     R6,RETRIEVE_OFFSET_PTR
         L     R7,RETRIEVE_LENGTH_PTR
         L     R8,RETURN_CODE_PTR
         L     R9,REASON_CODE_PTR
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 2 of 10)

```
     OSREQ (STORE),MF=(M,PARM_LIST),
               MSGAREA=(R0),          DB2 error messages returned here
               TOKEN=TOKEN_AREA,      Contains logical OAM connection
               COLLECTN=COLLECTION_NAME,
               NAME=OBJECT_NAME,
               SIZE=(R2),
               STORCLAS=(R3),
               MGMTCLAS=(R4),
               RETPD=(R5),
               OFFSET=(R6),           Starting byte for retrieve
               LENGTH=(R7),           Length of retrieve
               RETCODE=(R8),          Register 15 is stored here
               REACODE=(R9)           Register 0 is stored here
               CLC RELEASE_BUFFER,=CL3'YES'
               BNE NORELBUF
     OSREQ (STORE),MF=(M,PARM_LIST),
               RELBUF=YES             Will release pages after STORE
NORELBUF       DS   0H
               CLC FUNCTION_REQUEST,=CL8'ACCESS'
               BNE TRY_STORE
*
* The logical connection to OAM is made here.
* If this is MVS batch, the call attach facility will be used
* to connect to DB2, and a thread will be OPENed to Plan (CBRIDBS).
* If this program runs in an environment where the connection
* and the thread to DB2 must be done by the external environment
* rather than OSREQ ACCESS, then the IADDRESS keyword will
* allow OSREQ to use the existing DB2 SQL interface rather than
* set up the call attach facility linkage to DB2.  Use the
* IADDRESS parameter ONLY when this program MUST use the DB2
* interface established outside of OSREQ ACCESS.  The primary
* users of IADDRESS are IMS/VS transaction programs.
* In all cases system control blocks will be created and/or modified
* to provide this access to OAM.
*
* To generate the keyword IADDRESS in OSREQ ACCESS function, a SYSPARM
* have a value IADD is specified in PARM field of the EXEC JCL
* statement
*
      AIF ('&SYSPARM' EQ 'IADD').IA2
      OSREQ ACCESS,MF=(E,PARM_LIST)
          AGO   .SKIP1
.IA2     ANOP
* In this sample we use DSNHLI as SQL interface module to DB2
         L     R2,=V(DSNHLI)
     OSREQ ACCESS,MF=(E,PARM_LIST),
               IADDRESS=(R2)          Get the address of the interface
.SKIP1   ANOP
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 3 of 10)

```
*
* In the MVS batch environment, syncpoint processing may be desirable
* after ACCESS because the DB2 plan name can be changed at this time.
*
               B     TRY_SYNC_POINT
TRY_STORE      DS    0H
               CLC FUNCTION_REQUEST,=CL8'STORE'
               BNE TRY_CHANGE
*
* This will store an object in the DB2 object tables or on
* an optical disk, depending on the storage class specified.
*
         L     R10,STORE_BUFFER_PTR
     OSREQ STORE,MF=(E,PARM_LIST),
               BUFLIST=(R10)
               B     TRY_SYNC_POINT
TRY_CHANGE     DS    0H
               CLC FUNCTION_REQUEST,=CL8'CHANGE'
               BNE TRY_QUERY
*
* This invocation of the OSREQ macro will change information in the
* directory that has been specified. A zero pointer in DATAAREA
* will result in no change for the respective information. All
* pointers zero result in no change.
*
      OSREQ CHANGE,MF=(E,PARM_LIST)
               B     TRY_SYNC_POINT
TRY_QUERY      DS    0H
               CLC FUNCTION_REQUEST,=CL8'QUERY'
               BNE TRY_RETRIEVE
*
* Query the database for the directory information that was stored.
* The size of the object can be extracted from this information so
* that a GETMAIN can be done for the area necessary for the
* retrieve operation.
*
         L     R10,QUERY_BUFFER_PTR
      OSREQ QUERY,MF=(E,PARM_LIST),
               QEL=(R10)
              B     TRY_SYNC_POINT
TRY_RETRIEVE   DS    0H
               CLC FUNCTION_REQUEST,=CL8'RETRIEVE'
               BNE TRY_DELETE
*
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 4 of 10)

```
* A partial retrieve can be done to obtain the first xxx bytes of
* the object. In some cases the application may have some control
* information in this area to allow retrieval of still another part
* of the object, (which could be an image).
*
         L     R10,RETRIEVE_BUFFER_PTR
     OSREQ RETRIEVE,MF=(E,PARM_LIST),
               BUFLIST=(R10)
               B     TRY_SYNC_POINT
TRY_DELETE     DS    0H
               CLC FUNCTION_REQUEST,=CL8'DELETE'
               BNE TRY_UNACCESS
*
* This invocation will delete the object named from the object table
* and the directory.
*
      OSREQ DELETE,MF=(E,PARM_LIST)
               B     TRY_SYNC_POINT
TRY_UNACCESS   DS    0H
               CLC FUNCTION_REQUEST,=CL8'UNACCESS'
               BNE TRY_SYNC_POINT
*
* The logical connection to OAM should be broken before the TASK
* terminates so that OAM can remove any system control blocks
* that it built during ACCESS
*
      OSREQ UNACCESS,MF=(E,PARM_LIST)
*
TRY_SYNC_POINT DS    0H
*
* Save register 15 in the return code area and register 0 in the
* reason code area after return from OSREQ.  This is recommended
* because, under certain error conditions, the return code and reason
* code areas may not be set by OSREQ.
*
             ST    R15,0(,R8)        Save Return Code
             ST    R0,0(,R9)         Save Reason Code
*
* Each function should be "committed" or "rolled back" depending
* on the return and reason codes from OAM.
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 5 of 10)

```
* This routine should issue:
*     SYNCPOINT with optional ROLLBACK in the CICS environment
*  or SYNC or ROLL,ROLLB in the IMS environment
*  or COMMIT or ROLLBACK in the TSO environment
*  or CALL DSNALI to CLOSE and OPEN the thread to DB2 in the
*                    MVS batch environment (which is shown here).
*
               SR    R15,R15              Ensure return code 0 if
*                                         no syncpoint processing.
               CLC SYNC_POINT,=CL3'YES'
               BNE   EXIT
*
* A parameter list is constructed for the call to DSNALI
* to close the thread to commit or rollback changes.
*
               LA    R10,=CL12'CLOSE'
               ST    R10,WORK_AREA1       Set function to close.
               LA    R10,=CL8'SYNC'       Prime for sync.
            AIF ('&SYSPARM' EQ 'IADD').IA1
               L     R15,RETURN_CODE_PTR  Check OAM return code
               LA    R9,4                 to see if rollback should
               C     R9,0(R15)            be issued instead of sync.
               BNL   SET_SYNC
               LA    R10,=CL4'ABRT'
SET_SYNC       ST    R10,WORK_AREA2       Set the action parameter.
               OI    WORK_AREA2,X'80'     Set end of parameter list
               BAL   R10,LOAD_DSNALI      This points R15 to DSNALI.
               LA    R1,WORK_AREA1        Point to parameter list.
               CALL  (15)                 Call DSNALI
               LTR   R15,R15              Check for good return
               BNZ   EXIT                 This routine has no
*                                         recovery for bad returns
*                                         from CLOSE. The caller
*                                         should UNACCESS then ACCESS.
*
               AGO   .SKIP
.IA1  ANOP
               LA    R8,SQLSTUFF
               USING SQLDSECT,R8
               L     R15,RETURN_CODE_PTR
               LA    R9,4
               C     R9,0(R15)
               BNL   SET_SYNC
               EXEC  SQL ROLLBACK
               B     EXIT
SET_SYNC       EXEC  SQL COMMIT
               AGO   .SKIP2
.SKIP   ANOP
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 6 of 10)

```
*
* A parameter list is constructed for the call to DSNALI
* to open the thread to DB2. A new plan name could be specified
* or the same name (CBRIDBS) could be specified.
*
               LA    R10,=CL12'OPEN'
               ST    R10,WORK_AREA1        Set function to open.
               LA    R10,DB2_SUBSYS_ID
               ST    R10,WORK_AREA2        Set the ssid parameter.
               LA    R10,PLAN_NAME
               ST    R10,WORK_AREA3        Set the thread parameter.
               OI    WORK_AREA3,X'80'      Set end of parameter list
               BAL   R10,LOAD_DSNALI       This points R15 to DSNALI.
               LA    R1,WORK_AREA1         Point to parameter list.
               CALL  (15)                  Call DSNALI
.SKIP2    ANOP
EXIT           DS    0H
*
* Restore all registers except regs 15 and 0, then return to caller
*
               L     R13,SAVE_AREA+4
               L     R14,12(R13)
               LM    R1,R12,24(R13)
               BR    R14
*
* This subroutine will determine if DSNALI is loaded.
* If it is, register 15 will be return with the address of DSNALI.
* If it is not, the module will be loaded and the address returned
* in register 15.
* If DSNALI cannot be loaded an 806 abend will occur, so be sure
* that there is a JOBLIB or STEPLIB pointing to the library that
* contains the load module DSNALI.
*
LOAD_DSNALI    DS    0H
               L     R15,WORK_AREA4    DSNALI address is saved here.
               LTR   R15,R15
               BNZR  R10               Return with address of DSNALI
               LOAD  EP=DSNALI
               ST    R0,WORK_AREA4     Save for future calls.
               LR    R15,R0            Return address of DSNALI
               BR    R10                to caller
*
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 7 of 10)

```
* Register definitions
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
* All literals will be included at this point.
*
        LTORG
*
* This static parameter list will be used as a template for
* OSREQ invocations in the executable code.
*
STATIC_PARM_LIST OSREQ (STORE),MF=(L)
STATIC_LIST_END EQU *
*
* This area is provided by the caller of this routine
*
DATAAREA DSECT
*********************************************************************
*
*  This area must be obtained by the caller of OSRSAMPL and presented
* as a parameter to OSRSAMPL. It is expected that all subsequent calls
* will point to this same area. There is information in the area
* that will be used across calls.
*
*********************************************************************
SAVE_AREA            DS 18F    Save area for this module.
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 8 of 10)

```
*******
* The following two named fields are set by the caller of OSRSAMPL.
* If the value in the field is not a valid value, the respective
* activity will not be executed.
*******
FUNCTION_REQUEST      DS CL8    OSREQ function request value
*                               ACCESS, STORE, etc. or other
SYNC_POINT            DS CL3    Syncpoint request, YES or other
                      DS CL1    Reserved
*******
* The following five fields are set by OSRSAMPL and should not be
* altered by the caller. Subsequent calls to OSRSAMPL will rely
* on the information stored here.
*******
WORK_AREA1            DS A      Used
WORK_AREA2            DS A          for
WORK_AREA3            DS A             parameters.
WORK_AREA4            DS A      Holds address of DSNALI
TOKEN_AREA            DS 2F     OSREQ token, do not change it.
*******
* The following fields are set by the caller of OSRSAMPL
* The pointers are not altered by OSRSAMPL but the data that
* the pointers reference may be.
*******
RETURN_CODE_PTR       DS A      Pointer to OSREQ return code
*                               The return code is referenced by
*                               the syncpoint processing.
REASON_CODE_PTR       DS A      Pointer to OSREQ reason code
MESSAGE_AREA_PTR      DS A      Pointer to message area
RETENTION_PERIOD_PTR  DS A      Pointer to retention period
OBJECT_SIZE_PTR       DS A      Pointer to object size value
MANAGEMENT_CLASS_PTR  DS A      Pointer to management class parameter
STORAGE_CLASS_PTR     DS A      Pointer to storage class parameter
RETRIEVE_OFFSET_PTR   DS A      Pointer to offset value
RETRIEVE_LENGTH_PTR   DS A      Pointer to retrieve length value
RETRIEVE_BUFFER_PTR   DS A      Pointer to retrieve buffer list
STORE_BUFFER_PTR      DS A      Pointer to store buffer list
QUERY_BUFFER_PTR      DS A      Pointer to query buffer list
RELEASE_BUFFER        DS CL3    RELBUF value, YES or other
                      DS CL1    Reserved
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 9 of 10)

```
*
* Plan name and DB2 subsystem identification MUST be provided
* for MVS batch sync point processing.
*
PLAN_NAME             DS CL8    DB2 plan name for OPEN thread
DB2_SUBSYS_ID         DS CL4    Installation subsystem name for DB2.
*
* Collection name and object name MUST be provided with every
* request for STORE, RETRIEVE, QUERY, CHANGE, and DELETE.
*
COLLECTION_NAME       DS H      Length of collection name
                      DS CL44   Collection name character string
OBJECT_NAME           DS H      Length of object name
                      DS CL44   Object name character string
********
* The following area is completely overlaid each time OSRSAMPL
* is called
********
PARM_LIST DS CL(STATIC_LIST_END-STATIC_PARM_LIST)  Dynamic parm list
          DS CL2528    DO NOT USE THIS AREA, BELONG TO CALLER
          EXEC  SQL INCLUDE SQLCA
SQLSTUFF  DS CL(SQLDLEN)
DATA_AREA_END   EQU *
OSRSAMPL CSECT
*
         END   OSRSAMPL
```

Figure 8. Sample Program for an Object Storage Request Using the OSREQ Macro (Part 10 of 10)

Appendix B. Reason Codes

Table 1 contains only general-use return and reason codes. All other return and reason codes are for diagnostic use only and are reserved for IBM internal use. Refer to *z/OS DFSMSdfp Diagnosis Reference* for information about diagnostic return and reason codes. For more detailed information concerning the keywords referenced in this section, refer to “OSREQ Keyword Parameter Descriptions” on page 23.

Table 1. Return/Reason Codes

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
0	0	0	0	0	The request has successfully completed.	No action is required.
4	t	x	y	z	The request has completed with a warning condition: **t** UNIQUE OSREQ REASON CODE **x** INTERNAL FUNCTION CODE **y** ERROR INDICATION **z** RESERVED	Correct program, if necessary.
4	4	x	1	z	The QEL buffer segments are too short to accommodate all of the available entries. As many entries as can fit in the segments are returned.	Execute the QUERY with a larger QEL buffer.
4	4	x	2	z	An unavailable resource condition was detected during a generic group query which excludes one or more databases from the results. The QEL may contain entries from the available databases.	Activate the databases, if necessary.
4	4	x	3	z	An UNACCESS has completed. The token has been cleared. There are one or more requests outstanding. The outstanding requests are not terminated.	Correct the program, if necessary.

Table 1. Return/Reason Codes (continued)

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
4	4	x	4	z	A STORE or CHANGE request has completed. A collection name was created, but one or more of the following conditions has occurred as indicated by bits set in byte 3 (z): Z=BIT MAP: BIT 0 CATALOG ENTRY created for the collection BIT 1 RESERVED BIT 2 STORAGE CLASS specified for the collection overridden BIT 3 MANAGEMENT CLASS specified for the collection overridden BIT 4 RETENTION PERIOD specified for the object overridden BIT 5 RESERVED BIT 6 STORAGE CLASS Specified for the object overridden BIT 7 MANAGEMENT CLASS Specified for the object overridden	Issue a QUERY to see new parameters, if desired. BIT MAP OF BYTE 3: 1XXX XXXX XX1X XXXX XXX1 XXXX XXXX 1XXX XXXX XX1X XXXX XXX1
4	4	x	5	z	DB2 SQL return code conversion, Module DSNTIAR, was not found in the LINKLIST.	Ensure that module DSNTIAR is available in the LINKLIST.
4	4	x	6	z	First backup copy retrieved; primary copy of the object was not available with Access Backup active.	
4	4	x	7	z	Second backup copy retrieved; primary copy of the object was not available with Access Backup active.	
8	t	x	y	z	Request unsuccessful. **t** UNIQUE OSREQ REASON CODE **x** INTERNAL FUNCTION CODE **y** FIRST PARAMETER WITH AN ERROR **z** TYPE OF ERROR	Correct calling program.
8	24	x	y	z	The parameter is unusable, incorrect, invalid, or incomplete.	
8	24	x	1	z	PARAMETER LIST (MF=L)	
8	24	x	1	1	The parameter list is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the parameter list name or the parameter list name length.	
8	24	x	1	2	The parameter list is invalid or incomplete.	
8	24	x	2	z	SIZE	

Table 1. Return/Reason Codes (continued)

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
8	24	x	2	1	The size (fullword) passed to OAM on the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the size (fullword).	
8	24	x	2	2	The size passed to OAM on the OSREQ macro contains an invalid value.	
8	24	x	3	z	RETPD	
8	24	x	3	1	The RETPD area (fullword) passed to OAM on the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the RETPD (fullword).	
8	24	x	3	2	The RETPD passed to OAM on the OSREQ macro contains an invalid value.	
8	24	x	4	z	STORCLAS	
8	24	x	4	1	The STORCLAS area passed to OAM on the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the STORCLAS.	
8	24	x	4	2	The STORCLAS passed to OAM on the OSREQ macro contains an invalid character.	
8	24	x	4	3	The STORCLAS passed to OAM on the OSREQ macro contains an invalid length value.	
8	24	x	5	z	MGMTCLAS	
8	24	x	5	1	The MGMTCLAS area passed to OAM on the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the MGMTCLAS.	
8	24	x	5	2	The MGMTCLAS passed to OAM on the OSREQ macro contains an invalid character.	
8	24	x	5	3	The MGMTCLAS passed to OAM on the OSREQ macro contains an invalid length value.	
8	24	x	6	z	QEL	
8	24	x	6	1	The QEL Buffer List passed to OAM in the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the QEL Buffer List.	

Table 1. Return/Reason Codes (continued)

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
8	24	x	6	2	The QEL Buffer List passed to OAM in the OSREQ macro contains one of the following conditions: • Incorrect ID • Incorrect length field • Incorrect version field • The user turned the RESERVED BIT “on” in the Query Buffer List Control Block.	
8	24	x	6	4	The QEL Buffer passed to OAM in the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the QEL Buffer.	
8	24	x	7	z	REASON/RETURN CODE STORAGE	
8	24	x	7	1	The REASON code area passed to OAM from the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the REASON code.	Check REGISTER 0 for REASON code error conditions.
8	24	x	7	2	The RETURN code area passed to OAM from the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the RETURN code.	Check REGISTER 15 for RETURN code error conditions.
8	24	x	8	z	BUFLIST	
8	24	x	8	1	The BUFLIST passed to OAM from the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the BUFLIST.	
8	24	x	8	2	The BUFLIST passed to OAM in the OSREQ macro contains one of the following conditions: • Incorrect ID • Incorrect length field • Incorrect version field • The user turned the RESERVED BIT “on” in the Data Buffer List Control Block.	
8	24	x	8	4	The BUFFER passed to OAM from the OSREQ macro is in unusable storage.	
8	24	x	8	5	The amount of buffer data provided on the STORE request is less than the specified size of the object.	
8	24	x	8	6	The amount of buffer data provided on the STORE request is greater than the specified size of the object.	
8	24	x	8	8	The amount of buffer data space provided on the RETRIEVE request is insufficient for the object.	
8	24	x	9	z	TOKEN	

Table 1. Return/Reason Codes (continued)

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
8	24	x	9	1	The TOKEN area passed to OAM from the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the TOKEN.	
8	24	x	9	2	The TOKEN set by the ACCESS macro is not being specified correctly on subsequent OSREQ requests.	
8	24	x	A	z	OBJECT NAME	
8	24	x	A	1	The OBJECT NAME passed to OAM on the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the OBJECT NAME or the OBJECT NAME length.	
8	24	x	A	2	The OBJECT NAME passed to OAM on the OSREQ macro is not fully qualified. The OBJECT NAME contains an asterisk(*) as the last character in the name.	
8	24	x	A	3	The OBJECT NAME passed to OAM on the OSREQ macro contains a qualifier longer than 8 characters.	
8	24	x	A	4	The OBJECT NAME passed to OAM on the OSREQ macro contains an invalid character. One of the characters in the OBJECT NAME is not an uppercase alphabetic (A-Z), numeric (0–9), or national (@, #, $) character.	
8	24	x	A	5	The OBJECT NAME passed to OAM on the OSREQ macro contains a null qualifier, meaning ONE of the following is true: • The first character of the OBJECT NAME is a period. • The last character of the OBJECT NAME is a period. • The OBJECT NAME contains two consecutive periods.	
8	24	x	A	6	The OBJECT NAME passed to OAM on the OSREQ macro contains more than one asterisk.	
8	24	x	A	7	The OBJECT NAME passed to OAM on the OSREQ macro contains an invalid qualifier. One of the qualifiers does not start with an uppercase alphabetic character (A-Z) or national character ($, #, @).	
8	24	x	A	8	The OBJECT NAME passed to OAM on the OSREQ macro contains an imbedded blank.	
8	24	x	A	9	The OBJECT NAME passed to OAM on the OSREQ macro has an invalid length. The length is zero, negative, or longer than 44 characters.	
8	24	x	B	z	The OSREQ function.	
8	24	x	B	2	The function specified is unknown.	
8	24	x	C	z	OFFSET	

Table 1. Return/Reason Codes (continued)

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
8	24	x	C	1	The OFFSET passed to OAM from the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the OFFSET.	
8	24	x	C	2	The OFFSET value is larger than the length of the object.	
8	24	x	C	3	The OFFSET value is negative.	
8	24	x	D	z	LENGTH	
8	24	x	D	1	The LENGTH passed to OAM from the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the LENGTH.	
8	24	x	D	2	The LENGTH value requested, plus the value specified on the OFFSET keyword, is larger that the SIZE of the object.	
8	24	x	D	3	The LENGTH value is negative.	
8	24	x	E	z	MSGAREA	
8	24	x	E	1	The MSGAREA passed to OAM from the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the MSGAREA.	
8	24	x	E	2	The MSGAREA length value is negative.	
8	24	x	F	z	COLLECTION NAME	
8	24	x	F	1	The COLLECTION NAME passed to OAM on the OSREQ macro is in unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the COLLECTION NAME or the COLLECTION NAME length.	
8	24	x	F	2	The COLLECTION NAME passed to OAM on the OSREQ MACRO is not fully qualified. The COLLECTION NAME contains an asterisk (*) as the last character in the name.	
8	24	x	F	3	The COLLECTION NAME passed to OAM on the OSREQ macro contains a qualifier longer than 8 characters.	
8	24	x	F	4	The COLLECTION NAME passed to OAM on the OSREQ macro contains an invalid character. One of the characters in the COLLECTION NAME is not an uppercase alphabetic (A-Z), numeric (0–9), or national (@, #, $) character.	

Table 1. Return/Reason Codes (continued)

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
8	24	x	F	5	The COLLECTION NAME passed to OAM on the OSREQ macro contains a null qualifier, meaning ONE of the following is true. • The first character of the COLLECTION NAME is a period. • The last character of the COLLECTION NAME is a period. • The COLLECTION NAME contains two consecutive periods.	
8	24	x	F	6	Reserved	
8	24	x	F	7	The COLLECTION NAME passed to OAM on the OSREQ macro contains an invalid qualifier. One of the qualifiers does not start with an uppercase alphabetic character (A-Z) or national character ($, #, @).	
8	24	x	F	8	The COLLECTION NAME passed to OAM on the OSREQ macro contains an imbedded blank.	
8	24	x	F	9	The COLLECTION NAME passed to OAM on the OSREQ macro has an invalid length. The length is zero, negative, or longer than 44 characters.	
8	24	x	10	z	IADDRESS ERROR	
8	24	x	10	10	The IADDRESS passed to OAM from the OSREQ macro points to unusable storage. This means that OAM encountered a virtual storage translation exception (for example, an OC4 ABEND) when it attempted to reference the area of storage containing the IADDRESS.	
8	24	x	11	z	TTOKEN	
8	24	x	11	1	The TTOKEN passed to OAM is in unusable storage. This means that the tracking token is contained in the virtual storage area for which the application program does not have both fetch and store authorization. This is an indication of a programming logic error in the application program that is issuing the OSREQ macro invocation.	
8	28	x	y	z	An IADDRESS routine error was detected during execution of the DB2 language interface routine specified by IADDRESS **x, y, z** SYSTEM/USER COMPLETION CODE	
8	2C	x	y	z	No valid object was found. **z** RESERVED AND UNDEFINED	
8	2C	x	1	z	Segment not found in object storage 32k or 4k table.	
8	2C	x	2	z	The object segments entry was not found in the object directory.	
8	2C	x	3	z	An OSREQ retrieval request with VIEW=BACKUP was received, but a backup copy of the object does not exist.	
8	2C	x	4	z	An OSREQ retrieval request with VIEW=BACKUP2 was received, but a second backup copy of the object does not exist.	

Table 1. Return/Reason Codes (continued)

RETURN CODE	REASON CODE (BYTES) 0	1	2	3	ERROR DESCRIPTION	INSTALLATION ACTION
8	30	x	y	z	The object already exists. **z** RESERVED AND UNDEFINED	
8	30	x	1	z	The directory entry already exists.	
8	30	x	2	z	The object segment already exists.	
8	34	x	y	z	Request rejected for this task. **z** RESERVED AND UNDEFINED	
8	34	x	1	z	A request was issued from a task control block (TCB) other than the initial ACCESS request TCB.	
8	34	x	2	z	An ACCESS request is issued from the TCB while the prior ACCESS request is still active.	

Appendix C. Performance Considerations and Object Data Reblocking

This appendix documents diagnosis, modification or tuning information which is provided to help you write an efficient application program that uses the OSREQ macro.

Performance Considerations

Allowing page release by specifying RELBUF=YES on a STORE request minimizes unnecessary page-outs of buffer segment pages to auxiliary storage after they have been written to object storage.

Attention: RELBUF=YES may release pages that contain data that has not been committed to the database.

A generic QUERY request may use large amounts of instructions and virtual storage for the output, plus slow other accesses to the directory.

Database synchronization should follow the OSREQ invocation as soon as possible to minimize contention for resources.

When processing quantities of large objects, interactions among the following factors can degrade performance: virtual and real storage requirements, paging and auxiliary storage, data input/output, and movement (copying) of object data. All of these considerations can be affected by how the object data is structured by the application and what additional processing is required for OAM to complete the request. Applications can optimize the object data storage to minimize the impact of the above considerations, as described in the next section.

Object Data Reblocking

OAM attempts to process the data in the caller's buffers with a minimum of data movement. If the first or only buffer is large enough for all of the object data and the buffer immediately follows the buffer list, then the data is not moved within the caller's address space.

If the conditions described are not met, OAM obtains temporary storage to reblock the data. The virtual storage needed in addition to the calling program's requirements might be as great as the size of the largest object.

Object Storage

If the object data is not in one contiguous block in a storage area immediately following the end of the buffer list, the object data is reblocked into temporary storage within the caller's address space. The temporary storage requirements and uses are as follows:

- If the object is to be stored initially on DASD, temporary storage is obtained based on the total length of the object data:
  - If the total object data length is 3980 bytes or less, a temporary storage buffer of 4KB or less is obtained.
  - If the total object data length is greater than 3980 bytes, a temporary storage buffer of 32KB is obtained.

- If the object is to be stored initially on optical media, temporary storage that is large enough to contain the entire object is obtained.

In all cases where the object data requires reblocking, the object data segments are moved from the caller's buffers into the temporary storage buffer. The object data is reblocked into one contiguous block starting at the beginning of the temporary buffer.

For objects that are stored on DASD and are 3980 bytes or less in length, or for objects that are stored on optical media, only one block is created and stored.

For objects that are stored on DASD and are greater than 3980 bytes in length, the following steps are followed:
- Object data is moved into the temporary storage buffer until it is full.
- The object data in the temporary buffer is stored.
- The process of reblocking any remaining object data into the temporary buffer is repeated until all object data has been stored.

Object Retrieval

For objects that are retrieved from DASD (DB2), the object data is retrieved directly into the caller's buffer if the following conditions are met:
- The first or only buffer specified by the caller is contiguous to the buffer list.
- The first or only buffer is large enough to contain the entire object.
- The entire object is requested (not a part of the object).

For objects that are retrieved from optical storage, the object data is retrieved directly into the caller's buffer if the following conditions are met:
- The first or only buffer specified by the caller is contiguous to the buffer list.
- The first or only buffer is large enough to contain the entire object or the requested part of the object.

If any of the above conditions are not met, temporary storage is obtained for retrieving the object data. The virtual storage needed in addition to the calling program's requirements might be as great as the size of the largest object.

If the object data length is greater than the first buffer, the first buffer is completely filled, and the remainder of the object data is moved into the following buffers, filling each buffer until the last of the object data is moved into the last buffer.

Appendix D. Using the CBRUXSAE Installation Exit

The CBRUXSAE installation exit provides security authorization checking against users performing OSREQ transactions on object data. This exit is used at the application programming interface (OSREQ macro) level.

As originally provided, CBRUXSAE automatically returns a return code of zero indicating that the user ID is authorized to perform the OSREQ function. You must substitute this code with a validation routine to determine authority for a specific user ID if you wish authorization checking to be performed at the application interface level.

Register Contents on Entry to CBRUXSAE

The following are the register contents on entry to the CBRUXSAE installation exit:

Register	Contents
0	Contents on entry are unpredictable.
1	Contains the address of a parameter list, which contains four pointers: 1. Pointer to an 8-character field, which contains the OSREQ function being performed. Possible values are STORE, RETRIEVE, QUERY, CHANGE, DELETE. 2. Pointer to a 44-character field, which contains the object name associated with the requested function. 3. Pointer to a 44-character field, which contains the collection name associated with the requested function. 4. Pointer to an 8-character field, which contains the user ID associated with the requested function.
2–8	Contents on entry are unpredictable.
9	Contains the address of a 1024-byte storage area that can be used as automatic storage for the exit. The storage provided adheres to environment dependent restrictions. If more storage is needed, or there is a preference to obtain your own storage, environment dependent functions must adhere to GETMAIN restrictions. For example, a CICS environment must use CICS GETMAIN service to obtain storage instead of using MVS OBTAIN.
10–12	Contents on entry are unpredictable.
13	Contains the address of a 72 byte save area (standard linkage).
14	Contains a return point address to the caller (standard linkage).

If the return code from CBRUXSAE is not zero, return and reason codes are issued indicating that the user ID is not authorized to perform the particular OSR function. For more informtion concerning return and reason codes associated with this exit, refer to *z/OS DFSMSdfp Diagnosis Reference*.

Programming the CBRUXSAE Exit Correctly

CBRUXSAE is provided as a separate load module that needs to be link-edited into LINKLIB and invoked from OSR by the MVS LINK macro.

CBRUXSAE is invoked in the following state:

- Task mode (not SRB)
- Non-cross-memory mode (PASN=SASN=HASN)
- No MVS locks held
- Enabled for I/O and external interrupts
- Problem or supervisor state (the state of the invoker of the OSREQ macro interface)
- Key of the caller (invoker of the OSREQ macro interface)

CBRUXSAE must meet the following requirements:

- 31-bit addressing mode
- Reentrant
- Reusable
- Refreshable

Abends incurred by CBRUXSAE are sent to the caller's recovery routine; no additional ESTAE for this exit is provided. See Figure 9 on page 65 for a sample of the CBRUXSAE installation exit.

Sample CBRUXSAE Installation Exit

Figure 9 on page 65 shows the sample transaction security authorization installation exit, CBRUXSAE:

```
UXSAE    TITLE 'CBRUXSAE - SAMPLE OSREQ TX AUTH INSTALLATION EXIT'
CBRUXSAE START 0                   SAMPLE OSREQ TX AUTH INST EXIT
         SPACE 2
**** START OF SPECIFICATIONS ***************************************************
*                                                                               *
*    MODULE NAME:       CBRUXSAE                                              *
*                                                                               *
*    DESCRIPTIVE NAME:  SAMPLE OSREQ TRANSACTION SECURITY                     *
*                       AUTHORIZATION INSTALLATION EXIT                       *
*                                                                               *
*    FUNCTION:                                                                  *
*      MODULE CBRUXSAE IS INVOKED EACH TIME A REQUEST IS MADE TO              *
*      OAM VIA THE OSREQ INTERFACE.  CBRUXSAE MAY REFUSE TO ALLOW             *
*      THE USER TO PERFORM THE REQUESTED TRANSACTION BY RETURNING             *
*      A NON_ZERO RETURN CODE IN REGISTER 15.                                 *
*                                                                               *
*      THE INSTALLATION CAN PERFORM AUTHORIZATION CHECKING BY ANY             *
*      MEANS IT DEEMS REASONALBE.  FOR EXAMPLE:                               *
*         1. INVOKE RACF VIA THE SAF RACROUTE MACRO                           *
*         2. USE A TABLE-DRIVEN METHOD OF AUTHORIZATION CHECKING,             *
*            USING A DATASET OF USERIDS AND THE COLLECTIONS/OBJECTS           *
*            A USER IS AUTHORIZED TO PERFORM FUNCTIONS AGAINST.               *
*      THE AUTHORIZATION CHECKING MAY BE AT THE GRANULARITY THAT              *
*      THE INSTALLATION DECIDES IS NECESSARY, USING THE VALUES                *
*      PASSED IN TO THIS EXIT.                                                *
*                                                                               *
*    NOTES:                                                                     *
*      THIS SAMPLE RETURNS WITH A RETURN CODE OF 16, TELLING OAM              *
*      TO CONTINUE PROCESSING.                                                *
*                                                                               *
*      DEPENDENCIES:          MVS/SP VERSION 4.3.0                            *
*                             DFSMS/MVS 1.2.0                                 *
*                                                                               *
*      CHARACTER CODE:        EBCDIC                                          *
*                                                                               *
*      RESTRICTIONS:          NONE                                            *
*                                                                               *
*      REGISTER CONVENTIONS:                                                  *
*        R0  - UNPREDICTABLE                                                  *
*        R1  - STANDARD LINKAGE REGISTER                                      *
*        R2  - UNPREDICTABLE                                                  *
*        R3  - UNPREDICTABLE                                                  *
*        R4  - UNPREDICTABLE                                                  *
*        R5  - UNPREDICTABLE                                                  *
*        R6  - UNPREDICTABLE                                                  *
*        R7  - UNPREDICTABLE                                                  *
*        R8  - UNPREDICTABLE                                                  *
*        R9  - ADDRESS OF AUTODATA AREA FOR EXIT                              *
*        R10 - UNPREDICTABLE                                                  *
*        R11 - INPUT BASE REGISTER                                            *
*        R12 - CBRUXSAE BASE REGISTER                                         *
*        R13 - STANDARD LINKAGE REGISTER                                      *
*            - SAVE AREA ADDRESS                                              *
*        R14 - STANDARD LINKAGE REGISTER                                      *
*            - RETURN POINT ADDRESS                                           *
*        R15 - STANDARD LINKAGE REGISTER                                      *
*            - ENTRY POINT ADDRESS                                            *
*            - RETURN CODE                                                    *
*                                                                               *
```

Figure 9. Sample CBRUXSAE Installation Exit (Part 1 of 4)

```
*    MODULE TYPE:            CONTROL SECTION                       *
*                                                                  *
*      PROCESSOR:            ASSEMBLER H                           *
*                                                                  *
*      ATTRIBUTES:                                                 *
*                                                                  *
*        LOCATION:           LINKLIB                               *
*        STATE:              PROBLEM OR SUPERVISOR (CALLER)        *
*        AMODE:              31                                    *
*        RMODE:              ANY                                   *
*        KEY:                KEY OF CALLER                         *
*        MODE:               TASK                                  *
*        SERIALIZATION:      UNLOCKED                              *
*        TYPE:               REENTRANT, REUSABLE, REFRESHABLE      *
*        AUTHORIZATION:      NONE                                  *
*                                                                  *
*      LINKAGE:              STANDARD LINKAGE CONVENTIONS          *
*                                                                  *
*      CALLING SEQUENCE:                                           *
*        CBRUXSAE IS INVOKED IN THE USER'S ADDRESS SPACE USING THE *
*        MVS LINK MACRO                                            *
*                                                                  *
*      INPUT:                                                      *
*        REGISTER 1 WILL CONTAIN THE ADDRESS OF A PARAMETER LIST   *
*        WHICH WILL CONTAIN 4 POINTERS:                            *
*           1. POINTER TO 8 CHARACTER FIELD WHICH CONTAINS THE     *
*              OSREQ FUNCTION BEING PERFORMED                      *
*              POSSIBLE FUNCTIONS ARE:  STORE                      *
*                                       RETRIEVE                   *
*                                       CHANGE                     *
*                                       QUERY                      *
*                                       DELETE                     *
*           2. POINTER TO 44 CHARACTER FIELD WHICH CONTAINS THE    *
*              OBJECT NAME ASSOCIATED WITH THE REQUESTED FUNCTION  *
*           3. POINTER TO 44 CHARACTER FIELD WHICH CONTAINS THE    *
*              COLLECTION NAME ASSOCIATED WITH THE REQUESTED FUNCTION *
*           4. POINTER TO 8 CHARACTER FIELD WHICH CONTAINS THE     *
*              USERID ASSOCIATED WITH THE REQUESTED FUNCTION       *
*        REGISTER 9 WILL CONTAIN THE ADDRESS OF A 1024 BYTE AREA OF*
*        STORAGE WHICH CAN BE USED AS THIS PROGRAM'S AUTOMATIC STORAGE*
*                                                                  *
*      OUTPUT:                                                     *
*        A RETURN CODE IS PLACED IN REGISTER 15:                   *
*          CODE  MEANING                                           *
*            0   USER IS AUTHORIZED TO PERFORM THIS FUNCTION       *
*           16   USER IS AUTHORIZED TO PERFORM THIS FUNCTION, DO  @01A*
*                NOT CALL THIS EXIT AGAIN (BYPASS THIS METHOD OF  @01A*
*                AUTHORIZATION CHECKING                           @01A*
*          NON-ZERO OTHER THAN 16                                 @01C*
*                ANY NON-ZERO RC, OTHER THAN 16 IS TAKEN TO MEAN  @01C*
*                THE USER IS NOT AUTHORIZED TO PERFORM THIS FUNCTION. *
*                THE INSTALLATION CAN SPECIFY DIFFERENT RETURN CODES  *
*                TO MEAN DIFFERENT TYPES OF AUTHORIZATION FAILURES.   *
*                THE NON-ZERO RETURN CODE RETURNED BY THIS EXIT WILL  *
*                BE PRESENTED TO THE CALLER IN THE THIRD BYTE OF THE  *
*                FAILING REASON CODE.                              *
*                                                                  *
```

Figure 9. Sample CBRUXSAE Installation Exit (Part 2 of 4)

```
*      EXIT NORMAL:                                                    *
*        RETURN TO THE CALLER WITH RETURN CODE 0 OR NON-ZERO           *
*        RETURN CODE, INDICATING AUTHORIZATION FAILURE                 *
*                                                                      *
*      EXIT ERROR:  NONE                                               *
*                                                                      *
*    EXTERNAL REFERENCES:                                              *
*                                                                      *
*      ROUTINES:  NONE                                                 *
*                                                                      *
*      CONTROL BLOCKS:  NONE                                           *
*                                                                      *
*    EXECUTABLE MACROS:                                                *
*      RETURN                                                          *
*      SAVE                                                            *
*                                                                      *
*    MESSAGES:  NONE                                                   *
*                                                                      *
*    ABEND CODES:  NONE                                                *
*                                                                      *
*    CHANGE ACTIVITY:                                                  *
*                                                                      *
*     $L0=OW20657 1B0 950501 TUCLJT: Initial release                   *
*                                                                      *
*     $01=OW36250 1E0 990104 TUCLJT: Change default to return a   @01A*
*                                    RC=16 to indicate that the   @01A*
*                                    exit is not used, therefore  @01A*
*                                    should not be invoked again  @01A*
*                                    (Roll up of OW35784 1C0, 1D0)@01A*
*                                                                      *
**** END OF SPECIFICATIONS ***********************************************
         TITLE 'CBRUXSAE INPUT PARAMETERS'
*--------------------------------------------------------------------*
*                                                                    *
*        MODULE INPUT PARAMETER DEFINITIONS                          *
*                                                                    *
*--------------------------------------------------------------------*
UXSAEINP DSECT ,
FUNC_PTR DS    1F                  ADDRESS OF FUNCTION
OBJN_PTR DS    1F                  ADDRESS OF OBJECT NAME
COLN_PTR DS    1F                  ADDRESS OF COLLECTION NAME
USER_PTR DS    1F                  ADDRESS OF USERID
SAVE     DS    CL72                SAVE AREA
DATDPTR  DS    1F                  AUTO DATA AREA ADDRESS
         SPACE 2
         TITLE 'CBRUXSAE WORKING STORAGE'
*--------------------------------------------------------------------*
*                                                                    *
*        MODULE AUTO DATA AREA DEFINITIONS                           *
*                                                                    *
*--------------------------------------------------------------------*
WORKAREA DSECT ,                    CBRUXSAE AUTO DATA AREA
SAVEAREA DS    18F                 SAVE AREA
         DS    CL440               AVAILABLE STORAGE
WORKLEN  EQU   *-WORKAREA
         SPACE 2
```

Figure 9. Sample CBRUXSAE Installation Exit (Part 3 of 4)

```
         TITLE 'STANDARD REGISTER DEFINITIONS'
*--------------------------------------------------------------------*
*                                                                    *
*        STANDARD REGISTER DEFINITIONS                               *
*                                                                    *
*--------------------------------------------------------------------*
R0       EQU   0                   GENERAL REGISTER 0
R1       EQU   1                   GENERAL REGISTER 1
R2       EQU   2                   GENERAL REGISTER 2
R3       EQU   3                   GENERAL REGISTER 3
R4       EQU   4                   GENERAL REGISTER 4
R5       EQU   5                   GENERAL REGISTER 5
R6       EQU   6                   GENERAL REGISTER 6
R7       EQU   7                   GENERAL REGISTER 7
R8       EQU   8                   GENERAL REGISTER 8
R9       EQU   9                   GENERAL REGISTER 9
R10      EQU   10                  GENERAL REGISTER 10
R11      EQU   11                  GENERAL REGISTER 11
R12      EQU   12                  GENERAL REGISTER 12
R13      EQU   13                  GENERAL REGISTER 13
R14      EQU   14                  GENERAL REGISTER 14
R15      EQU   15                  GENERAL REGISTER 15
*--------------------------------------------------------------------*
*        MISCELLANEOUS CONSTANT VALUES                               *
*--------------------------------------------------------------------*
UXSAEDIS EQU   16                 RC=16 TELLS OSR TO DISABLE      @01A
*                                 FURTHER CALLS TO THIS SECURITY  @01A
*                                 AUTHORIZATION EXIT AND HANDLE   @01A
*                                 SUBSEQUENT INVOCATIONS AS       @01A
*                                 AUTHORIZED USERS                @01A
*--------------------------------------------------------------------*
*--------------------------------------------------------------------*
         TITLE 'CBRUXSAE - SAMPLE OSREQ TX AUTH INSTALLATION EXIT'
*--------------------------------------------------------------------*
*                                                                    *
*        CBRUXSAE ENTRY POINT                                        *
*                                                                    *
*--------------------------------------------------------------------*
CBRUXSAE CSECT ,                  SAMPLE OSREQ TX AUTH INST EXIT
CBRUXSAE AMODE 31
CBRUXSAE RMODE ANY
         SAVE  (14,12),,          SAVE CALLER'S REGISTERS AND        +
               'CBRUXSAE'            MARK ENTRY POINT
         LR    R12,R15            COPY ENTRY POINT ADDRESS
         USING CBRUXSAE,R12       CBRUXSAE BASE REGISTER
         USING WORKAREA,R9        ADDRESSIBILITY TO DATA AREA
         ST    R13,SAVEAREA+4     BACKWARD CHAIN SAVE AREAS
         LA    R0,SAVEAREA        CBRUXSAE SAVE AREA ADDRESS
         ST    R0,8(,R13)         FORWARD CHAIN SAVE AREAS
         LR    R13,R0             SET CBRUXSAE SAVE AREA ADDRESS
         LR    R11,R1             STORE PARAMETERS IN DATA AREA
         USING UXSAEINP,R11       ADDRESSIBILITY TO PARAMETERS
         SPACE 2
*--------------------------------------------------------------------*
*                                                                    *
*        RETURN TO THE CALLER                                        *
*                                                                    *
*--------------------------------------------------------------------*
EXIT     DS    0H
         L     R13,SAVEAREA+4     RESTORE CALLER'S SAVE AREA
         LA    R10,UXSAEDIS       SET DISABLE RETURN CODE         @01A
         LR    R15,R10            SAVE RETURN CODE                @01C
         RETURN  (14,12),         RESTORE CALLER'S REGISTERS, THEN   +
               RC=(15)              RETURN TO CALLER
         SPACE 2
         END   CBRUXSAE
```

Figure 9. Sample CBRUXSAE Installation Exit (Part 4 of 4)

Appendix E. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen-readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen-readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using it to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Volume I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Information Enabling Requests
Dept. DZWA
5600 Cottle Road
San Jose, CA 95193 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Programming Interface Information

This publication does not contain programming interface information.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States, or other countries, or both:

BookManager	DFSMS/MVS	IMS	MVS/SP
CICS	ESA/370	IMS/ESA	OS/390
DATABASE2	ESA/390	MVS	RACF
DB2	GDDM	MVS/DFP	z/OS
DFSMSdfp	IBM	MVS/ESA	

The following terms are trademarks of other companies:

LMSI Laser Magnetic Storage International

Glossary

The terms in this glossary are defined as they pertain to the Object Access Method.

This glossary may include terms and definitions from:

- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright (ANSI). Copies may be purchased from the American National Standards Institute, 11 West 42nd Street, New York 10036.
- The *Information Technology Vocabulary*, developed by Subcommittee 1, Joint Technical Committee 1, of the International Electrotechnical Commission (ISO/IEC JTC2/SC1).
- IBM Dictionary of Computing, New York: McGraw-Hill, 1994.

A

access path. The path DB2 uses to get to data specified in SQL statements. An access path can involve an index, a sequential search, or a combination of both.

ACS. Automatic class selection.

application plan. The control structure produced during the bind process and used by DB2 to process SQL statements during application execution.

attribute. A named property of an entity.

automatic class selection (ACS). Routines that determine the storage class, management class, and storage group for a collection. The storage administrator is responsible for establishing ACS routines appropriate to an installation's storage requirements.

B

bind. The process by which the output from the DB2 precompiler is converted to a usable control structure called an application plan. This process is the one during which access paths to the data are selected and some authorization checking is performed.

block. See *sector*.

C

CAF. Call attachment facility.

call attachment facility (CAF). A DB2 attachment facility that allows application programs to connect to and use DB2.

cartridge. See *optical cartridge*.

Channel-to-channel (CTC). A method of connecting two computing devices.

CICS. Customer Information Control System.

class transition. A change in an object's management class or storage class when an event occurs that brings about a change in an object's service level or management criteria. Class transition occurs during a storage management cycle.

collection. A group of objects that have similar performance, availability, backup, retention, and class transition characteristics. A collection is used to catalog a large number of objects which, if cataloged separately, could require an extremely large catalog.

commit. In DB2, to cause all changes that have been made to the database file since the last commitment operation to become permanent, and the records to be unlocked so they are available to other users.

CTC. Channel-to-channel.

D

data class. A list of allocation attributes that the system uses for the creation of data sets.

DASD. Direct Access Storage Device.

DATABASE 2. A relational database management system.

DATABASE 2 interactive. An interactive relational database management program.

DB2. IBM DATABASE 2.

DB2I. DATABASE 2 interactive.

DFSMSdfp. Data Facility Storage Management Subsystem data facility product.

DFSMS/MVS. Data Facility Storage Management Subsystem/Multiple Virtual Storage.

disk. See *optical disk*.

G

gigabyte. When referring to storage capacity, two to the thirtieth power; 1 073 741 824 in decimal notation.

grant. A DB2 process that authorizes users to access data.

GTF. Generalized trace facility.

I

ICF. Integrated catalog facility.

ID. Identification.

image copy. An exact reproduction of all or part of a table space. DB2 provides utilities to make full image copies or incremental image copies.

IMS. Information Management System.

index. A set of pointers that are logically ordered by the values of a key. Indexes provide quick access to data and can enforce uniqueness on the rows in a DB2 storage table.

installation-wide exit. The means specifically described in an IBM software product's documentation by which an IBM software product may be modified by a customer's system programmers to change or extend the functions of the IBM software product. Such modifications consist of exit routines written to replace one or more existing modules of an IBM software product, or to add one or more modules or subroutines to an IBM software product, for the purpose of modifying (including extending) the functions of the IBM software product.

Interactive System Productivity Facility. An interactive base for ISMF.

IPL. Initial program load.

ISMF. Interactive Storage Management Facility.

ISO. International Organization for Standardization.

ISPF. Interactive System Productivity Facility.

L

LCS. Library Control System.

Library Control System. Component of OAM that writes and reads objects on optical disk storage, and manipulates the optical volumes on which the objects reside.

M

management class. A named collection of management attributes describing the retention, backup, and storage class transition characteristics for a group of objects in an object storage hierarchy.

MVS/ESA. Multiple Virtual Storage/Enterprise System Architecture.

O

OAM. Object Access Method.

OAM Storage Management Component (OSMC). Determines where object should be stored, manages object movement within the objects storage hierarchy, and manages expiration attributes based on the installation storage management policy.

object. A named byte stream having no specific format or orientation.

Object Access Method (OAM). A program that provides object storage, object retrieval, and object storage hierarchy management. OAM isolates applications from storage devices, storage management, and storage device hierarchy management.

Object Storage and Retrieval (OSR). Component of OAM that stores, retrieves, and deletes objects. OSR stores objects in the storage hierarchy and maintains the information about these objects in DB2 databases.

Object Storage Request macro (OSREQ). This macro serves as an application program interface for storing, retrieving, and deleting objects using OAM.

optical cartridge. A plastic case that protects and contains the optical disk and permits insertion into an optical drive.

optical disk. A disk that uses laser technology for data storage and retrieval.

optical disk drive. The mechanism used to seek, read, and write data on an optical disk. An optical disk drive may reside in an optical library or as a stand-alone unit.

optical library. A disk storage device that houses optical disk drives and optical disks, and contains a mechanism for moving optical disks between a storage area and optical disk drives.

optical volume. One side of a double-sided optical disk.

OSMC. OAM Storage Management Component.

OSR. Object Storage and Retrieval.

OSREQ. Object Storage Request macro.

OVTOC. Optical volume table of contents.

P

pseudo optical library. A set of shelf-resident optical volumes associated with either a stand-alone or an operator-accessible optical disk drive; see also *real optical library*.

R

real optical library. Physical storage device that houses optical disk drives and optical cartridges, and contains a mechanism for moving optical disks between a cartridge storage area and optical disk drives; see also *pseudo optical library*.

row. The horizontal component of a DB2 table. A row consists of a sequence of values, one for each column of a table.

S

SCDS. Source control data set.

sector. On disk storage, an addressable subdivision of a track used to record one block of a program or data.

shelf-resident optical volume. An optical volume that resides outside of an optical library.

SMP/E. System Modification Program/Extended.

SMS. Storage Management Subsystem.

SPUFI. SQL processing using file input.

SQL. Structured query language.

SQLCODE. Structured query language return code.

SQL Processing Using File Input. Used to perform groups of SQL statements in batch or online mode. SPUFI is option one under DB2I.

stand-alone optical drive. An optical drive housed outside of an optical library.

storage class. A named list of storage attributes. The list of attributes identifies a storage service level provided for data associated with the storage class. No physical storage is directly implied or associated with a given storage class name.

storage group. A named collection of physical devices to be managed as a single object storage area. It consists of an object directory (DB2 table space) and object storage on DASD (DB2 table spaces), with optional library-resident and shelf-resident optical volumes.

storage hierarchy. An arrangement in which data can be stored in several types of storage devices that have different characteristics, such as capacity and speed of access.

storage management cycle. An invocation of the OAM Storage Management Component (OSMC). The purpose of the storage management cycle is to ensure that every object scheduled for processing is placed in the proper level of the object storage hierarchy (as specified by its storage class), is expired or is backed up (as specified by its management class or by an explicit application request), and, if necessary, is flagged for action during a subsequent storage management cycle.

structured query language. A DB2 query tool.

System Modification Program/Extended. Basic tool for installing software changes in programming systems. It controls these changes at the element (module or macro) level, which helps protect system integrity.

T

table. In DB2, a named data object consisting of a specific number of columns and some number of unordered rows.

table space. A page set used to store the records of one or more DB2 tables.

TSO. Time Sharing Option.

U

user exit. A programming service provided by an IBM software product that may be requested by an application program for the service of transferring control back to the application program upon the later occurrence of a user-specified event.

V

vary offline. To change the status of an optical library or an optical drive from online to offline. Varying a library offline does not affect the online/offline status of the drives it contains. When a library or drive is offline, no data may be accessed on optical disks through the offline drive or the drives in the offline library.

vary online. To change the status of an optical library or an optical drive from offline to online. This makes the drive or drives in the library being varied online available for the optical disk access.

Index

D

E

F

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

Readers' Comments — We'd Like to Hear from You

z/OS
Object Access Method Application
Programmer's Reference

Publication No. SC35-0425-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC35-0425-01

IBM®

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department 61C
9000 South Rita Road
TUCSON AZ 85775-4401

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5694-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC35-0425-01