z/OS

DFSMS Object Access Method Application Programmer's Reference

Version 2 Release 1

SC23-6865-00

Note

Before using this information and the product it supports, read the information in “Notices” on page 99.

This edition applies to Version 2 Release 1 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC35-0425-09.

© Copyright IBM Corporation 1986, 2013.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures

© Copyright IBM Corp. 1986, 2013

Tables

© Copyright IBM Corp. 1986, 2013

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 9 contains detailed information about the OSREQ macro and how it is used by application programs.
- Appendix A, "Sample program for object storage," on page 55 provides assembler source code for a sample object storage request interface.
- Appendix B, "Reason codes," on page 75 provides error descriptions and recommended responses for OAM return codes and reason codes.
- Appendix C, "Performance considerations and object data reblocking," on page 83 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 85 details how this exit is used to provide security checking for the OSREQ macro.
- "Glossary" on page 103 defines acronyms, abbreviations, and terms used in this document.

Required product knowledge

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

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

z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS®, see *z/OS Information Roadmap*.

To find the complete z/OS library, including the z/OS Information Center, see z/OS Internet Library (http://www.ibm.com/systems/z/os/zos/bkserv/).

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:

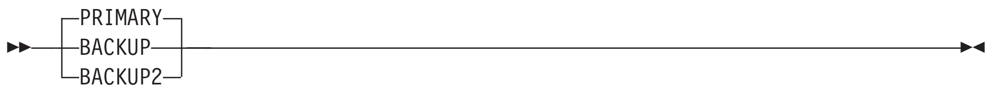
- Read the syntax diagrams from left to right and from top to bottom.
- Each syntax diagram begins with a double arrowhead (►►) and ends with opposing arrows (►◄).
- An arrow (─►) at the end of a line indicates that the syntax continues on the next line. A continuation line begins with an arrow (►─).
- 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**.

```
►►──┬─L─────────────────────────────┬──────────────────────────────────►◄
    ├─(M,parameter_list─┬───────────┬─)─┤
    │                   └─,COMPLETE─┘   │
    └─(E,parameter_list─┬───────────┬─)─┘
                        └─,COMPLETE─┘
```

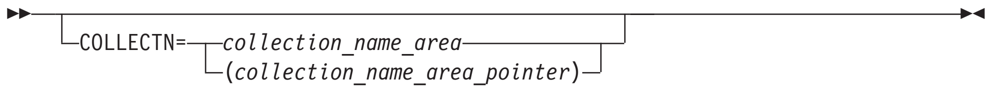
- If a stack is placed below the main path, a keyword is optional, and you can choose one or none. In the following example, **TOKEN**, **COLLECTN**, and **NAME** are optional keywords. You can choose any one of the three.

```
►►──┬──────────┬───────────────────────────────────────────────────────►◄
    ├─COLLECTN─┤
    ├─NAME─────┤
    └─TOKEN────┘
```

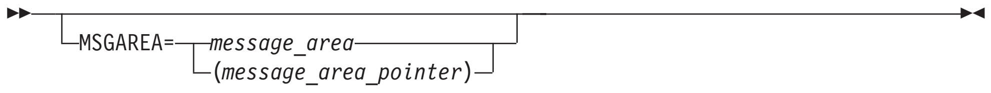
- Where you can choose from two or more keywords and one of the keywords appears above the main path, that keyword is the default. You may choose one or the other of the keywords, but if none is entered, the default keyword is automatically selected. In the following example you may choose either **PRIMARY**,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

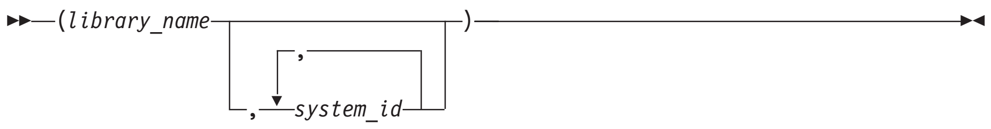


- You must provide all items enclosed in parentheses ( ). You must include the parentheses. In the following example, you must supply the volume serial number (*message_area_pointer*) and it must be enclosed in parentheses.



- 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.

How to send your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or provide any other feedback that you have.

Use one of the following methods to send your comments:

1. Send an email to mhvrcfs@us.ibm.com.
2. Send an email from the "Contact us" web page for z/OS (http://www.ibm.com/systems/z/os/zos/webqs.html).
3. Mail the comments to the following address:
   IBM Corporation
   Attention: MHVRCFS Reader Comments
   Department H6MA, Building 707
   2455 South Road
   Poughkeepsie, NY 12601-5400
   US
4. Fax the comments to us, as follows:
   From the United States and Canada: 1+845+432-9405
   From all other countries: Your international access code +1+845+432-9405

Include the following information:

- Your name and address.
- Your email address.
- Your telephone or fax number.
- The publication title and order number:
  z/OS V2R1.0 DFSMS OAM Application Programmer's Reference
  SC23-6865-00
- The topic and page number that is related to your comment.
- The text of your comment.

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

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

Do not use the feedback methods that are listed for sending comments. Instead, take one of the following actions:

- Contact your IBM service representative.
- Call IBM technical support.
- Visit the IBM Support Portal at z/OS support page (http://www.ibm.com/systems/z/support/).

© Copyright IBM Corp. 1986, 2013

Summary of changes

z/OS Version 2 Release 1 summary of changes

See the following publications for all enhancements to z/OS Version 2 Release 1 (V2R1):

- *z/OS Planning for Installation*
- *z/OS Introduction and Release Guide*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Migration*

© Copyright IBM Corp. 1986, 2013

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 can contain 1 byte or up to 2000 MB (2 097 152 000 bytes) of data. The maximum object size for the disk and tape levels of the OAM storage hierarchy is 2000 MB. The maximum object size for the optical level of the OAM storage hierarchy is 256 MB (268 435 456 bytes).

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

© Copyright IBM Corp. 1986, 2013

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 z/OS 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 a file system, tape volumes, or optical disk storage, and 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 disk (DB2 or file system), 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 disk (DB2 or file system), 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 (disk sublevel 1 (DB2), disk sublevel 2 (file system), optical, tape sublevel 1, or tape sublevel 2).

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* parmlib member. 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* 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
- MVS™ batch
- TSO

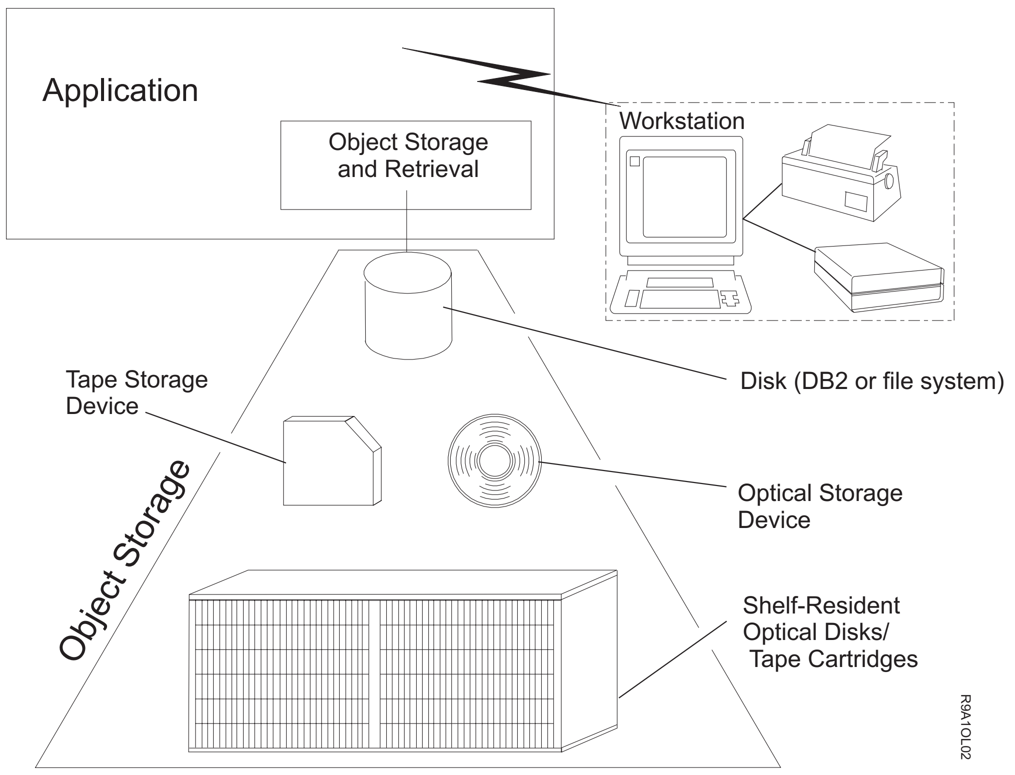

Application
Object Storage and Retrieval
Workstation
Disk (DB2 or file system)
Tape Storage Device
Optical Storage Device
Object Storage
Shelf-Resident Optical Disks/ Tape Cartridges
R9A1OL02

Figure 1. Example of devices that application may use

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 55 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 disk 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. Objects greater than 256 megabytes cannot be retrieved using a single OSREQ Retrieve. To retrieve an object greater than 256 megabytes, the object must be retrieved in pieces using multiple OSREQ Retrieves specifying the offset and length (maximum length allowed for each piece is 256 megabytes).

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).

Note: When objects are stored directly to the file system sublevel from an application program the application must perform the DB2 "commit" within 24 hours of storing the object. Failure to do this will ultimately result in loss of object data stored in the file system.

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.

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 25.

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. IBM High Level Assembler (HLASM) is required to assemble this macro. For a list of books that contain more information about HLSAM, see "About this book" on page ix.

See Appendix C, "Performance considerations and object data reblocking," on page 83 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 85 for information on and a sample of the CBRUXSAE security authorization installation exit that is used at the OSREQ macro level.

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. It is also used to change an object's deletion-hold status and to inform OAM of an external event trigger expiration criteria for an object in event-based-retention mode.
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.

Use Store for objects less than or equal to 256 megabytes. Use the Store Sequence functions (Storebeg, Storeprt, and Storeend) for storing objects greater than 256 megabytes.

Storebeg
Begins the Store Sequence processing of an object. Store Sequence processing can be used for an object whose total size is greater than 50 megabytes that is to be written to disk or tape (but not to optical). Store Sequence processing ***must*** be used for storing objects greater than 256 megabytes. See “Adding objects to the object storage hierarchy” on page 21 and “STOREBEG—Beginning a Store Sequence operation” on page 25 for more information.

Storeprt
Stores the next sequential contiguous part of an object being stored with Store Sequence processing. See “Adding objects to the object storage hierarchy” on page 21 and “STOREPRT—Storing an individual part in a Store Sequence operation” on page 27 for more information.

Storeend
Ends the Store Sequence processing of an object, either to complete the storage of the object or to effectively cancel the storage of the object. See “Adding objects to the object storage hierarchy” on page 21 and “STOREEND—Ending a Store Sequence operation” on page 29 for more information.

Unaccess
Frees the resources obtained with an OSREQ ACCESS request. The token cannot be used after the UNACCESS invocation.

“Implementing the functions” on page 11 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 “What you can do with OSREQ” on page 9. 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 40.

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 31 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 31.

- "ACCESS—Initializing the OSREQ interface" on page 12
- "CHANGE—Changing an object's management characteristics" on page 13

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

```
►►──OSREQ ACCESS──MF=──┬─L──────────────────────────────┬──────────────────►
                       ├─(M,parameter_list─┬──────────┬─)─┤
                       │                   └─,COMPLETE─┘  │
                       └─(E,parameter_list─┬──────────┬─)─┘
                                           └─,COMPLETE─┘

►──┬────────────────────────────────────┬──────────────────────────────────►
   │      (1)                           │
   └─TOKEN─────=──┬─token_area───────────┤
                  └─(token_area_pointer)─┘

►──┬──────────────────────────────────────────────────────────┬────────────►
   │         (2)                                               │
   └─IADDRESS─────=──┬─SQL_interface_module_address─────────┤
                     └─(SQL_interface_module_pointer)─┘

►──┬─────────────────────────────────────────┬─────────────────────────────►
   └─MSGAREA=──┬─message_area───────────────┤
               └─(message_area_pointer)─┘

►──┬───────────────────────────────────────┬───────────────────────────────►
   └─RETCODE=──┬─return_code──────────────┤
               └─(return_code_pointer)─┘

►──┬───────────────────────────────────────┬───────────────────────────────►
   └─REACODE=──┬─reason_code──────────────┤
               └─(reason_code_pointer)─┘

►──┬──────────────────────────────────────────┬────────────────────────────►◄
   └─TTOKEN=──┬─tracking_token───────────────┤
              └─(tracking_token_pointer)─┘
```

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 Table 1 for the effects of the IADDRESS parameter when used in various processing environments.

Table 1. IADDRESS parameter effects 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.		

Note: Environments or invocations other than those listed in Table 1 have not been tested by IBM and the results may be unpredictable. An example of an untested, unpredictable environment would be the DB2 Stored procedure environment.

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 IBM Information Management Software for z/OS Solutions Information Center.

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

```
►►──OSREQ CHANGE──┬──────────────────────────────────────────────┬──────────────────────►
                  └─MF=──┬─L──────────────────────────────────────┤
                         ├─(M,parameter_list──┬───────────┬──)─┤
                         │                    └─,COMPLETE─┘    │
                         └─(E,parameter_list──┬───────────┬──)─┘
                                              └─,COMPLETE─┘

►──┬────────────────────────────────────────────┬────────────────────────────────────────►
   │        (1)                                 │
   └─TOKEN────────=──┬─token_area───────────────┤
                     └─(token_area_pointer)─────┘

►──┬─────────────────────────────────────────────────────────────┬───────────────────────►
   │           (1)                                             │
   └─COLLECTN─────────=──┬─collection_name_area──────────────────┤
                         └─(collection_name_area_pointer)────────┘

►──┬───────────────────────────────────────────────────┬─────────────────────────────────►
   │       (1)                                         │
   └─NAME────────=──┬─object_name_area─────────────────┤
                    └─(object_name_area_pointer)───────┘

►──┬───────────────────────────────────────────────────────────┬─────────────────────────►
   │           (2)                                             │
   └─STORCLAS─────────=──┬─storage_class_area──────────────────┤
                         └─(storage_class_area_pointer)────────┘

►──┬────────────────────────────────────────────────────────────────┬────────────────────►
   │           (2)                                                  │
   └─MGMTCLAS─────────=──┬─management_class_area────────────────────┤
                         └─(management_class_area_pointer)──────────┘

►──┬───────────────────────────────────────────────────────────┬─────────────────────────►
   │        (2)                                                │
   ├─RETPD────────────=──┬─retention_period────────────────────┤
   │                     └─(retention_period_pointer)──────────┤
   │           (3)                                             │
   └─EVENTEXP────────────=──┬─number_of_days───────────────────┘
                            └─(number_of_days_pointer)─────────┘

►──┬───────────────────────────────────┬──┬──────────────────────────────────────────┬───►
   │           (4)                     │  └─MSGAREA=──┬─message_area─────────────────┤
   └─DELHOLD─────────=──┬─HOLD─────────┤              └─(message_area_pointer)───────┘
                        └─(NOHOLD)─────┘

►──┬───────────────────────────────────────────┬─────────────────────────────────────────►
   └─RETCODE=──┬─return_code───────────────────┤
               └─(return_code_pointer)─────────┘

►──┬───────────────────────────────────────────┬─────────────────────────────────────────►
   └─REACODE=──┬─reason_code───────────────────┤
               └─(reason_code_pointer)─────────┘

►──┬──────────────────────────────────────────────┬──────────────────────────────────────►◄
   └─TTOKEN=──┬─tracking_token────────────────────┤
              └─(tracking_token_pointer)──────────┘
```

Notes:

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

2 These keyword parameters result in object's pending action date set to current date.

3 The EVENTEXP keyword cannot be issued in the same statement as the RETPD keyword. Also, EVENTEXP is valid only if the object is in event-based-retention mode (for example: the expiration date is 0002-02-02 as a result of RETPD=-2 (X'FFFFFFFE') being specified on a previous STORE or CHANGE request). If EVENTEXP is specified on a CHANGE request when the expiration date is anything other than 0002-02-02, the CHANGE request fails.

4 The DELHOLD keyword issued without any type (2) keywords will not result in ACS routines run or pending action date set.

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 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 one of the following conditions is met:
  - RETPD was set to -1 (X'FFFFFFFF'), in which case the expiration date is set to the reserved value '0001–01–01' and the expiration date for the object is then based solely on the object's management class expiration attributes.
  - RETPD was set to -2 (X'FFFFFFFE'), in which case the expiration date is set to the reserved value '0002–02–02' and the expiration date for the object is

dependent on receipt of notification of an external event by an OSREQ CHANGE that includes the EVENTEXP keyword.

- 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.

  Note: Special rules apply for retention-protected objects. See "Expiration date processing" on page 44 to see the rules in more detail.
- 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 93 000, and none of the preceding conditions apply, 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 Table 2 on page 45).

If the EVENTEXP parameter is specified, a new expiration date is calculated using one of the following two formulas. The formula used is the one that produces the earliest expiration date.

- Today + the number of days specified with the EVENTEXP keyword
- The object's creation date + the maximum retention limit for the object's management class.

If the object is retention-protected and the retention date (contained in ODRETDT in the object directory) is later than the expiration date determined by these formulas, then the expiration date is set to the retention date.

See "Expiration date processing" on page 44 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 storage used for the object data is released. Primary object data stored on disk sublevel 1 (DB2), disk sublevel 2 (file system), optical, tape sublevel 1, or tape sublevel 2, 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*.

Note: The object cannot be deleted and the OSREQ DELETE will fail if either of the following are true:

1. The object is in deletion-hold mode
2. Retention-protection or deletion-protection are enabled and the object's expiration date is the special value 0002–02–02 or the explicit or calculated expiration date is later than the current date.

Syntax for OSREQ DELETE

```
►►──OSREQ DELETE──MF=──┬─L──────────────────────────────────┬──────────────►
                       ├─(M,parameter_list─┬───────────┬─)─┤
                       │                   └─,COMPLETE─┘   │
                       └─(E,parameter_list─┬───────────┬─)─┘
                                           └─,COMPLETE─┘
```

```
►─┬────────────────────────────────────────────┬──────────────────────►
  │      (1)                                   │
  └─TOKEN─────=─┬─token_area─────────────────┬─┘
                └─(token_area_pointer)───────┘

►─┬────────────────────────────────────────────────────────┬──────────►
  │         (1)                                            │
  └─COLLECTN─────=─┬─collection_name_area────────────────┬─┘
                   └─(collection_name_area_pointer)──────┘

►─┬──────────────────────────────────────────────────┬────────────────►
  │     (1)                                          │
  └─NAME─────=─┬─object_name_area──────────────────┬─┘
               └─(object_name_area_pointer)────────┘

►─┬──────────────────────────────────────────┬────────────────────────►
  └─MSGAREA=─┬─message_area────────────────┬─┘
             └─(message_area_pointer)──────┘

►─┬─────────────────────────────────────────┬─────────────────────────►
  └─RETCODE=─┬─return_code────────────────┬─┘
             └─(return_code_pointer)──────┘

►─┬─────────────────────────────────────────┬─────────────────────────►
  └─REACODE=─┬─reason_code────────────────┬─┘
             └─(reason_code_pointer)──────┘

►─┬─────────────────────────────────────────┬────────────────────────►◄
  └─TTOKEN=─┬─tracking_token──────────────┬─┘
            └─(tracking_token_pointer)────┘
```

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 49.

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 one of the following methods:

1. 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.

2. Substituting one or more percent signs (%) and/or underscores (_) anywhere in the object name. The percent sign character is interpreted as a wildcard to replace zero or more characters in the object name. The underscore character represents a single character. 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.M%.P_L% 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.

Note: The two methods for setting up a generic search are mutually exclusive. You cannot mix asterisk wildcards with either percent sign or underscore wildcards in a single QUERY request. The generic search is only supported for OSREQ QUERY requests.

The syntax diagram for the OSREQ QUERY function follows.

Syntax for OSREQ QUERY

```
►►──OSREQ QUERY──MF=──┬─L──────────────────────────────┬──────────────────────────►
                      ├─(M,parameter_list─┬───────────┬─)─┤
                      │                   └─,COMPLETE─┘   │
                      └─(E,parameter_list─┬───────────┬─)─┘
                                          └─,COMPLETE─┘

►──┬────────────────────────────────────────┬─────────────────────────────────────►
   │       (1)                              │
   └─TOKEN──────=──┬─token_area───────────┬─┘
                   └─(token_area_pointer)─┘

►──┬────────────────────────────────────────────────────────────┬─────────────────►
   │          (1)                                               │
   └─COLLECTN──────=──┬─collection_name_area───────────┬─┘
                      └─(collection_name_area_pointer)─┘

►──┬─────────────────────────────────────────────────┬────────────────────────────►
   │      (1)                                        │
   └─NAME──────=──┬─object_name_area───────────┬─┘
                  └─(object_name_area_pointer)─┘

►──┬───────────────────────────────────────┬──────────────────────────────────────►
   │     (2)                               │
   └─QEL──────=──┬─query_list───────────┬─┘
                 └─(query_list_pointer)─┘

►──┬────────────────────────────────────────┬─────────────────────────────────────►
   └─MSGAREA=──┬─message_area───────────┬─┘
               └─(message_area_pointer)─┘

►──┬───────────────────────────────────────┬──────────────────────────────────────►
   └─RETCODE=──┬─return_code───────────┬─┘
               └─(return_code_pointer)─┘

►──┬───────────────────────────────────────┬──────────────────────────────────────►
   └─REACODE=──┬─reason_code───────────┬─┘
               └─(reason_code_pointer)─┘
```

```
►─┬───────────────────────────────────────┬──────────────────────────────────────►◄
  └─TTOKEN=─┬─tracking_token────────────┬─┘
            └─(tracking_token_pointer)──┘
```

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 35.

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. Objects greater than 256 megabytes cannot be retrieved using a single OSREQ Retrieve. To retrieve an object greater than 256 megabytes, an object must be retrieved in pieces using multiple OSREQ Retrieves specifying the offset and length (maximum length allowed for each piece is 256 megabytes). The syntax diagram for the OSREQ RETRIEVE function follows.

Syntax for OSREQ RETRIEVE

```
►►──OSREQ RETRIEVE──MF=─┬─L──────────────────────────────────┬────────────────────►
                        ├─(M,parameter_list─┬───────────┬─)─┤
                        │                   └─,COMPLETE─┘   │
                        └─(E,parameter_list─┬───────────┬─)─┘
                                            └─,COMPLETE─┘

►─┬──────────────────────────────────────┬─────────────────────────────────────────►
  │        (1)                           │
  └─TOKEN──────=─┬─token_area──────────┬─┘
                 └─(token_area_pointer)─┘

►─┬────────────────────────────────────────────────────────┬───────────────────────►
  │           (1)                                          │
  └─COLLECTN──────=─┬─collection_name_area────────────┬────┘
                    └─(collection_name_area_pointer)──┘

►─┬──────────────────────────────────────────────┬─────────────────────────────────►
  │       (1)                                    │
  └─NAME──────=─┬─object_name_area────────────┬──┘
                └─(object_name_area_pointer)──┘

►─┬────────────────────────────────────────┬──┬─────────────────────┬──────────────►
  │          (2)                           │  │       ┌─PRIMARY─┐   │
  └─BUFLIST──────=─┬─buffer_list─────────┬─┘  └─VIEW=─┼─BACKUP──┼───┘
                   └─(buffer_list_pointer)─┘          └─BACKUP2─┘
```

```
►──┬──────────────────────────────────────────────┬──────────────────────────►
   └─OFFSET=──┬─offset_of_starting_byte───────────┤
              └─(offset_of_starting_byte_pointer)─┘

►──┬─────────────────────────────────┬───────────────────────────────────────►
   └─LENGTH=──┬─number_bytes─────────┤
              └─(number_bytes_pointer)─┘

►──┬──────────────────────────────────┬──────────────────────────────────────►
   └─MSGAREA=──┬─message_area─────────┤
               └─(message_area_pointer)─┘

►──┬─────────────────────────────────┬───────────────────────────────────────►
   └─RETCODE=──┬─return_code─────────┤
               └─(return_code_pointer)─┘

►──┬─────────────────────────────────┬───────────────────────────────────────►
   └─REACODE=──┬─reason_code─────────┤
               └─(reason_code_pointer)─┘

►──┬────────────────────────────────┬────────────────────────────────────────►
   └─RECALL=──┬─number_days─────────┤
              └─(number_days_pointer)─┘

►──┬───────────────────────────────────┬─────────────────────────────────────►
   └─RETCODE2=──┬─return_code2─────────┤
                └─(return_code2_pointer)─┘

►──┬───────────────────────────────────┬────────────────────────────────────►◄
   └─TTOKEN=──┬─tracking_token─────────┤
              └─(tracking_token_pointer)─┘
```

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 3 on page 47.

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 75) 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* parmlib member 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.

The RECALL keyword can be used to explicitly recall a full copy of an object from removable media to DB2 for the specified number of days at the time the object is retrieved. This can result in improved performance for subsequent retrieves of this object. Refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support* for more information on explicit and implicit recalls.

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

Adding objects to the object storage hierarchy

OAM provides these functions for adding objects to the object storage hierarchy:

STORE function
: STORE can be used for objects whose size is less than or equal to 256 megabytes that are to be written to the disk, tape, or optical levels of the storage hierarchy. STORE processing requires that the entire object be kept in storage. See “STORE function” on page 22 for more information.

Store Sequence functions
: Store Sequence can be used for objects whose size is greater than 50 megabytes that are to be written to the disk or tape (but not optical) levels of the storage hierarchy. Store Sequence processing handles objects in smaller chunks, rather than having the entire object in storage (as required by STORE processing), which can reduce the storage requirements for an application. See “STOREBEG—Beginning a Store Sequence operation” on page 25, “STOREPRT—Storing an individual part in a Store Sequence operation” on page 27, and “STOREEND—Ending a Store Sequence operation” on page 29 for more information.

The Store Sequence functions must be used when writing objects whose size is greater than 256 megabytes to disk or tape.

Objects whose size is greater than 50 megabytes and less than or equal to 256 megabytes can be written to disk or tape using either the STORE function or Store Sequence functions, thus providing flexibility when storing such objects.

When storing objects to DB2, Store Sequence processing always writes the objects to a LOB table. If LOB=N is specified on the OAM1 entry in the IEFSSN*xx* parmlib member, or if a LOB storage structure does not exist for the target object storage group, then an attempt to do a Store Sequence to DB2 will fail.

STORE function

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. Use STORE for objects less than or equal to 256 megabytes. See the store sequence functions STOREBEG, STOREPRT, and STOREEND for storing objects greater than 256 megabytes.

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 RETPD is set to -2 (X'FFFFFFFE'), the expiration date is set to special value '0002–02–02'. The object is considered in event-based-retention mode and the expiration date for the object will be derived when an OSREQ CHANGE request with the EVENTEXP keyword is received for this object. See Table 2 on page 45.
  - 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 Table 2 on page 45 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 93 000, and none of these conditions apply, expiration date is set to the creation date of the object plus the number of days specified in the RETPD.

See “Expiration date processing” on page 44 for more information.

- 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.
- Set the retention date:
  - If retention-protection is not enabled for the object's storage group or RETPD is -2 (X'FFFFFFFE'), the retention date is set to the reserved value '0001–01–01'.
  - If retention-protection is enabled for the object's storage group and the expiration date is set to special value '0001–01–01', the retention date is set to a value determined by the expiration date rules of the object's management class.
  - If retention-protection is enabled for the object's storage group and expiration date is set to any value other than '0001–01–01' or '0002–02–02', the retention date is set to the same value as the expiration date.

Syntax for OSREQ STORE

```
►►──OSREQ STORE──MF=──┬─L────────────────────────────────────┬──────────────────►
                      ├─(M,parameter_list──┬───────────┬─)─┤
                      │                    └─,COMPLETE─┘   │
                      └─(E,parameter_list──┬───────────┬─)─┘
                                           └─,COMPLETE─┘

►──┬────────────────────────────────────────────┬─────────────────────────────►
   │         (1)                                │
   └─TOKEN──────=──┬─token_area───────────┬─────┘
                   └─(token_area_pointer)─┘

►──┬────────────────────────────────────────────────────────────┬─────────────►
   │            (1)                                             │
   └─COLLECTN──────=──┬─collection_name_area───────────┬────────┘
                      └─(collection_name_area_pointer)─┘

►──┬───────────────────────────────────────────────────┬──────────────────────►
   │        (1)                                        │
   └─NAME──────=──┬─object_name_area───────────┬───────┘
                  └─(object_name_area_pointer)─┘

►──┬───────────────────────────────────────────────┬──────────────────────────►
   │           (2)                                 │
   └─BUFLIST──────=──┬─buffer_list───────────┬─────┘
                     └─(buffer_list_pointer)─┘

►──┬────────────────────────────────────────┬──┬──────────────────┬───────────►
   │        (1)                             │  │          ┌─NO──┐ │
   └─SIZE──────=──┬─object_byte───────────┬─┘  └─RELBUF=──┴─YES─┴─┘
                  └─(object_byte_pointer)─┘

►──┬────────────────────────────────────────────────┬─────────────────────────►
   └─STORCLAS=──┬─storage_class_area───────────┬────┘
                └─(storage_class_area_pointer)─┘
```

```
►─┬──────────────────────────────────────────────────┬──────────────────────►
  └─MGMTCLAS=─┬─management_class_area─────────────┬─┘
              └─(management_class_area_pointer)───┘

►─┬───────────────────────────────────────────┬────────────────────────────►
  └─RETPD=─┬─retention_period──────────────┬─┘
           └─(retention_period_pointer)────┘

►─┬─────────────────────────────────────────┬──────────────────────────────►
  └─MSGAREA=─┬─message_area─────────────┬──┘
             └─(message_area_pointer)───┘

►─┬────────────────────────────────────────┬───────────────────────────────►
  └─RETCODE=─┬─return_code─────────────┬──┘
             └─(return_code_pointer)───┘

►─┬────────────────────────────────────────┬───────────────────────────────►
  └─REACODE=─┬─reason_code─────────────┬──┘
             └─(reason_code_pointer)───┘

►─┬──────────────────────────────────────────┬─────────────────────────────►
  └─TTOKEN=─┬─tracking_token─────────────┬──┘
            └─(tracking_token_pointer)───┘

►─┬──────────────────────────────────────────┬─┬─────────────────────────┬─►◄
  └─RETCODE2=─┬─return_code2─────────────┬──┘ │        (3)              │
              └─(return_code2_pointer)───┘    └─DELHOLD─────=─┬─HOLD───┬┘
                                                              └─(NOHOLD)┘
```

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 3 on page 47.

3 If DELHOLD is not specified, the default value is DELHOLD=NOHOLD.

Processing a store to a new collection

The following section describes new collection processing for an OSREQ Store type request which includes an OSREQ STORE request as previously described and the OSREQ STOREBEG request as described later.

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

The following section describes existing collection processing for an OSREQ Store type request that includes an OSREQ STORE request as previously described and the OSREQ STOREBEG request as described later.

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:

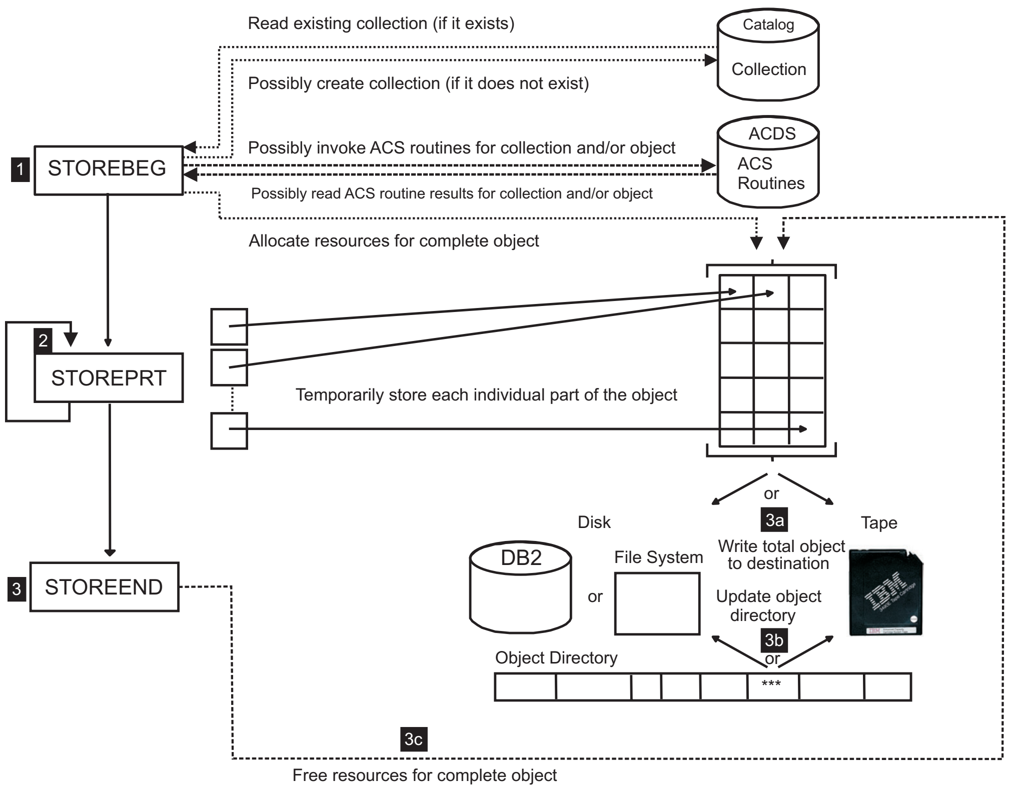
- If the management class and storage class are not specified on the OSREQ Store request, the default assignments contained in the MVS catalog entry for the collection are used as the assignments for the object.
- If management class and storage class are specified in the OSREQ Store request, the names 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.

STOREBEG—Beginning a Store Sequence operation

A Store Sequence operation begins with STOREBEG, which provides much of the same information that is provided on a STORE. See the description of OSREQ STORE for the description of keyword parameters. For STOREBEG, no buffers with object data are provided and therefore no keyword parameters related to these buffers are allowed. A store token (STOKEN) is provided as an output so an area to return this new store token must be provided. This store token must be provided on the subsequent STOREPRT and STOREEND functions. The size specified on STOREBEG is the total object size, which is required for OAM to acquire resources necessary to store the complete object. STOREBEG, STOREPRT and STOREEND cannot be used for objects less than or equal to 50 megabytes, nor can they be used for optical volumes.. Every STOREBEG request must have a corresponding STOREEND request.

Also see “Processing a store to a new collection” on page 24 and “Processing a store to an existing collection.” Note that during a store sequence, collection related processing is only performed for the OSREQ STOREBEG request and there is no additional interaction with the MVS catalog or ACS routines during OSREQ

STOREPRT or OSREQ STOREEND requests.



Read existing collection (if it exists)
Catalog
Collection
Possibly create collection (if it does not exist)
ACDS
Possibly invoke ACS routines for collection and/or object
ACS Routines
1
STOREBEG
Possibly read ACS routine results for collection and/or object
Allocate resources for complete object
2
STOREPRT
Temporarily store each individual part of the object
or
3a
Disk
Tape
DB2
File System
Write total object to destination
3
STOREEND
or
Update object directory
3b
Object Directory
or
***
3c
Free resources for complete object

Figure 2. Conceptual view of a Store Sequence operation

Syntax for OSREQ STOREBEG

```
►►──OSREQ STOREBEG──MF=──┬─L──────────────────────────────────────────►
                         ├─(M,parameter_list─┬───────────┬─)─┤
                         │                   └─,COMPLETE─┘   │
                         └─(E,parameter_list─┬───────────┬─)─┘
                                             └─,COMPLETE─┘

►─┬──────────────────────────────────────────┬────────────────────────►
  │      (1)                                 │
  └─TOKEN─────=─┬─token_area───────────┬─────┘
                └─(token_area_pointer)─┘

►─┬───────────────────────────────────────────┬───────────────────────►
  │       (1)                                 │
  └─STOKEN─────=─┬─store_token───────────┬────┘
                 └─(store_token_pointer)─┘

►─┬──────────────────────────────────────────────────────┬────────────►
  │         (1)                                          │
  └─COLLECTN─────=─┬─collection_name_area───────────┬────┘
                   └─(collection_name_area_pointer)─┘
```

```
►─┬─────────────────────────────────────────────────┬───────────────────────────────────────────►
  │      (1)                                         │
  └─NAME──────=─┬─object_name_area───────────────┬──┘
                └─(object_name_area_pointer)─────┘

►─┬─────────────────────────────────────────────────┬───────────────────────────────────────────►
  │      (1) (2)                                     │
  └─SIZE──────────=─┬─object_byte────────────┬─────┘
                    └─(object_byte_pointer)──┘

►─┬───────────────────────────────────────────────────┬─────────────────────────────────────────►
  └─STORCLAS=─┬─storage_class_area───────────────┬──┘
              └─(storage_class_area_pointer)─────┘

►─┬────────────────────────────────────────────────────────┬────────────────────────────────────►
  └─MGMTCLAS=─┬─management_class_area────────────────┬───┘
              └─(management_class_area_pointer)──────┘

►─┬──────────────────────────────────────────────┬──────────────────────────────────────────────►
  └─RETPD=─┬─retention_period─────────────────┬─┘
           └─(retention_period_pointer)───────┘

►─┬──────────────────────────────────────────┬──────────────────────────────────────────────────►
  └─MSGAREA=─┬─message_area────────────────┬─┘
             └─(message_area_pointer)──────┘

►─┬────────────────────────────────────────┬────────────────────────────────────────────────────►
  └─RETCODE=─┬─return_code────────────────┬┘
             └─(return_code_pointer)──────┘

►─┬────────────────────────────────────────┬────────────────────────────────────────────────────►
  └─REACODE=─┬─reason_code────────────────┬┘
             └─(reason_code_pointer)──────┘

►─┬──────────────────────────────────────────┬──────────────────────────────────────────────────►
  └─TTOKEN=─┬─tracking_token────────────────┬┘
            └─(tracking_token_pointer)──────┘

►─┬────────────────────────────────────────┬──┬───────────────────────────────────┬────────────►◄
  └─STIMEOUT=─┬─stimeout──────────────────┬┘  │              (3)                  │
              └─(stimeout_pointer)────────┘   └─DELHOLD──────=─┬─HOLD──────────┬─┘
                                                               └─(NOHOLD)──────┘
```

Notes:

1 These keywords are required and therefore they must be specified on the MF=E form if it indicates COMPLETE or they must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

2 The size specified must be the exact total size of the object.

3 If DELHOLD is not specified, the default value is DELHOLD=NOHOLD.

STOREPRT—Storing an individual part in a Store Sequence operation

Use one or more STOREPRT requests to store each individual part of the object following the prerequisite STOREBEG. For each STOREPRT, you must provide the store token that OAM uses to obtain information about this particular store request initiated with STOREBEG. You must specify the OFFSET where this part of the object is to be stored; for the first STOREPRT this offset must be 0 and for each subsequent STOREPRT, this offset must be the next byte following the previously stored part. Each part of the object therefore must be stored contiguously, in order, with no overlapping from beginning to end. The SIZE specified on STOREPRT indicates the size of the part of the object that is being stored. Note that this part of the object should be contained in either a single buffer or multiple contiguous buffers. It is suggested that the object be stored in as few parts as possible, because of the overhead involved in individually storing each part of the object. The minimum size for each part is 1 megabyte (1,048,576), except for the last part of the object. STOREBEG, STOREPRT and STOREEND cannot be used for objects with a total size less than or equal to 50 megabytes.

Syntax for OSREQ STOREPRT

```
►►──OSREQ STOREPRT──MF=──┬─L─────────────────────────────────┬──────────────────────────►
                         ├─(M,parameter_list──┬───────────┬─)─┤
                         │                    └─,COMPLETE─┘   │
                         └─(E,parameter_list──┬───────────┬─)─┘
                                              └─,COMPLETE─┘

►──┬──────────────────────────────────────────┬──────────────────────────────────────────►
   │       (1)                                │
   └─TOKEN───────=──┬─token_area───────────┬──┘
                    └─(token_area_pointer)─┘

►──┬──────────────────────────────────────────────┬──────────────────────────────────────►
   │        (1)                                   │
   └─STOKEN───────=──┬─store_token───────────┬────┘
                     └─(store_token_pointer)─┘

►──┬──────────────────────────────────────────────────┬──────────────────────────────────►
   │      (1) (2)                                     │
   └─SIZE───────────=──┬─object_byte───────────┬──────┘
                       └─(object_byte_pointer)─┘

►──┬──────────────────────────────────────────────────────────────────────┬──────────────►
   │        (1) (3)                                                       │
   └─OFFSET───────────=──┬─offset_of_starting_byte───────────┬────────────┘
                         └─(offset_of_starting_byte_pointer)─┘

►──┬────────────────────────────────────────────────────┬──┬─────────────────────┬───────►
   │         (1) (4)                                    │  └─RELBUF=──┬─YES─┬────┘
   └─BUFLIST───────────=──┬─buffer_list───────────┬─────┘             └─NO──┘
                          └─(buffer_list_pointer)─┘

►──┬──────────────────────────────────────────┬──────────────────────────────────────────►
   └─MSGAREA=──┬─message_area───────────┬─────┘
               └─(message_area_pointer)─┘

►──┬─────────────────────────────────────────┬───────────────────────────────────────────►
   └─RETCODE=──┬─return_code───────────┬─────┘
               └─(return_code_pointer)─┘

►──┬─────────────────────────────────────────┬───────────────────────────────────────────►
   └─REACODE=──┬─reason_code───────────┬─────┘
               └─(reason_code_pointer)─┘
```

```
►──┬───────────────────────────────────────────┬──────────────────────────────►◄
   └─TTOKEN=──┬─tracking_token──────────────┬──┘
              └─(tracking_token_pointer)────┘
```

Notes:

1. These keywords are required and therefore they must be specified on the MF=E form if it indicates COMPLETE or they must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.
2. The size specified must be the size of just this part of the object being stored.
3. The offset must be zero for the first part stored for the object, and for each subsequent store you must identify the offset of the next byte immediately following the previous part stored for the object (that is, the sum of the offset and size for the previous part stored).
4. The buffers provided must be contiguous and it is recommended that the amount of object data provided on each STOREPRT is maximized to minimize the number of individual STOREPRT requests.

STOREEND—Ending a Store Sequence operation

The STOREEND request follows a prerequisite STOREBEG request and typically one or more STOREPRT requests, and is required to complete the storage of the object. Every STOREBEG request must have a corresponding STOREEND request. For STOREEND, you must provide the store token that OAM uses to obtain information about this particular store request that was initiated with STOREBEG. The SIZE specified on STOREEND confirms the total size of the object to be stored, and is compared with the total object size specified on STOREBEG and with the object data that OAM has received with previous STOREPRT requests. The sum of the sizes of all parts stored with STOREPRT must equal the total storage size specified on STOREBEG. The SIZE keyword is ignored if CANCEL=YES is supplied. STOREBEG, STOREPRT and STOREEND cannot be used for objects less than or equal to 50 megabytes.

Syntax for OSREQ STOREEND

```
►►──OSREQ STOREEND──MF=──┬─L─────────────────────────────┬──────────────────────►
                         ├─(M,parameter_list──┬───────────┬─)─┤
                         │                    └─,COMPLETE─┘   │
                         └─(E,parameter_list──┬───────────┬─)─┘
                                              └─,COMPLETE─┘

►──┬──────────────────────────────────────────┬─────────────────────────────────►
   │        (1)                               │
   └─TOKEN──────=──┬─token_area─────────────┬─┘
                   └─(token_area_pointer)───┘

►──┬───────────────────────────────────────────┬────────────────────────────────►
   │         (1)                               │
   └─STOKEN──────=──┬─store_token────────────┬─┘
                    └─(store_token_pointer)──┘

►──┬────────────────────────────────────────────┬──┬──────────────────────┬─────►
   │       (1) (2)                              │  └─CANCEL=──┬─YES─┬─────┘
   └─SIZE──────────=──┬─object_byte───────────┬─┘             └─NO──┘
                      └─(object_byte_pointer)─┘
```

```
►─┬──────────────────────────────────────┬──────────────────────────────────►
  └─MSGAREA=──┬─message_area───────────┬─┘
              └─(message_area_pointer)─┘

►─┬────────────────────────────────────┬────────────────────────────────────►
  └─RETCODE=──┬─return_code──────────┬─┘
              └─(return_code_pointer)─┘

►─┬─────────────────────────────────────────────────┬───────────────────────►
  │            (3)                                  │
  └─RETCODE2─────────=──┬─return_code2───────────┬──┘
                        └─(return_code2_pointer)─┘

►─┬────────────────────────────────────┬────────────────────────────────────►
  └─REACODE=──┬─reason_code──────────┬─┘
              └─(reason_code_pointer)─┘

►─┬──────────────────────────────────────┬─────────────────────────────────►◄
  └─TTOKEN=──┬─tracking_token───────────┬─┘
             └─(tracking_token_pointer)─┘
```

Notes:

1 These keywords are required and therefore they must be specified on the MF=E form if it indicates COMPLETE, or they must be specified on at least one of the forms if the MF=E does not indicate COMPLETE.

2 The size specifies the total size of the object to be stored. Note that when specifying the total size of the object, it must match the total size specified on the STOREBEG and that exactly this amount of object data must have been previously provided with one or more STOREPRT requests for the object to be stored successfully.

3 If the immediate backup is configured with an optical target, the RETCODE2 keyword will return a value of 16 to indicate the immediate backup copy to optical is not supported for STOREEND in this release.

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

```
►►──OSREQ UNACCESS──MF=──┬─L─────────────────────────────────┬──────────────►
                         ├─(M,parameter_list─┬───────────┬─)─┤
                         │                   └─,COMPLETE─┘   │
                         └─(E,parameter_list─┬───────────┬─)─┘
                                             └─,COMPLETE─┘
```

```
►─┬──────────────────────────────────────┬──────────────────────────────►
  │       (1)                            │
  └─TOKEN─────=─┬─token_area───────────┬─┘
                └─(token_area_pointer)─┘

►─┬─────────────────────────────────────┬───────────────────────────────►
  └─MSGAREA=─┬─message_area───────────┬─┘
             └─(message_area_pointer)─┘

►─┬──────────────────────────────────┬──────────────────────────────────►
  └─RETCODE=─┬─return_code─────────┬─┘
             └─(return_code_pointer)─┘

►─┬──────────────────────────────────┬──────────────────────────────────►
  └─REACODE=─┬─reason_code─────────┬─┘
             └─(reason_code_pointer)─┘

►─┬────────────────────────────────────┬───────────────────────────────►◄
  └─TTOKEN=─┬─tracking_token─────────┬─┘
            └─(tracking_token_pointer)─┘
```

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 47.

CANCEL=YES

CANCEL=NO

The CANCEL keyword is used only on a STOREEND request to indicate if the storage of the object in a store sequence (using functions STOREBEG and STOREPRT) should be cancelled. CANCEL=NO indicates that this is a normal end of a store sequence and that the object should be stored to OAM. CANCEL=YES indicates that the store sequence should be cancelled, in which case the object is not stored to OAM and any resources held on behalf of the store sequence are then freed. CANCEL=NO is the default.

Please note that the SIZE keyword is ignored on STOREEND requests where CANCEL=YES.

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).

DELHOLD=NOHOLD

DELHOLD=HOLD

The DELHOLD parameter indicates whether or not a deletion-hold should be put on this object. An object cannot be deleted (either by an OSREQ DELETE request or by OSMC expiration processing) if it has a deletion-hold in effect. The DELHOLD keyword is only valid on CHANGE, STORE and STOREBEG requests and is ignored on all other requests. DELHOLD=NOHOLD is the default if DELHOLD is not specified on a STORE or STOREEND request. However, there is no default if DELHOLD is not specified on a CHANGE request.

DELHOLD=HOLD indicates that a deletion-hold is in effect for this object.

DELHOLD=NOHOLD indicates that there is not a deletion-hold in effect for this object.

Note: A **DELHOLD=HOLD** request for an object that is already in deletion-hold mode is ignored. Similarly, a **DELHOLD=NOHOLD** request for an object that is not in deletion-hold mode is also ignored.

EVENTEXP=*number_of_days*

EVENTEXP=(*number_of_days_pointer***)**

The EVENTEXP parameter provides a mechanism for the application to inform OAM that an external event has occurred for an object currently in event-based-retention mode. Receipt of the EVENTEXP parameter on the OSREQ CHANGE request starts the clock for expiration processing for this object, and takes the object out of event-based-retention mode. OAM sets the object's expiration date as follows.

If specified, *number_of_days* must be a four byte area containing a value in the range of 0 to 93 000.

The expiration date (ODEXPDT) is set to the earlier of the following two dates:

1. the creation date of the object plus the object's management class retention limit
2. today's date + the EVENTEXP value.

For retention-protected objects:

- ODRETDT is set to whichever is later; the newly calculated ODEXPDT or the current ODRETDT.
- ODEXPDT is set to whichever is later; the ODRETDT or the ODEXPDT.

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. 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*

LENGTH=(*number_bytes_pointer***)**

number_bytes specifies a four byte area 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 greater than the remaining portion of the object, or greater than 268,435,456 bytes, 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, 120 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.

Note: Applications that obtain storage explicitly for the OSREQ parameter list, rather than using the list macro form (MF=L) of the OSREQ macro, must ensure that they obtain a minimum of 120 bytes. Applications that use the list form (MF=L) will automatically acquire the 120 byte parameter list in a modifiable area of storage in the caller's key.

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. Within the message area, information is grouped into 72–byte lines. When displaying the information in the message area, breaking it into 72–byte segments and displaying one segment per output line will provide the best readability.

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 of the following wildcard methods can be used to request a generic search:

1. Legacy asterisk wildcard
   - 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.

     Note: Matching objects will be excluded if an additional qualifier to the right of the asterisk exists. For example, for objects name A.B and A.B.C, a query using A.* would return only A.B, not A.B.C.
2. New percent and underscore wildcards
   - One or more percent signs (X''6C') can be inserted anywhere in the object name. The percent sign is interpreted as a wildcard to replace zero or more characters in the object name.
   - One or more underscores (X'6D') can be inserted anywhere in the object name. The underscore is interpreted as a wildcard to replace a single character in the object name. The percent/underscore style wildcard uses the DB2 “LIKE” predicate as described in the DB2 SQL reference. Unlike the asterisk style, no exclusion will be done for objects having qualifiers to the right of the wildcard character. For example, for objects A.B and A.B.C, a query using A.% will return both objects.

OFFSET=*offset_of_starting_byte*

OFFSET=(*offset_of_starting_byte_pointer*)

The OFFSET keyword is only used by a RETRIEVE request or a STOREPRT request and is ignored on all other requests.

For a RETRIEVE request, *offset_of_starting_byte* is a four byte area 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. 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.

For a STOREPRT request, *offset_of_starting_byte* is a four byte area that specifies the offset of the first byte where the next part of the object is to be stored. For storing the first part of the object, the offset must be zero; for subsequent parts of the object, the offset is the next byte immediately following the previous part stored for the object (that is, the sum of the offset and size for the previous part stored).

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 49.

REACODE=*reason_code*

REACODE=(*reason_code_pointer*)

reason_code specifies an optional four byte 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* is not set, such as the *reason_code* area is invalid or a major error occurs before the *reason_code* area has been validated. The reason code value is always returned to register 0.

RECALL=*number_days*

RECALL=(*number_days_pointer*)

The RECALL keyword specifies that a temporary copy of the object being retrieved is to be written to disk sublevel 1 (DB2) or disk sublevel 2 (file system) and retained there for the specified number of days. This keyword is an optional parameter used only on a RETRIEVE request and ignored on all other requests.

number_days is a four byte area that specifies how many days a recalled object is to remain on disk sublevel 1 or 2 before OSMC transitions it back to its original location. The valid number of days that can be specified is 0 to 255. An invalid value for *number_days* results in the RETRIEVE request failing.

Note:
1. Regardless of whether the RETRIEVE request is for a full object or for a partial object, the RECALL keyword always results in a copy of the full object being written to disk sublevel 1 or 2.
2. The RECALL keyword is required on the OSREQ RETRIEVE request to initiate an explicit recall, however, implicit recalls can be activated by the SETOSMC statement in the CBROAM*xx* parmlib member.
3. The MAXRECALLTASKS must be set to a non-zero value in a SETOSMC statement in the CBROAM*xx* parmlib member to enable explicit or implicit recalls.
4. See *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support* for more information on explicit and implicit recalls.

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*

RETCODE=(*return_code_pointer*)

return_code is a four byte 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* is not set, such as the *return_code* area is invalid or a major error occurs before the *return_code* area has been validated. The return code value will always be returned to register 15.

RETCODE2=*return_code2*

RETCODE2=(*return_code2_pointer*)

RETCODE2 is an optional keyword that can be used to determine if OAM scheduled additional processing for this OSREQ request. *return_code2* is a four byte area into which the return code value is copied. The information returned in *return_code2* depends on the OSREQ function (RETRIEVE, STORE, or STOREEND) requested.

For an OSREQ RETRIEVE request, RETCODE2 specifies whether this RETRIEVE request resulted in scheduling a RECALL of the object to disk sublevel 1 or 2. **RETCODE2** is valid only when the RETRIEVE is successful, in which case it provides the following information:

RETCODE2	Meaning
0	Either RECALL not specified with RETRIEVE; no attempt to schedule RECALL or RECALL specified with RETRIEVE and successfully scheduled
4	RECALL not specified with RETRIEVE, but RECALL successfully scheduled owing to CBROAM*xx* parmlib member specifications
8	An attempt to schedule a RECALL was not successful because OSMC=NO was specified on OAM started procedure
10	An attempt to schedule a RECALL was not successful because MAXRECALLTASKS(0) was specified in the CBROAM*xx* parmlib member
12	An attempt to schedule a RECALL was not successful because RECALLOFF(ON) was specified in the CBROAM*xx* parmlib member
14	An attempt to schedule a RECALL was not successful because of a scheduling error

RETCODE2	Meaning
16	An attempt to schedule a RECALL was not successful because the RETRIEVE was performed on a downlevel OAMplex member that does not support RECALL processing

For an OSREQ STORE or STOREEND request, RETCODE2 specifies whether this STORE or STOREEND request resulted in scheduling an immediate backup copy to be written for this object.

return_code2 is valid only when the STORE or STOREEND is successful, in which case it provides the following information:

RETCODE2	Meaning
0	Immediate backup copy request successfully scheduled.
4	Immediate backup copy request not required.
8	An attempt to schedule an immediate backup for this object was not successful because OSMC is not up and running.
14	An attempt to schedule an immediate backup for this object was not successful due to unexpected scheduling error.
16	Immediate backup to optical not supported for STOREEND.

RETPD=*retention_period*

RETPD=(*retention_period_pointer*)

retention_period specifies a four byte area or an expression that contains the override retention period. See Table 2 on page 45 for valid retention periods.

SIZE=*object_byte*

SIZE=(*object_byte_pointer*)

The SIZE keyword is used on STORE, STOREBEG, STOREPRT, and STOREEND requests.

For STORE and STOREBEG requests, *object_byte* specifies a four byte area that contains the total object length in bytes.

The MOS=*nnnn* parameter in the IEFSSN*xx* parmlib member defines the maximum object size that can be stored. The maximum size is 50 megabytes (52,428,800 bytes) unless a larger maximum object size up to 2000 megabytes (2,097,152,000 bytes), has been defined. Refer to MOS=*nnnn* parameter in the IEFSSN*xx* parmlib member for more information on object sizes greater than 50 megabytes. Once this maximum object size has been defined, the length of the object determines which OSREQ function can be used to store the object.

For STORE requests, *object_byte* specifies a four byte area that contains the length in bytes of the object to be stored. STORE requests can be used for objects with a length up to 256 megabytes (268,435,456 bytes).

For STOREBEG requests, *object_byte* specifies a four byte area that contains the total length in bytes of the object to be stored. STOREBEG requests can be used only for objects with a total length greater than 50 megabytes (52,428,800 bytes).

For STOREPRT requests, *object_byte* specifies a four byte area that contains the length in bytes of the part of the object to be stored. The minimum length allowed on a STOREPRT is 1 megabyte (1,048,576 bytes). Only the last STOREPRT in the store sequence may specify a length less than 1 megabyte.

For STOREEND requests, *object_byte* specifies a four byte area that contains the total object length in bytes to complete storage of the object. The length specified must match the total object length in bytes specified on the STOREBEG request and that exactly this amount of object data must have been previously provided with one or more STOREPRT requests for the object to be stored successfully.

Note: When CANCEL=YES is specified, the SIZE keyword is ignored.

STIMEOUT=*stimeout*

STIMEOUT=(*stimeout_pointer*)

The STIMEOUT keyword is only used by a STOREBEG request and is ignored on all other requests.

The *stimeout* is a four byte area that specifies the maximum interval in seconds between STOREBEG, STOREPRT, and STOREEND requests that OAM should wait before OAM will assume that there will be no more activity for this store sequence and will free resources held on behalf of this store sequence. OAM will normally attempt to detect cases when there has been no activity from the application during a store sequence in progress and free limited resources that are being held on behalf of the application. This can occur if the application abnormally ends or encounters an error or otherwise does not normally complete the individual function calls in a store sequence. Specify a value if there will be an unusually long delay between the requests in a store sequence to ensure that OAM does not free resources used for the store sequence.

Note: This interval does not apply to the disk sublevel 1 of the OAM storage hierarchy.

Valid values for the number of seconds that can be specified are 0–9999. If the STIMEOUT keyword is not specified (or if the STIMEOUT value is specified as zero), then the STIMEOUT value defaults to 300 seconds (5 minutes).

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.

STOKEN=*stoken_area*

STOKEN=(*stoken_area_pointer*)

stoken_area specifies a 16-byte area on a double word boundary into which OSREQ STOREBEG stores a value. *stoken_area* must be specified on subsequent STOREPRT and STOREEND requests. The token becomes invalid after OSREQ STOREEND 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 75); 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), or when running under CICS with z/OS V1R12 OAM or after (where running under multiple CICS TCBs is supported), 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:

  1. All processing must be done under the same TCB that issued the ACCESS. The token cannot be used by more than one task.
  2. With z/OS V1R12 OAM and after, when running under CICS, this restriction no longer applies. A CICS OAM application program may perform OSREQ ACCESS and then other OSREQ calls under different CICS TCBs.
- 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 (by 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 IBM Information Management Software for z/OS Solutions Information Center.
- 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 IBM Information Management Software for z/OS Solutions Information Center). In the TSO environment, synchronization is accomplished through the use of COMMIT and ROLLBACK functions, as described in the IBM Information Management Software for z/OS Solutions Information Center. In the IMS environment, synchronization is accomplished through the use of COMMIT and ROLLBACK functions (see the IBM Information Management Software for z/OS Solutions Information Center), 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.

Note: If an application invokes the OSREQ API without passing an IADDRESS, OAM assumes the application is running in one of the CAF supported environments, Batch, IMS, CICS, TSO, or DSN. If an application invokes the OSREQ API using the IADDRESS parameter, it will be assumed that the application has done the connection to DB2 and has loaded the appropriate DB2 module. Environments or invocations other than those listed in Table 1 on page 13 in “ACCESS—Initializing the OSREQ interface” on page 12 have not been tested by IBM and the results may be unpredictable. An example of an untested, unpredictable environment would be the DB2 Stored procedure environment.

Restrictions and limitations

OAM supports a maximum object size of 50 megabytes (52 428 800 bytes) unless a larger maximum object size, up to 2000 megabytes (2 097 152 000 bytes), has been defined using the MOS=*nnnn* parameter in the IEFSSN*xx* parmlib member. Refer to *z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object*

Support for more information on using the MOS=*nnnn* keyword to specify a maximum object size greater than 50 megabytes.

Note:

1. When storing an object greater than 50 MB, if multiple data buffers are supplied, however the data buffers are not in contiguous storage, the request fails with OSREQ return/reason code: Return Code=08 , Reason Code=2402080A or 2409080A
2. When retrieving an object greater than 50 MB, if the first data buffer supplied is not large enough to contain the requested or partial object, the request fails with OSREQ return/reason code: Return Code=08, Reason Code = 2403080B

These buffer restrictions ensure that extra GETMAINs are not made in the user's (applications) address space. 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 31.
- If you do not specify a collection name on any function other than ACCESS or UNACCESS, STOREPRT, or STOREEND 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, STOREBEG, 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 or STOREBEG 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, STOREEND, DELETE, CHANGE, and RETRIEVE as soon as possible (to minimize DB2 timeouts or deadlocks), depending on return code. The synchronization must occur within 24 hours for objects stored in the file system (to avoid loss of data).
- 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 CBRABIND (or CBRIBIND for DASD-only users) is used to

create the CBRIDBS plan in DB2. For more information on the CBRABIND, CBRIBIND jobs, 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, the event expiration time period (EVENTEXP) specified on OSREQ CHANGE, or on the object's management class expiration rules. 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 expiration date in the object's directory entry is set to the reserved value of '0002–02–02' when the object is in event-based-retention mode (as a result of RETPD being set to -2 (X'FFFFFFFE') on an OSREQ STORE, STOREBEG, or CHANGE). In this case, the object has an indefinite expiration date which will be set at some point in the future when a particular event has occurred (which is indicated by an OSREQ CHANGE with the EVENTEXP keyword). 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.

Table 2 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 93 000 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.

Table 2. Valid Retention Periods for Expiration Date Processing

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.
X'FFFFFFFF' (-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.
X'FFFFFFFE' (-2)	Set expiration date to 0002–02–02 and set indicator in ODSTATF to show this object is in event-based-retention mode. The expiration date for the object is then based on notification of an external event as specified by the OSREQ CHANGE EVENTEXP=*number_of_days*.	Set expiration date to 0002–02–02 and set indicator in ODSTATF to show this object is in event-based-retention mode. The expiration date for the object is then based on notification of an external event as specified by the OSREQ CHANGE EVENTEXP=*number_of_days*.
1 to 93 000	If the RETPD value specified is greater than the object's management class retention limit, the expiration date (ODEXPDT) is set to the creation date of the object plus the object's management class retention limit. Otherwise, the ODEXPDT is set to sum of the object create date + RETPD value. For retention-protected objects: • the ODRETDT is set to whichever is later; the newly calculated ODEXPDT or the current ODRETDT. • the ODEXPDT is set to whichever is later; the ODRETDT or the ODEXPDT.	If the RETPD value specified is greater than the object's management class retention limit, the expiration date (ODEXPDT) is set to the creation date of the object plus the object's management class retention limit. Otherwise, the ODEXPDT is set to sum of the object create date + RETPD value. For retention-protected objects: • the ODRETDT is set to whichever is later; the newly calculated ODEXPDT or the current ODRETDT. • the ODEXPDT is set to whichever is later; the ODRETDT or the ODEXPDT.
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.

Note: If the current expiration date is '0002–02–02' (which means the object is in event-based-retention-mode), the expiration date cannot be changed with the RETPD keyword. Any attempt to do so results in the OSREQ CHANGE failing. The only way to change the expiration date for an object in event-based-retention mode is by specifying the EVENTEXP keyword on an OSREQ CHANGE.

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.

OAM 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 75 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.

DB2 SQL error reason codes

When a DB2 error is encountered, OAM issues messages that display DB2 SQL error reason codes. For a selected subset of these SQL codes, OAM also issues additional messages to explain the SQL codes to save the operator and storage administrator the trouble of having to look up the codes in the DB2 information. The DB2 SQL codes and the OAM messages that explain them are:

DB2 SQL code	OAM message
-204	CBR7540I
-205, -206	CBR7541I
-501	CBR7542I
-805	CBR7543I
-818	CBR7544I
-904	CBR7545I

See *z/OS MVS System Messages, Vol 4 (CBD-DMO)* for a description of these messages.

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 3 and Figure 4 on page 48 describe the contents of the DSECTs.

Figure 3. Fields Described by CBRIBUFL

```
    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 3. Fields Described by CBRIBUFL

Figure 4 on page 48 is a structure diagram of the data buffer list (CBRIBUFL) pointed to by the BUFLIST keyword on an OSREQ STORE or OSREQ RETRIEVE macro.

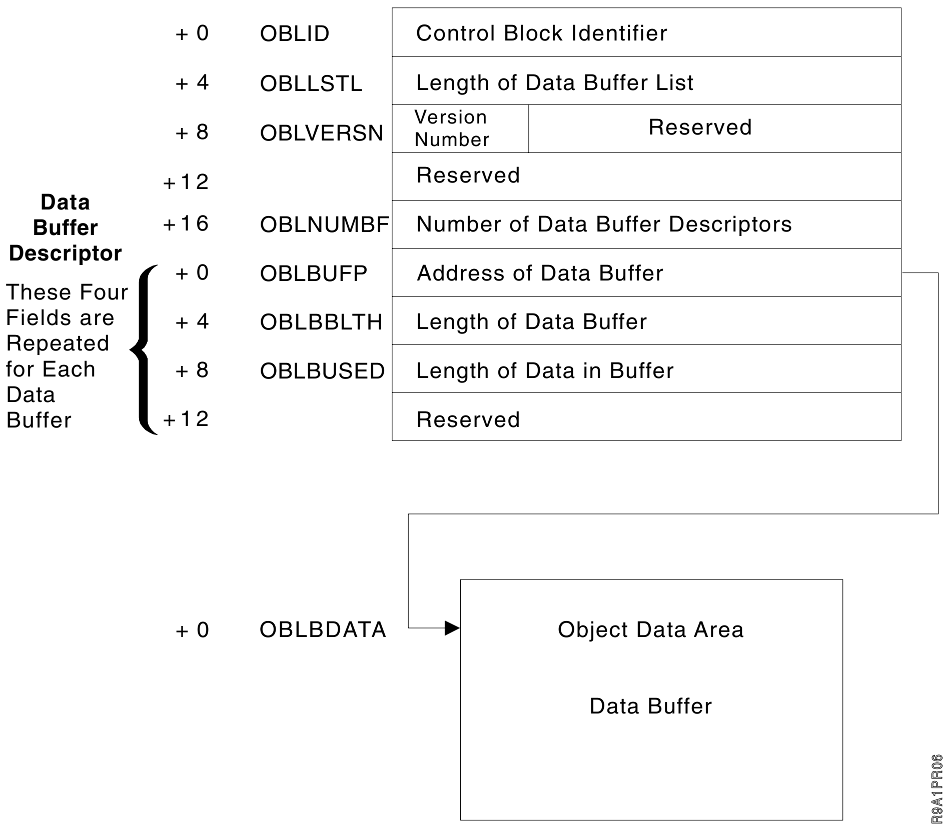


Figure 4. Data Buffer List Structure Diagram

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 5 on page 50 and Figure 6 on page 52 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.

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.

If the primary copy of the object is on disk, then the primary retrieval order key will contain binary zeros. Similarly, if a backup or secondary backup copy of the object does not exist, then the corresponding backup or second backup retrieval order key will contain binary zeros. Also, if the QB= keyword in the IEFSSN*xx* parmlib member is set to QB=N, then the OAM address space will not be invoked to obtain any existing backup retrieval order keys. This will result in the backup and second backup retrieval order keys containing binary zeros.

```
    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&colon;

```
    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&colon;

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

Each query element is described by the following&colon;

```
     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)
+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
+242 QELQEPD   DS   CL10           Pending action date (yyyy-mm-dd)
+252 QELQERD   DS   CL10           Retention date (yyyy-mm-dd)
+262 QELQESF   DS   XL2            Status flags
+264 QELQELF   DS   CL1            Location flag
+265 QELQEDP   DS   CL1            Deletion protection indicator
+266           DS   CL2            Reserved
```

Figure 5. 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, QELQB2OK, QELQEPD, QELQERD, QELQESF, QELQELF, and QELQEDR fields in the total length of the QUERY element.

- If QELVERSN>=6, then the query buffer (QELQ) contains the QELQPROK, QELQBROK, QELQB2OK, and QELLNARE fields. The backup retrieval order key fields contain binary zeroes if none of the backup copies exist.
- 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, QELQB2OK, QELQEPD, QELQERD, QELQESF, QELQELF, and QELQEDR) are 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 disk sublevel 1 (DB2).
9 000	Object resides on disk sublevel 2 (file system).
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.

The actual file system sublevel retrieval response time can vary significantly and depends on many factors, including the size of the object, whether the object resides in zFS or NFS, the underlying disk used for a zFS file system, the hardware device, configuration, and network implications for NFS, and the overall z/OS UNIX workload. The estimated retrieval response time therefore is intended to provide only a comparative response time relative to the other OAM storage hierarchy targets for objects.

If the retrieval response time cannot be determined, QELQERRT is set to the reserved value of -1 (X'FFFFFFFF').

Figure 6 is a structure diagram of the query buffer list (CBRIQEL) pointed to by the QEL keyword on an OSREQ QUERY macro:

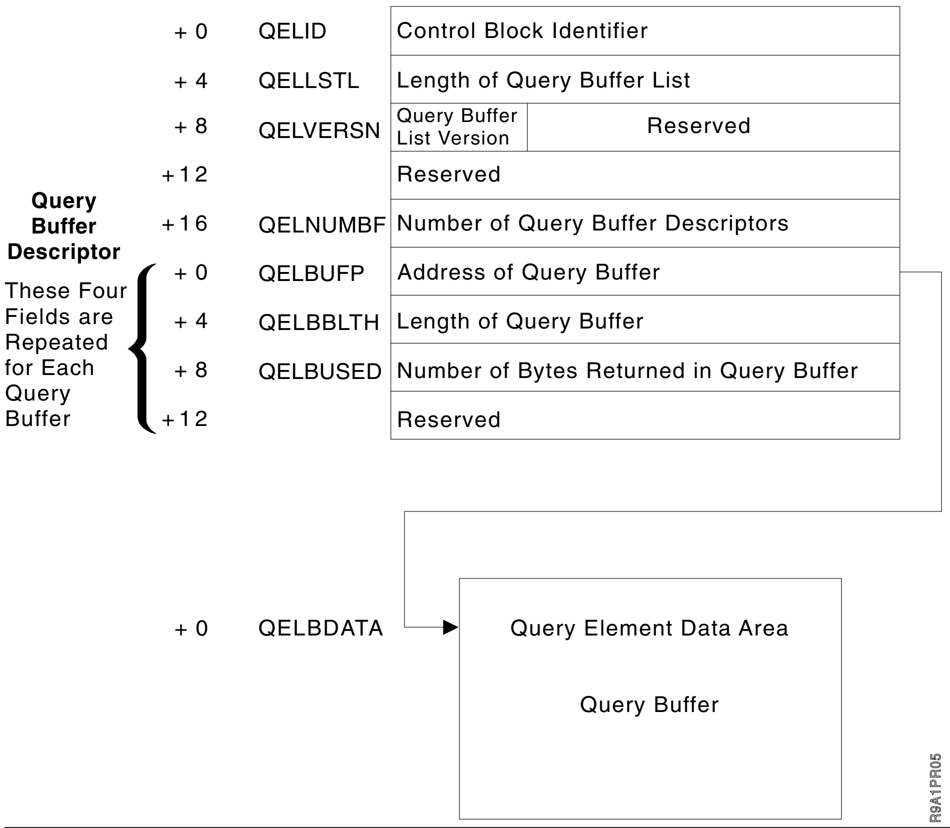


Figure 6. Query Buffer List Structure Diagram

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. An expiration date of “0002–02–02” indicates that this object is currently in event-based-retention mode, and that it is waiting on receipt of an EVENTEXP keyword on an OSREQ CHANGE request before calculating the object's 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* parmlib member 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 two sample programs that use the OSREQ macro for object manipulation. See “CBROSREQ” and “CBROSR2” on page 64 for these sample programs. These programs are available as members CBROSREQ or CBROSR2 in SAMPLIB.

There are two basic differences between the two samples:

- CBROSR2 supports the new store sequence OSREQ functions of STOREBEG, STOREPRT, and STOREEND. CBROSREQ does not support these new functions.
- CBROSREQ includes DB2 related functions such as CAF OPEN and CAF CLOSE as well the EXEC SQL statements COMMIT and ROLLBACK. CBROSR2 does not contain any DB2 related functions or EXEC SQL statements.

You can use these samples 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=ASMA90,PARM='RENT,DECK,SYSPARM(IADD)'
  ```
- The CBROSREQ sample uses the DSNHLI entry point for the OSREQ IADDRESS parameter. Whereas the CBROSR2 sample uses the IADDRESS_PTR field located in the caller supplied DATAAREA for the OSREQ IADDRESS parameter.
- You can link-edit members CBROSREQ or CBROSR2 as part of the application load module. You do not need to issue LOAD request before using the OSREQ calls.
- You can use members CBROSREQ or CBROSR2 without modification to support application programs written in PL/1 or COBOL.
- You can modify members CBROSREQ or CBROSR2 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 statement in the code for the CBROSREQ sample. Please note that the CBROSR2 sample contains no EXEC SQL statements, so the DB2 pre-compiler does not need to be run for it.

CBROSREQ

Sample program for an object storage request using the OSREQ macro:

```
***********************************************************************
*
* DESCRIPTIVE NAME: Object Storage Request Sample interface
*
* FUNCTION: Provides a generalized interface for the Object Storage
*           Request (OSREQ) macro.
*
*           This interface includes support to perform a DB2 CAF
*           sync (commit) or DB2 CAF abort (rollback).           @L1A
*
*           This interface does not support the following OSREQ
*           functions:  STOREBEG, STOREPRT, and STOREEND.
*           Please see sample job CBROSR2 for support of the
*           STOREBEG, STOREPRT, and STOREEND functions.          @L1A*
* 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 data bases 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 data bases.
*            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 data bases
*            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.
*
* DB2 SYNC and ROLLBACK Notes:                                   @L1A
*    This sample is setup to assume the MVS batch environment.
*    Changes related to executing the DB2 SYNC and ROLLBACK
*    functions will need to be made for other environments.
*    For example in a CICS environment, EXEC CICS SYNCPOINT
*    would need to be performed instead of calling DSNALI to do
*    a CAF CLOSE.                                                @L1A
*
*    If this sample is NOT compiled with the IADD SYSPARM or a
*    a DB2 connection is not already established, then
*    a DB2 connection or thread will be established by OAM
*    performing a CAF OPEN during the OSREQ ACCESS request.  If
*    SYNC in the DATAAREA equals "YES", then a CAF CLOSE is used to
*    perform either a DB2 sync or rollback.  At this point the
*    applications DB2 thread will be closed.  To reopen this
*    thread, this sample will perform a CAF OPEN.  The values of
*    the return and reason code for the CAF open is stored in the
*    fields pointed to by CAFOPEN_RC_PTR and CAFOPEN_RS_PTR.     @L1A
*
*    If this sample IS compiled with the IADD SYSPARM, then a DB2
*    connection and open thread is assumed and this sample will do
*    an SQL COMMIT and SQL ROLLBACK instead of a CAF CLOSE to perform
*    a DB2 sync or rollback.  The CAFOPEN_RC_PTR and CAFOPEN_RS_PTR
*    fields will not be set.                                     @L1A
**    If a DB2 sync or rollback is performed because the SYNC field
*    in the DATAAREA equals "YES", then the return and reason code
*    values of the commit or rollback will be stored in the fields
*    pointed to by CAFCLOSE_RC_PTR and CAFCLOSE_RS_PTR.  This sample
*    uses a CAF CLOSE with SYNCH or a CAF CLOSE with ABRT for the
*    MVS batch environment when the SYSPARM IADD is NOT specified
*    and an SQL COMMIT or SQL ROLLBACK when IADD is specified.    @L1A
*
*
* Valid values for FUNCTION_REQUEST:                             @L1A
*    "ACCESS  " : OSREQ ACCESS
*    "STORE   " : OSREQ STORE
*    "RETRIEVE" : OSREQ RETRIEVE
*    "QUERY   " : OSREQ QUERY
*    "CHANGE  " : OSREQ CHANGE
*    "DELETE  " : OSREQ DELETE
*    "UNACCESS" : OSREQ UNACCESS*
*
*
* IADDRESS NOTE:
* 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=ASMA90,PARM='RENT,DECK,SYSPARM(IADD)'  @L1C
*
* 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 restore 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.
* 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 for OSREQ function requests.            @L1C
*                These fields will contain the reason and return
*                codes for DB2 sync & rollback function requests.  @L1A
*         Fields pointed to by CAFOPEN_RC_PTR and CAFOPEN_RS_PTR
*                will contain the reason and return codes returned
*                from calling DSNALI to do a CAF OPEN.  See 'DB2
*                SYNC and ROLLBACK Notes' above for more info.     @L1A
*         Fields pointed to by CAFCLOSE_RC_PTR and CAFCLOSE_RS_PTR
*                will contain the reason and return codes returned
*                from calling DSNHLI to do a SQL COMMIT or ROLLBACK.
*                See 'DB2 SYNC and ROLLBACK Notes' above for more
*                information.                                      @L1A
*          Areas defined by the CBRIBUFL (for retrieve) and CBRIQEL
*               (for query) will be filled in when the respective
*               function is requested.
* CHANGE-ACTIVITY:
*    $L0=JDP3227 320 881229 TPCTGT: OAM Release 1
*    $01=OY29609 320 900219 TPCTGT: Remove unknown macro call
*    $P0=KBE0022 331 911216 TUCTNN: IADDRESS support
*    $02=OY59202 110 921111 TUCHAD: Save R15, R0 after OSREQ
*    $L1=OAM2GB  R1A 070316 TUCGPW: OAM2GB Phase 1
*    $P1=K1A2012 R1A 080109 TUCGPW: Fixed loading VIEW into register
*    $L2=OAMR1B  R1B 080409 TUCDVH: OAMARE Archive retention       @L2A
**********************************************************************
OSRSAMPL CSECT ,
OSRSAMPL AMODE 31
OSRSAMPL RMODE ANY
          USING *,R15                  USING to allow branch to STRTOSR
*
          SPACE 2
          B     STRTOSREQ              BRANCH TO ACTIVE PART OF MODULE
LENGOSR   DC    X'18'                  LENGTH OF HEADER INFORMATION
NAMEOSR   DC    CL8'CBROSREQ'          MODULE NAME FOR TRACING
DATEOSR   DC    CL8'&SYSDATE'          MODULE ASSEMBLY DATE
APAROSR   DC    CL8'HDZ1B10'          APAR LEVEL FOR THIS MODULE
          DROP  R15
          SPACE 2
STRTOSREQ DS    0H                     START THE ACTIVE PART OF MODULE
         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     R6,RETRIEVE_OFFSET_PTR
         L     R7,RETRIEVE_LENGTH_PTR
         L     R8,RETURN_CODE_PTR
         L     R9,REASON_CODE_PTR
     OSREQ (STORE),MF=(M,PARM_LIST),                                   X
               MSGAREA=(R0),          DB2 error messages returned here X
               TOKEN=TOKEN_AREA,      Contains logical OAM connection  X
               COLLECTN=COLLECTION_NAME,                               X
               NAME=OBJECT_NAME,                                       X
               SIZE=(R2),                                              X
               STORCLAS=(R3),                                          X
               MGMTCLAS=(R4),                                          X
               RETPD=(R5),                                             X
               OFFSET=(R6),           Starting byte for retrieve       X
               LENGTH=(R7),           Length of retrieve               X
               RETCODE=(R8),          Register 15 is stored here       X
               REACODE=(R9)           Register 0 is stored here
*
         L     R0,TRACKING_TOKEN_PTR                               @L1A
         L     R2,RETURN_CODE2_PTR                                 @L1A
         L     R3,RECALL_NUM_DAYS_PTR                              @L1A
     OSREQ (STORE),MF=(M,PARM_LIST),                               @L1AX
               VIEW=PRIMARY,          Retrieve Primary Copy        @L1AX
               TTOKEN=(R0),           User Tracking Token          @L1AX
               RETCODE2=(R2),         Return Code 2                @L1AX
               RECALL=(R3)            Recall Number of Days        @L1A
*
* if view=2, the set VIEW=BACKUP                                   @L1A
               SLR   R6,R6            Zero Register                @L1A
               L     R6,VIEW          Load view into R6            @P1C
               LA    R10,2            Load value 2 into R10        @L1A
               CR    R6,R10           Does view = 2?               @L1A
               BNE   TRYVIEW3         No, then see if view = 3     @L1A
     OSREQ (STORE),MF=(M,PARM_LIST),                               @L1AX
               VIEW=BACKUP            Retrieve First Backup Copy   @L1A
               B     TRYRELBUF        Skip test 'if view=3'        @L1A
*
* else if view=3, then set VIEW=BACKUP2                            @L1A
TRYVIEW3       DS   0H                                             @L1A
               LA    R10,3            Load value 3 into R10        @L1A
               CR    R6,R10           Does view = 3?               @L1A
               BNE   TRYRELBUF        Nope, so leave VIEW=PRIMARY  @L1A
     OSREQ (STORE),MF=(M,PARM_LIST),                               @L1AX
               VIEW=BACKUP2           Retrieve First Backup Copy   @L1A
*
TRYRELBUF      DS   0H                                             @L1A
               CLC RELEASE_BUFFER,=CL3'YES'
               BNE NORELBUF
     OSREQ (STORE),MF=(M,PARM_LIST),                                   X
```

```
               RELBUF=YES             Will release pages after STORE
NORELBUF       DS   0H
*                                                                @L2A
* Set RETPD or EVENTEXP or both, based on caller's parm list.    @L2A
*                                                                @L2A
* Note that a runtime error will occur if non-zero pointers are  @L2A
* present for both RETPD and EVENTEXP. Supplying both RETPD and  @L2A
* EVENTEXP is generally only useful for testing the error checking @L2A
* features of the OSREQ processing code.                         @L2A
*                                                                @L2A
         L     R5,RETENTION_PERIOD_PTR                           @L2A
         OSREQ (STORE),MF=(M,PARM_LIST),                         @L2AX
               RETPD=(R5)                                        @L2A
*                                                                @L2A
         L     R5,EVENTEXP_PTR                                   @L2A
         OSREQ (CHANGE),MF=(M,PARM_LIST),  EVENTEXP only on CHANGE @L2AX
               EVENTEXP=(R5)                                     @L2A
*                                                                @L2A
* Set the DELHOLD parm or leave it off.                          @L2A
*                                                                @L2A
DELHCHK  DS    0H                                                @L2A
         CLC   =C'HOLD',DELHOLD                                  @L2A
         BE    DELHYES                                           @L2A
*                                                                @L2A
         CLC   =C'NOHOLD',DELHOLD                                @L2A
         BE    DELHNO                                            @L2A
         B     DELHDONE                                          @L2A
*                                                                @L2A
DELHNO   DS    0H                                                @L2A
         OSREQ (STORE),MF=(M,PARM_LIST),                         @L2AX
               DELHOLD=NOHOLD                                    @L2A
         B     DELHDONE                                          @L2A
*                                                                @L2A
DELHYES  DS    0H                                                @L2A
         OSREQ (STORE),MF=(M,PARM_LIST),                         @L2AX
               DELHOLD=HOLD                                      @L2A
DELHDONE DS    0H                                                @L2A
               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)
* otherwise, the connection is done by the environment in which
* this program is executing.
* In all cases system control blocks will be created and/or modified
* to provide this access to OAM.
*
* 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. See NOTE in prolog.
*
      AIF ('&SYSPARM' EQ 'IADD').IA2
     OSREQ ACCESS,MF=(E,PARM_LIST)
         AGO   .SKIP1
.IA2     ANOP
* In this sample we use DSNHLI for SQL interface module to DB2
         L     R2,=V(DSNHLI)
     OSREQ ACCESS,MF=(E,PARM_LIST),                                X
               IADDRESS=(R2)          GET THE ADDRESS OF THE INTERFACE
.SKIP1   ANOP
*
* 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),                                    X
               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 data base 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),                                   X
               QEL=(R10)
               B     TRY_SYNC_POINT
TRY_RETRIEVE   DS    0H
               CLC FUNCTION_REQUEST,=CL8'RETRIEVE'
               BNE TRY_DELETE
*
* 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),                                 X
               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                                   @L1C
*
* 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.
* 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
* Save CAF return code
* Note: We already saved the rc for other functions (access,
*       store, etc), so don't want to overwrite that rc w/ the
*       commit/rollback rc                                    @L1A
SAVE_CAFRC     L     R8,CAFCLOSE_RC_PTR                       @L1A
               L     R9,CAFCLOSE_RS_PTR                       @L1A
               ST    R15,0(,R8)             Save CAFCLOSE RETCODE  @L1A
               ST    R0,0(,R9)              Save CAFCLOSE REASCODE @L1A
               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     SAVE_SQLCODES                            @L1C
SET_SYNC       EXEC  SQL COMMIT
* Save SQL return codes
* Note: We already saved the rc for other functions (access,
*       store, etc), so don't want to overwrite that rc w/ the
*       commit/rollback rc                                     @L1A
SAVE_SQLCODES  L     R8,CAFCLOSE_RC_PTR                        @L1A
               L     R9,CAFCLOSE_RS_PTR                        @L1A
               ST    R15,0(,R8)          Save SQL RETURN CODE   @L1A
               ST    R0,0(,R9)           Save SQL REASON CODE   @L1A
               AGO   .SKIP2                                    @L1C
.SKIP   ANOP
*
* 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
               L     R8,CAFOPEN_RC_PTR                         @L1A
               L     R9,CAFOPEN_RS_PTR                         @L1A
               ST    R15,0(,R8)           Save Return Code      @L1A
               ST    R0,0(,R9)            Save Reason Code      @L1A
.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         DB2 CAF MVS batch LI services
               ST    R0,WORK_AREA4     Save for future calls.
               LR    R15,R0            Return address of DSNALI
               BR    R10                to caller
*
* 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    Savearea for this module.
*******
* 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 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
VIEW                 DS F      Retrieve Object Copy               @L1A
*                              1 = PRIMARY                        @L1A
*                              2 = First BACKUP Copy              @L1A
*                              3 = Second BACKUP Copy             @L1A
TRACKING_TOKEN_PTR   DS A      User Tracking Token Pointer        @L1A
RECALL_NUM_DAYS_PTR  DS A      Recall Number of Days Pointer      @L1A
RETURN_CODE2_PTR     DS A      Return Code 2 Pointer              @L1A
CAFOPEN_RC_PTR       DS A      Pointer to the OPEN CAF return code@L1A
CAFOPEN_RS_PTR       DS A      Pointer to the OPEN CAF reason code@L1A
CAFCLOSE_RC_PTR      DS A      Pointer to the CLOS CAF return code@L1A
CAFCLOSE_RS_PTR      DS A      Pointer to the CLOS CAF reason code@L1A
*
* 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
DELHOLD              DS CL8    DELHOLD= HOLD | NOHOLD | blank     @L2A
EVENTEXP_PTR         DS A      Pointer to EVENTEXP value          @L2A
********
* 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
*                                                                 @O1D
         END   OSRSAMPL
```

CBROSR2

Sample Program for an Object Storage Request Using the OSREQ Macro

```
***********************************************************************
*
* DESCRIPTIVE NAME: Object Storage Request Sample interface #2
*
* 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.
*
*            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.
*
*            1. Validity check the DATAAREA Header.  Exit if error.
*            2. Fill in the OSREQ PARM_LIST with all of the optional
*               keywords using MF=M form of the macro.
*            3. If FUNCTION_REQUEST = "ACCESS  "                  @P1C
*               a. IF CBROSR2 was compiled with IADD option, then
*                  set IADDRESS OSREQ macro keyword to the address of
*                  the DB2 library entry point DSNHLI using the MF=M
*                  form of the macro.                               @P1C
*               b. ELSE set IADDRESS OSREQ macro keyword to
*                  IADDRESS_PTR using the MF=M form of the macro.  @P1A
*            4. SELECT FUNCTION_REQUEST
*               WHEN(ACCESS, STORE, RETRIEVE, QUERY, CHANGE, DELETE,
*                    UNACCESS, STOREBEG, STOREPRT, STOREEND)
*               a. Set any function specific keywords
*               b. Execute specified function using the MF=E form
*                  of the macro.
*               c. Set R15 to 0, to indicate successful OSREQ
*                  macro invocation
*               OTHERWISE:
*               a. Set R15 to Invalid Function Request
*            5. Return to caller
*
* Valid values for FUNCTION_REQUEST:
*    "ACCESS  " : OSREQ ACCESS
*    "STORE   " : OSREQ STORE
*    "RETRIEVE" : OSREQ RETRIEVE
*    "QUERY   " : OSREQ QUERY
*    "CHANGE  " : OSREQ CHANGE
*    "DELETE  " : OSREQ DELETE
*    "UNACCESS" : OSREQ UNACCESS
*    "STOREBEG" : OSREQ STOREBEG
*    "STOREPRT" : OSREQ STOREPRT
*    "STOREEND" : OSREQ STOREEND
*
*
* IADDRESS NOTE:
*   To specify the default DSNHLI entry point for the
*   IADDRESS keyword in the OSREQ function, specify
*   the SYSPARM value as IADD in the PARM field of
*   the EXEC JCL statement. For example:                            @P1C
*
*    //ASSEMBLE  EXEC PGM=ASMA90,PARM='RENT,DECK,SYSPARM(IADD)'
*
* REGISTER CONVENTIONS:
*    R0  - WORK REGISTER
*    R1  - STANDARD LINKAGE REGISTER
*        - PARAMETER LIST ADDRESS
*    R2  - WORK REGISTER
*    R3  - WORK REGISTER
*    R4  - WORK REGISTER
*    R5  - WORK REGISTER
*    R6  - WORK REGISTER
*    R7  - WORK REGISTER
*    R8  - WORK REGISTER
*    R9  - WORK REGISTER
*    R10 - WORK REGISTER
*    R11 - DATAAREA BASE REGISTER
*    R12 - OSR2SAMP 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
*
* 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 restore 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.
* OUTPUT: Register 15 will contain the return code from DATAAREA
*                     validity checking.
*                         CODE  MEANING
*                         0     SUCCESS--OSREQ Function invoked
*                         6     Invalid DATAAREA FUNCTION_REQUEST
*                         8     Invalid DATAAREA hdr ID
*                         10    Invalid DATAAREA hdr length
*                         12    Invalid DATAAREA hdr version
*                         14    Invalid DATAAREA hdr release
*
*          Fields pointed to by REASON_CODE_PTR and RETURN_CODE_PTR
*                 will contain the reason and return codes returned
*                 from OAM for OSREQ function requests.
*          Areas defined by the CBRIBUFL (for retrieve) and CBRIQEL
*                (for query) will be filled in when the respective
*                function is requested.
*
* CHANGE-ACTIVITY:
*    $L0=OAM2GB  R1A 070316 TUCGPW: OAM2GB Phase 1
*    $P0=K1A2012 R1A 080109 TUCGPW: Fixed loading VIEW into register
*    $P1=K1A2309 R1A 080228 TUCGPW: Clarify how and when we set
*                                   the IADDRESS OSREQ function
*                                   keyword.
*    $01=OA25764 R1A 080725 TUCGPW: Add backward compatibility
*    $L1=OAMR1B  R1B 080716 TUCDVH: OAMARE Archive retention       @L1A
*    $P2=K1B0132 R1B 080721 TUCDVH: STIMEOUT support               @P2A
*
**********************************************************************
OSR2SAMP  CSECT ,
OSR2SAMP  AMODE 31
OSR2SAMP  RMODE ANY
          USING *,R15                 USING to allow branch to STRTOSR2
*
          SPACE 2
          B     STRTOSR2              BRANCH TO ACTIVE PART OF MODULE
LENGOSR2  DC    X'18'                 LENGTH OF HEADER INFORMATION
NAMEOSR2  DC    CL8'CBROSR2 '         MODULE NAME FOR TRACING
DATEOSR2  DC    CL8'&SYSDATE'         MODULE ASSEMBLY DATE
APAROSR2  DC    CL8'HDZ1B10'          APAR LEVEL FOR THIS MODULE
          DROP  R15
          SPACE 2
STRTOSR2  DS    0H                    START THE ACTIVE PART OF MODULE
*
         STM   R14,R12,12(R13)
*
* Register 12 is the base for the code
*
         LR    R12,R15
         USING OSR2SAMP,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

*
* Do some DATAAREA Header Validity Checking
*

* Make sure the ID of the user's dataarea = current ORSSS ID
               LA    R15,ERR_ID       Load ERR_ID into R15
               CLC   DA_ID,=CL4'OSR2' Does DA_ID == ID
               BNE   EXIT             Exit if not equal

* Make sure the length of the user's dataarea = current ORSSS length
               LA    R15,ERR_LEN      Load ERR_LEN into R15
               L     R0,DA_LEN
               CFI   R0,DATAAREA_LEN  Does DA_LEN = LENGTH
               BNE   EXIT             Exit if not equal

* Make sure user's dataArea version is <= current OSR2 version     @01C
               LA    R15,ERR_VER      Load ERR_VER into R15
               SLR   R2,R2            Zero Register
               IC    R2,DA_VER        Load DA_VER into R2          @01C
               LA    R3,OSR2_VER      Load VERSION into R3
               CR    R3,R2            Does DA_VER = VERSION?       @P0C
               BL    EXIT             Exit w/ err if VERSION
                                      < DA_VER                     @01C

* Make sure user's dataArea release is <= current OSR2 release     @01C
               LA    R15,ERR_REL      Load ERR_REL into R15
               SLR   R2,R2            Zero Register
               IC    R2,DA_REL        Load DA_REL into R2          @01C
               LA    R3,OSR2_REL      Load RELEASE into R3
               CR    R3,R2            Does DA_REL = RELEASE?       @P0C
               BL    EXIT             Exit w/ err if RELEASE
                                      < DA_REL                     @01C

* Modify the parameter list to establish all the basic OSREQ function
* parameters.
*
* Note: A pointer with a value of zero is equivalent to an omitted parm
*
OSR_FUNC       DS    0H
         L     R0,COLLECTION_NAME_PTR
         L     R2,MANAGEMENT_CLASS_PTR
         L     R3,MESSAGE_AREA_PTR
         L     R4,OBJECT_NAME_PTR
         L     R5,OBJECT_SIZE_PTR
         L     R6,OFFSET_PTR
         L     R7,REASON_CODE_PTR
         L     R8,RECALL_NUM_DAYS_PTR
         L     R10,RETRIEVE_LENGTH_PTR
*
* Removed RETPD parm from this initial OSREQ invocation            @L1D
*
     OSREQ (STORE),MF=(M,PARM_LIST),                                   X
               TOKEN=TOKEN_AREA,      Contains logical OAM connection  X
               COLLECTN=(R0),                                          X
               MGMTCLAS=(R2),                                          X
               MSGAREA=(R3),          DB2 error messages returned here X
               NAME=(R4),                                              X
               SIZE=(R5),                                              X
               OFFSET=(R6),           Starting byte for retrieve       X
               REACODE=(R7),          Register 0 is stored here        X
               RECALL=(R8),           Recall Number of Days            X
               LENGTH=(R10)          Length of retrieve
* Ran out of registers above -- add remaining PTRs
         L     R0,RETURN_CODE_PTR
         L     R2,RETURN_CODE2_PTR
         L     R3,STORAGE_CLASS_PTR
         L     R4,TRACKING_TOKEN_PTR
*
     OSREQ (STORE),MF=(M,PARM_LIST),                                   X
               RETCODE=(R0),           Register 15 is stored here      X
               RETCODE2=(R2),          Return Code 2                   X
               STORCLAS=(R3),                                          X
               TTOKEN=(R4)             User Tracking Token
*
* Set RELBUF=YES if DATAAREA RELEASE_BUFFER == "YES"
TRYRELBUF      DS   0H
               CLC RELEASE_BUFFER,=CL3'YES'
               BNE BUFDONE             RELBUF=NO is default         @L1C
     OSREQ (STORE),MF=(M,PARM_LIST),                                   X
               RELBUF=YES              Will release pages after STORE
BUFDONE        DS   0H                                              @L1A
*                                                                   @L1A
* Set RETPD or EVENTEXP or both, based on caller's parm list.      @L1A
*                                                                   @L1A
* Note that a runtime error will occur if non-zero pointers are    @L1A
* present for both RETPD and EVENTEXP. Supplying both RETPD and    @L1A
* EVENTEXP is generally only useful for testing the error checking @L1A
* features of the OSREQ processing code.                           @L1A
*                                                                   @L1A
         L     R9,RETENTION_PERIOD_PTR                              @L1A
         OSREQ (STORE),MF=(M,PARM_LIST),                            @L1AX
               RETPD=(R9)                                           @L1A
*                                                                   @L1A
         L     R9,EVENTEXP_PTR                                      @L1A
         OSREQ (CHANGE),MF=(M,PARM_LIST),  EVENTEXP only on CHANGE @L1AX
               EVENTEXP=(R9)                                        @L1A
*                                                                   @L1A
* Set the DELHOLD parm or leave it off.                             @L1A
*                                                                   @L1A
DELHCHK  DS    0H                                                   @L1A
         CLC   DELHOLD,=CL8'HOLD'                                   @L1A
         BE    DELHYES                                              @L1A
*                                                                   @L1A
         CLC   DELHOLD,=CL8'NOHOLD'                                 @L1A
         BE    DELHNO                                               @L1A
         B     DELHDONE                                             @L1A
*                                                                   @L1A
DELHNO   DS    0H                                                   @L1A
         OSREQ (STORE),MF=(M,PARM_LIST),                            @L1AX
               DELHOLD=NOHOLD                                       @L1A
         B     DELHDONE                                             @L1A
*                                                                   @L1A
DELHYES  DS    0H                                                   @L1A
         OSREQ (STORE),MF=(M,PARM_LIST),                            @L1AX
               DELHOLD=HOLD                                         @L1A
DELHDONE DS    0H                                                   @L1A
*
* Keep testing FUNCTION_REQUEST until an OSREQ FUNCTION match is found
* or no more functions are found
* If a match is found, then go ahead and execute that function
*

* Execute ACCESS if FUNCTION_REQUEST == "ACCESS"
TRY_ACCESS     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)
* otherwise, the connection is done by the environment in which
* this program is executing.
* In all cases system control blocks will be created and/or modified
* to provide this access to OAM.
*
* To specify the default DSNHLI entry point for the
* IADDRESS keyword in the OSREQ function, specify
* the SYSPARM value as IADD in the PARM field of
* the EXEC JCL statement. See NOTE in prolog.                   @P1C
*
      AIF ('&SYSPARM' EQ 'IADD').IA2

*
* IADD not specified, so set IADDRESS OSREQ macro keyword to
* IADDRESS_PTR using the MF=M form of the macro.                @P1A
*
         L     R2,IADDRESS_PTR        Load IADR from parmList
     OSREQ ACCESS,MF=(E,PARM_LIST),                              X
               IADDRESS=(R2)
         AGO   .SKIP1
.IA2     ANOP
* IADD was specified so set default entry point.                @P1A
* In this sample we use DSNHLI for SQL interface module to DB2
*
         L     R2,=V(DSNHLI)
     OSREQ ACCESS,MF=(E,PARM_LIST),                              X
               IADDRESS=(R2)          GET THE ADDRESS OF THE INTERFACE
.SKIP1   ANOP
               B     SAVE_RC

* Execute STORE if FUNCTION_REQUEST == "STORE"
TRY_STORE      DS    0H
               CLC FUNCTION_REQUEST,=CL8'STORE'
               BNE TRY_RETRIEVE
*
* 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),                               X
               BUFLIST=(R10)
               B     SAVE_RC

* Execute RETRIEVE if FUNCTION_REQUEST == "RETRIEVE"
TRY_RETRIEVE   DS    0H
               CLC FUNCTION_REQUEST,=CL8'RETRIEVE'
               BNE TRY_QUERY
*
* 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=(M,PARM_LIST),                          X
               VIEW=PRIMARY,          Retrieve Primary Copy      X
               BUFLIST=(R10)
*
* if view=2, the set VIEW=BACKUP
TRYVIEW2       DS    0H
               SLR   R6,R6            Zero Register
               L     R6,VIEW          Load view into R6          @P0C
               LA    R10,2            Load value 2 into R10
               CR    R6,R10           Does view = 2?
               BNE   TRYVIEW3         No, then see if view = 3
     OSREQ (RETRIEVE),MF=(M,PARM_LIST),                          X
               VIEW=BACKUP            Retrieve First Backup Copy
               B     DO_RETRIEVE      Skip test 'if view=3'
```

```
*
* else if view=3, then set VIEW=BACKUP2
TRYVIEW3       DS    0H
               LA    R10,3            Load value 3 into R10
               CR    R6,R10           Does view = 3?
               BNE   DO_RETRIEVE      Nope, so leave VIEW=PRIMARY
     OSREQ (RETRIEVE),MF=(M,PARM_LIST),                                X
               VIEW=BACKUP2           Retrieve First Backup Copy

* Execute the Retrieve
DO_RETRIEVE    DS   0H
     OSREQ RETRIEVE,MF=(E,PARM_LIST)
               B     SAVE_RC

* Execute QUERY if FUNCTION_REQUEST == "QUERY"
TRY_QUERY      DS    0H
               CLC FUNCTION_REQUEST,=CL8'QUERY'
               BNE TRY_CHANGE
*
* Query the data base 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),                                    X
               QEL=(R10)
               B     SAVE_RC

* Execute CHANGE if FUNCTION_REQUEST == "CHANGE"
TRY_CHANGE     DS    0H
               CLC FUNCTION_REQUEST,=CL8'CHANGE'
               BNE TRY_DELETE
*
* 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     SAVE_RC
*
* Execute DELETE if FUNCTION_REQUEST == "DELETE"
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     SAVE_RC
*
* Execute UNACCESS if FUNCTION_REQUEST == "UNACCESS"
TRY_UNACCESS   DS    0H
               CLC FUNCTION_REQUEST,=CL8'UNACCESS'
               BNE TRY_STOREBEG
*
* 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)
               B     SAVE_RC
*
* Execute STOREBEG if FUNCTION_REQUEST == "STOREBEG
TRY_STOREBEG   DS    0H
```

```
               CLC FUNCTION_REQUEST,=CL8'STOREBEG'
               BNE TRY_STOREPRT
*
               ICM   R9,15,STIMEOUT_PTR Any STIMEOUT value?        @P2A
               BZ    DO_STOREBEG        No                         @P2A
*                                                                  @P2A
      OSREQ STOREBEG,MF=(M,PARM_LIST),                             @P2AX
               STIMEOUT=(R9)                                       @P2A
DO_STOREBEG    DS    0H                                            @P2A
* Begin the sequential storage of an object in parts.
       OSREQ STOREBEG,MF=(E,PARM_LIST),                                 X
                STOKEN=STOKEN_AREA
               B     SAVE_RC
*
* Execute STOREPRT if FUNCTION_REQUEST == "STOREPRT"
TRY_STOREPRT   DS    0H
               CLC FUNCTION_REQUEST,=CL8'STOREPRT'
               BNE TRY_CANCEL
* Store the next sequential contiguous part of an object
               L     R9,STORE_BUFFER_PTR
       OSREQ STOREPRT,MF=(E,PARM_LIST),                                 X
                BUFLIST=(R9),                                           X
                STOKEN=STOKEN_AREA
                B     SAVE_RC
*

* Set CANCEL=YES if DATAAREA CANCEL == "YES"
TRY_CANCEL     DS   0H
               CLC CANCEL,=CL3'YES'
               BNE TRY_STOREEND       CANCEL=NO is default
     OSREQ (STOREEND),MF=(M,PARM_LIST),                                 X
               CANCEL=YES             Will CANCEL Store Sequence
* Execute STOREEND if FUNCTION_REQUEST == "STOREEND"
TRY_STOREEND   DS    0H
               CLC FUNCTION_REQUEST,=CL8'STOREEND'
               BNE INVALID_FUNC
* End the sequential storage of an object in parts.
*               L     R10,CANCEL
       OSREQ STOREEND,MF=(E,PARM_LIST),                                 X
               STOKEN=STOKEN_AREA
               B     SAVE_RC
*
* None of the OSREQ functions matched FUNCTION_REQUEST, so set error
INVALID_FUNC   DS    0H
               LA    R15,ERR_FUNC   Set invalid function request
               B     EXIT
*
* 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.
*
SAVE_RC        DS    0H
               L     R2,RETURN_CODE_PTR
               L     R3,REASON_CODE_PTR
               ST    R15,0(,R2)     Save Return Code to RETURN_CODE_PTR
               ST    R0,0(,R3)      Save Reason Code to REASON_CODE_PTR
               LA    R15,0          Reset R15 back to zero to indicate
*                                   that the osreq function was
*                                   invoked
*
* Restore all registers except regs 15 and 0, then return to caller
EXIT           DS    0H
               L     R13,SAVE_AREA+4
               L     R14,12(R13)
               LM    R1,R12,24(R13)
               BR    R14
```

```
*
* 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
*
* Header Constants
*
*OSR2_ID   EQU   "OSR2"
OSR2_VER   EQU   1
OSR2_VER   EQU   2
*
* Header Validity Checking Error Codes
ERR_FUNC EQU   6                  Invalid Function Request
ERR_ID   EQU   8                  Invalid Header ID
ERR_LEN  EQU   10                 Invalid Header Length
ERR_VER  EQU   12                 Invalid Header Version
ERR_REL  EQU   14                 Invalid Header Release

*
* 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
**********************************************************************
*
* Th DATAAREA must be obtained by the caller of OSR2 and presented
* as a parameter (R1) to OSR2. 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.
*
***********************************************************************
*
* DATAAREA Header
DA_ID                DS CL4    x0 identifier
DA_LEN               DS F      x4 DATAAREA length--x280 (640) @01C
DA_VER               DS X      x8 DATAAREA version
DA_REL               DS X      x9 DATAAREA release
                     DS CL6    xA Reserved

*******
* The following two named fields are set by the caller of OSR2.
* If the value in the field is not a valid value, the respective
* activity cannot be executed.
```

```
*******
FUNCTION_REQUEST      DS CL8    x10 OSREQ function request value
*                               ACCESS, STORE, etc. or other
                      DS CL8    x18 Reserved
*******
* The following fields are set by OSR2 and should not be
* altered by the caller. Subsequent calls to OSR2 will rely
* on the information stored here.
*
* STOKEN NOTE:  The STOKEN must be kept on a DOUBLE WORD boundary
*******
TOKEN_AREA            DS 2F     x20 OSREQ token, do not change it.
STOKEN_AREA           DS 4F     x28 OSREQ stoken, do not change it.
                      DS 8F     x38 Reserved
*******
* The following fields are set by the caller of OSR2.
* The pointers are not altered by OSR2 but the data that
* the pointers reference may be.
*******
*
CANCEL                DS CL3    x58 CANCEL value, YES or other
                      DS CL1    x5B Reserved
COLLECTION_NAME_PTR   DS A      x5C Pointer to collection name
IADDRESS_PTR          DS A      x60 Reserved for IADDRESS_PTR
MANAGEMENT_CLASS_PTR  DS A      x64 Pointer to management class parm
MESSAGE_AREA_PTR      DS A      x68 Pointer to message area
OBJECT_NAME_PTR       DS A      x6C Pointer to object name
OBJECT_SIZE_PTR       DS A      x70 Pointer to object size value
OFFSET_PTR            DS A      x74 Pointer to offset value
QUERY_BUFFER_PTR      DS A      x78 Pointer to query buffer list
REASON_CODE_PTR       DS A      x7C Pointer to OSREQ reason code
RECALL_NUM_DAYS_PTR   DS A      x80 Recall Number of Days Pointer
RELEASE_BUFFER        DS CL3    x84 RELBUF value, YES or other
                      DS CL1    x87 Reserved
RETENTION_PERIOD_PTR  DS A      x88 Pointer to retention period
RETRIEVE_LENGTH_PTR   DS A      x8C Pointer to retrieve length value
RETRIEVE_BUFFER_PTR   DS A      x90 Pointer to retrieve buffer list
RETURN_CODE_PTR       DS A      x94 Pointer to OSREQ return code
RETURN_CODE2_PTR      DS A      x98 Return Code 2 Pointer
STIMEOUT_PTR          DS A      x9C Store Timeout Pointer          @P2C
STORE_BUFFER_PTR      DS A      xA0 Pointer to store buffer list
STORAGE_CLASS_PTR     DS A      xA4 Pointer to storage class parameter
TRACKING_TOKEN_PTR    DS A      xA8 User Tracking Token Pointer
VIEW                  DS F      xAC Retrieve Object Copy
*                                   1 = PRIMARY
*                                   2 = First BACKUP Copy
*                                   3 = Second BACKUP Copy
DELHOLD               DS CL8    xB0 DELHOLD= HOLD | NOHOLD | blank @L1A
EVENTEXP_PTR          DS A      xB8 Pointer to EVENTEXP            @L1A
                      DS CL124  xBC Reserved for future keywords   @01C
*
* Register Save Area
SAVE_AREA             DS 18F    x138 Savearea for this module.     @01C
*
********
* The following area is completely overlaid each time OSR2
* is called
********
PARM_LIST DS CL(STATIC_LIST_END-STATIC_PARM_LIST)  x180 Dynamic
*                                                    parm list  @01C
*
                      DS CL136   x1F8 Reserved -- To keep the DATAAREA
*                                     length constant, please subtract
*                                     PARM_LIST growth from this
*                                     reserved space.              @01C
*
```

```
DATAAREA_LEN   EQU *-DATAAREA
OSR2SAMP CSECT
*
         END   OSR2SAMP
```

Appendix B. Reason codes

Table 3 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* 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 31.

Table 3. Return/Reason Codes

Return code	Reason code (bytes)				Error description	Installation action
	0	1	2	3		
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.
4	4	x	4	z	A STORE or CHANGE request has completed but one or more of the following conditions occurred, as indicated by bits set in byte 3 (z). Z=BIT MAP: **1xxx xxxx** Catalog entry was created for the collection **x1xx xxxx** ODRETDT overrode RETPD, EVENTEXP, or Management Class expiration date **xx1x xxxx** Storage class specified for the collection was overridden **xxx1 xxxx** Management class specified for the collection was overridden **xxxx 1xxx** Retention period specified for the object by RETPD or EVENTEXP was overridden **xxxx x1xx** Reserved **xxxx xx1x** Storage class specified for the object was overridden **xxxx xxx1** Management class specified for the object was overridden	Issue query to see new parameters, if desired.
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 3. 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	2	3	The size specified on an OSREQ STOREBEG is not greater than 50 megabytes.	
8	24	x	2	4	The size specified on an OSREQ STOREPRT is not less than or equal to the total object size specified on the OSREQ STOREBEG for this store sequence.	
8	24	x	2	5	The size specified on an OSREQ STOREPRT when added to all of the previous OSREQ STOREPRT requests exceeds the total object size specified on the OSREQ STOREBEG for this store sequence.	
8	24	x	2	6	The size specified on an OSREQ STOREEND is not equal to the size specified on the OSREQ STOREBEG for this store sequence.	
8	24	x	2	7	The size specified on an OSREQ STOREEND is not equal to the total of the object sizes provided with previous OSREQ STOREPRT requests for this store sequence.	Check the previous STOREPRT requests to ensure that they provided all of the parts of the object data and that these previous STOREPRT requests were all successful.
8	24	x	2	8	The size specified on an OSREQ STOREPRT is less than the minimum part size allowed. Only the last STOREPRT in the store sequence can be less than the minimum.	
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	RETPD invalid value, must be -2 thru 93000 or X'7FFFFFFF'.	
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.	
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.	

Table 3. Return/Reason Codes (continued)

Return code	Reason code (bytes) 0	1	2	3	Error description	Installation action
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	8	A	When storing an object greater than 50 MB and less than or equal to 256 MB, multiple data buffers are supplied, which are not in contiguous storage.	
8	24	x	9	z	TOKEN	
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 one or more wildcard characters ('*','%', '_') but the function is not QUERY.	
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.	

Table 3. Return/Reason Codes (continued)

Return code	Reason code (bytes)				Error description	Installation action
	0	1	2	3		
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 ("*") wildcard and/or an invalid mix of asterisks with percent and/or underscore ("%" or "_") characters.	
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	
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	C	4	The offset specified on an OSREQ STOREPRT is not immediately following the last part of the object stored on the previous OSREQ STOREPRT for this store sequence or is not zero for the first OSREQ STOREPRT for this store sequence.	
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	D	4	The length specified on an OSREQ RETRIEVE is greater than 256 megabytes.	
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.	

Table 3. Return/Reason Codes (continued)

Return code	Reason code (bytes) 0	1	2	3	Error description	Installation action
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 (@, #, $) character.	
8	24	x	F	5	The COLLECTION NAME passed to OAM on the OSREQ macro contains a null qualifier, meaning ONE of the following is true. • The first character of the COLLECTION NAME is a period. • The last character of the COLLECTION NAME is a period. • The COLLECTION NAME contains two consecutive periods.	
8	24	x	F	6	Reserved	
8	24	x	F	7	The COLLECTION NAME passed to OAM on the OSREQ macro contains an invalid qualifier. One of the qualifiers does not start with an uppercase alphabetic character (A-Z) or national character ($, #, @).	
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	24	x	12	1	The RECALL parameter is in unusable storage 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	24	x	12	2	The RECALL parameter is larger than the maximum allowed.	
8	24	x	12	3	The RECALL parameter is a negative number.	
8	24	x	13	1	The RETCODE2 parameter is in unusable storage 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	24	x	14	z	STOKEN	
8	24	x	14	1	The STOKEN parameter is in unusable storage 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	24	x	14	2	The STOKEN value provided does not represent a store sequence currently in progress.	
8	24	x	15	z	STIMEOUT	
8	24	x	15	1	The STIMEOUT 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 STIMEOUT (fullword).	
8	24	x	15	2	The value specified for STIMEOUT is invalid.	
8	24	x	16	1	EVENTEXP area is unusable storage.	
8	24	x	16	2	EVENTEXP invalid value, must be 0 to 93 000.	
8	24	x	16	3	EVENTEXP and RETPD both supplied, only one allowed.	
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	

Table 3. Return/Reason Codes (continued)

Return code	Reason code (bytes)				Error description	Installation action
	0	1	2	3		
8	2C	x	y	z	No valid object was found. **z** RESERVED AND UNDEFINED	
8	2C	x	1	z	The directory entry was not found.	
8	2C	x	2	z	The object segment was not found.	
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.	
8	2C	x	5	z	The specified object's size is larger than the maximum object size supported by the OSREQ function at the current system level. Retry the requested OSREQ function on a system that supports objects of such size.	
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.	
8	38	x	y	z	Store sequence with STOREBEG, STOREPRT, STOREEND error **z** RESERVED AND UNDEFINED	
8	38	x	1	z	A store sequence function (STOREBEG, STOREPRT, STOREEND) was issued while a STOREBEG is in progress	
8	38	x	2	z	A store sequence function (STOREBEG, STOREPRT, STOREEND) was issued while a STOREPRT is in progress	
8	38	x	3	z	A store sequence function (STOREBEG, STOREPRT, STOREEND) was issued while a STOREEND is in progress	
8	38	x	4	z	A store sequence could not be begun (STOREBEG) because the object location of Optical is not supported for a store sequence	
8	38	x	6	z	On a STOREPRT or STOREEND request for an object to be stored to disk sublevel 1, an attempt to access the DB2 buffer resulted in a -423 DB2 SQL code. It was determined that the DB2 buffer can no longer be accessed. **Note:** This could be the case that the DB2 locator is invalid because the application did a COMMIT during the store sequence. A STOREBEG or STOREPRT request has completed. This store sequence is not finished because an UNACCESS has been issued.	Ensure that the application program did not issue a COMMIT or ROLLBACK during a store sequence, which can be a cause of the -423 DB2 SQL code. Once a STOREBEG has been issued, the application cannot perform a COMMIT or ROLLBACK until after the corresponding STOREEND for the store sequence. Also see the IBM Information Management Software for z/OS Solutions Information Center for more information on the -423 DB2 SQL code.
8	38	x	7	z	A STOREBEG or STOREPRT request has completed after an UNACCESS has been issued. In this case, the UNACCESS will be deferred and will fail because of the pending store sequence.	Ensure that the application program does a STOREEND to finish the store sequence or a STOREEND with CANCEL=YES to cancel the store sequence, then issue UNACCESS again.
8	38	x	8	z	The expected length of the object to be retrieved is greater than the maximum retrieval buffer size of 256 megabytes. If LENGTH has a value of 0 or is not specified on an OSREQ RETRIEVE request, then by default the length will be set to the length from either the offset (if OFFSET specified) or beginning (if OFFSET not specified) to the end of the object.	

Table 3. Return/Reason Codes (continued)

Return code	Reason code (bytes)				Error description	Installation action
	0	1	2	3		
8	38	x	9	z	UNACCESS request can not be processed because of a pending store sequence.	Ensure that the application program did a STOREEND to finish the store sequence or a STOREEND with CANCEL=YES to cancel the store sequence before issuing UNACCESS.
8	38	x	0A	z	On a STOREPRT or STOREEND request for an object to be stored to Tape, it was determined that the buffer can no longer be accessed in the OAM address space. A possible cause could be too low STIMEOUT value specified on the OSREQ STOREBEG.	
8	3C	5	1	1	OSREQ CHANGE: The EVENTEXP parameter is not allowed because the object is not waiting for an event-based-retention event.	
8	3C	5	2	2	OSREQ CHANGE: The RETPD parameter not allowed for an event-based-retention object.	
8	40	6	1	0	OSREQ DELETE: Deletion is not allowed because the object is in DELHOLD=HOLD state.	
8	40	6	2	1	OSREQ DELETE: Deletion is not allowed because the object is under deletion-protection and is still in event-based-retention state.	
8	40	6	2	2	OSREQ DELETE: Deletion is not allowed because the object is under deletion-protection and the object's expiration date has not yet been reached.	
8	40	6	3	1	OSREQ DELETE: Deletion is not allowed because the object is under retention-protection and is still in event-based-retention state.	
8	40	6	3	3	OSREQ DELETE: Deletion is not allowed because the object is under retention-protection and the object's retention date has not yet been reached.	

Appendix C. Performance considerations and object data reblocking

This appendix documents diagnosis, modification, or tuning information that 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 these 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. On OSREQ STORE function, if the object data is in one contiguous block in a storage area immediately following the end of the buffer list, then the data is not moved within the caller's address space. On OSREQ RETRIEVE function, 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 might obtain temporary storage to reblock the data. The virtual storage needed, in addition to the calling program's requirements, might be as great as the lesser of 256 megabytes or the size of the largest object.

Object storage

When using the OSREQ STORE function, 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 might be 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 disk sublevel 1 (DB2), 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 is obtained.
  - If the total object data length is greater than 3980 bytes and the destination is a DB2 32K table, a temporary storage buffer of 32KB is obtained.
- If the object is to be stored initially on disk sublevel 2 (file system), optical media, or tape 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 disk sublevel 1 (DB2) and are 3980 bytes or less in length, or for objects that are stored on disk sublevel 1 and are greater than 32640 bytes in length and the destination is a DB2 LOB table, or for objects that are stored on disk sublevel 2 (file system), optical media, or tape media, only one block is created and stored.

For objects that are stored on disk sublevel 1 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.

When using the OSREQ store sequence functions (STOREBEG, STOREPRT, and STOREEND) to store an object in multiple parts, there is no temporary storage needed within the caller's address space. It is recommended to avoid unnecessary overhead by:
- Maximizing the size of each part of the object to be stored with STOREPRT and
- Minimizing the number of STOREPRT invocations.

Object retrieval

For objects that are retrieved from disk sublevel 1 , 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 disk sublevel 2 (file system), optical, or tape 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 these 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 lesser of 256 megabytes or 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.

The sample CBRUXSAE exit in SAMPLIB, defaults to returning a return code 16 indicating "Bypassed", meaning that the current and all future user IDs are authorized to perform all OSREQ functions and that the exit need not be called again. Installations must substitute this code with a validation routine to determine authority for a specific user ID in order for authorization checking to be performed at the application interface level.

This support provides more return codes to be processed by the CBRUXSAE security authorization user exit. The additional return codes enable an installation to code up their CBRUXSAE user exit to:

- Bypass the exit for any combination of functions. For example, the exit can be bypassed for OSREQ QUERY and RETRIEVE requests but active for OSREQ STORE, CHANGE and DELETE requests.
- Authorize users to store objects into an existing collection while preventing them from creating new collections.

If the return code from CBRUXSAE is not 0, 16 or 255 (or 253 or 254 when storing to an existing collection); return and reason codes are issued indicating that the user ID is not authorized to perform the particular OSR function. For more information concerning return and reason codes associated with this exit, refer to *z/OS DFSMSdfp Diagnosis*.

Return codes from CBRUXSAE are interpreted as follows:

Table 4. CBRUXSAE return codes

Return Code	**Description**
0	AUTHORIZEDUser is authorized to perform this function. The exit will continue to be called for all normally called OSREQ functions: STORE, RETRIEVE, QUERY, CHANGE, DELETE and STORE BEGIN.
16	BYPASSED The current user and all future users are authorized. Exit will now be BYPASSED (not called again for any function.)
224-252	RESERVED (Not Authorized) Reserved for IBM. It is recommended that installations do not use return code values in this range because their meaning could change in the future. However, they are currently interpreted as: User is not authorized to perform this function. No change is made to the BYPASS status of any OSREQ function.

Table 4. CBRUXSAE return codes (continued)

Return Code	Description
253	STORE RESTRICTED (No Bypass) Store to existing collection only. • For STORE (and STORE BEGIN) function: User is authorized to store into an existing collection only. Attempts to store into a collection that does not exist will fail • All other OSREQ functions: NOT Authorized This is valid for the current invocation only. No change is made to the BYPASS status of any OSREQ function.
254	BYPASS CURRENT FUNCTION (IF STORE, RESTRICTED) Current and future users are authorized to perform the current function. The exit will be BYPASSED (not called again) for the current function. If the current function is a STORE (or STORE BEGIN) then this exit will be bypassed for subsequent STORE requests. This STORE request and subsequent STORE requests will be allowed into existing collections only. Attempts to store into a collection that does not exist will fail. **Note:** If an administrator needs to create a new collection after this has been set, he or she will have to first reset the exit with the LIBRARY RESET,CBRUXSAE operator command. For all other OSREQ functions, this exit will be bypassed (Authorized) for that particular function. For example, if the current function is RETRIEVE, then this RETRIEVE request and all subsequent RETRIEVE requests will be allowed. The same applies for QUERY, CHANGE and DELETE.
255	BYPASS CURRENT FUNCTION (IF STORE, NOT RESTRICTED) Current and future users are authorized to perform the current function. The exit will be BYPASSED (not called again) for the current function. If the current function is a STORE (or STORE BEGIN) then this exit will be bypassed for subsequent STORE requests. This STORE request and subsequent STORE requests will be allowed to store to both new and existing collections. For all other OSREQ FUNCTIONS, this exit will be bypassed (Authorized) for that particular function. For example, if current function is RETRIEVE, then this RETRIEVE request and all subsequent RETRIEVE requests will be allowed. The same applies for QUERY, CHANGE and DELETE. **Note:** Return codes 254 and 255 have the same meaning for all functions except the store functions (STORE and STORE BEGIN).
Any other non-zero	User is not authorized to perform this function.

Note: OSREQ STOREBEG is considered a STORE function from a CBRUXSAE exit perspective.

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. Note that during a store sequence using the STOREBEG, STOREPRT, and STOREEND functions, the CBRUXSAE exit is only invoked once for the sequence, the invocation will occur during the STOREBEG function, and will be identified to the exit with the value STORE. 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	If the return code from CBRUXSAE is not 0, 16 or 255 (or 253 or 254 when storing to an existing collection); return and reason codes are issued indicating that the user ID is not authorized to perform the particular OSR function. For more information concerning return and reason codes associated with this exit, refer to *z/OS DFSMSdfp Diagnosis*.

Programming the CBRUXSAE exit correctly

CBRUXSAE is provided as a separate load module that must 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 "Sample CBRUXSAE installation exit" for a sample of the CBRUXSAE installation exit.

Sample CBRUXSAE installation exit

Here is 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             *
*                                                                   *
*    PROPRIETARY V3 STATEMENT                                       *
*    LICENSED MATERIALS - PROPERTY OF IBM                           *
*    5694-A01                                                       *
*    Copyright IBM Corp. 1996, 2009                                 *
*    END PROPRIETARY V3 STATEMENT                                   *
*                                                                   *
*    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   *
*      an appropriate return code in register 15 (described in      *
*      the OUTPUT section below).                                   *
*                                                                   *
*      Starting with z/OS V1R11, more granular return codes have    *
*      been implemented to allow bypassing the exit for each of the *
*      individual OSREQ functions in addition to the ability to     *
*      restrict STOREs to existing collections only.                *
*      The additional return codes enable an installation to bypass *
*      the exit for any combination of functions.  For example, the *
*      exit can be bypassed for OSREQ QUERY and RETRIEVE requests   *
*      but active for OSREQ STORE, and DELETE requests.             *
*                                                                   *
************************** !!! WARNING !!! ****************************
*      WARNING: Prior to z/OS V1R11, ANY non-zero return code (except *
*      RC 16 for BYPASS) meant "authorization failed".  Starting with *
*      z/OS V1R11, return codes 253, 254, and 255 have new meaning  *
*      as described in the OUTPUT section below. If you used 253,   *
*      254, or 255 in a pre-V1R11 version of CBRUXSAE, please review *
*      the new meanings and modify your exit appropriately.         *
************************** !!! WARNING !!! ****************************
*      THE INSTALLATION CAN PERFORM AUTHORIZATION CHECKING BY ANY   *
*      MEANS IT DEEMS REASONABLE.  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 0, 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                                            *
*
 *    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:                    *
* Return                                                            *
* Code    Description                                               *
* ------  ----------------------------------------------------------*
* 0       AUTHORIZED                                                *
*         User is authorized to perform this function.  The exit will *
*         continue to be called for all normally called OSREQ       *
*         functions:                                                *
*         STORE, RETRIEVE, QUERY, CHANGE, DELETE, and STORE BEGIN.   *
*                                                                   *
* 16      BYPASSED                                                  *
*         The current user and all future users are authorized. Exit  *
*         will now be BYPASSED (not called again for any function).  *
*                                                                   *
* 224-252 RESERVED (Not Authorized)                             @L1A*
*         Reserved for IBM.  It is recommended that installations do  *
*         not use return code values in this range because their     *
*         meaning could change in the future. However, they are      *
*         currently interpreted as:                                 *
*         User is not authorized to perform this function.  No change *
*         is made to the BYPASS status of any OSREQ function.        *
*                                                                   *
* 253     STORE RESTRICTED (No Bypass)                          @L1A*
*         Store to existing collection only.                        *
*         - For STORE (and STORE BEGIN) function:  User is authorized *
*           to store into an existing collection only. Attempts to   *
*           store into a collection that does not exist will fail.   *
*         - All other OSREQ functions: NOT Authorized.              *
*                                                                   *
*         This is valid for the current invocation only.  No change  *
*         is made to the BYPASS status of any OSREQ function.        *
*                                                                   *
* 254     BYPASS CURRENT FUNCTION (IF STORE, RESTRICTED)        @L1A*
*         Current and future users are authorized to perform the     *
*         current function. The exit will be BYPASSED (not called    *
*         again) for the current function.  If the current function  *
*         is a STORE (or STORE BEGIN) then this exit will be bypassed *
*         for subsequent STORE requests.  This STORE request and     *
*         subsequent STORE requests will be allowed into existing    *
*         collections only.  Attempts to store into a collection that *
*         does not exist will fail.                                 *
*         Note: If an administrator needs to create a new collection  *
*         after this has been set, he'll have to first reset the exit *
*         via the  LIBRARY RESET,CBRUXSAE operator command.          *
*                                                                   *
*         For all other OSREQ FUNCTIONS, this exit will be bypassed  *
*         (Authorized) for that particular function. For example, if  *
*         current function is RETRIEVE, then this RETRIEVE request   *
*         and all subsequent RETRIEVE requests will be allowed. The  *
*         same applies for QUERY, CHANGE, and DELETE.               *
*                                                                   *
* 255     BYPASS CURRENT FUNCTION (IF STORE, NOT RESTRICTED)    @L1A*
*         Current and future users are authorized to perform the     *
*         current function. The exit will be BYPASSED (not called    *
*         again) for the current function.  If the current function  *
*         is a STORE (or STORE BEGIN) then this exit will be bypassed *
*         for subsequent STORE requests.  This STORE request and     *
*         subsequent STORE requests will be allowed to store to both  *
*         new and existing collections.                             *
*                                                                   *
*         For all other OSREQ FUNCTIONS, this exit will be bypassed  *
*         (Authorized) for that particular function. For example, if  *
*         current function is RETRIEVE, then this RETRIEVE request   *
*         and all subsequent RETRIEVE requests will be allowed. The  *
*         same applies for QUERY, CHANGE, and DELETE.               *
*         Note: Return codes 254 and 255 have the same meaning for   *
*         all functions except the store functions (STORE and STORE   *
*         BEGIN).                                                     *
*                                                                     *
* Any                                                                 *
* other                                                               *
* non-                                                                *
* zero     User is not authorized to perform this function.       @L1A*
*                                                                     *
*      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*
*     $L1=OAMR1B  R11 080523 TUCTMD: OAMR1B CBRUXSAE Enhancement  @L1A*
*                                    Add new return codes for     @L1A*
*                                    STORE to existing Collection @L1A*
*                                    only, and BYPASS individual  @L1A*
*                                    OSREQ Functions              @L1A*
*                                                                     *
**** 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
         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&SYSDATE'   MARK ENTRY POINT
         LR    R12,R15            COPY ENTRY POINT ADDRESS
         USING CBRUXSAE,R12       CBRUXSAE BASE REGISTER
         USING WORKAREA,R9        ADDRESSABILITY 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       ADDRESSABILITY 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
```

Appendix E. Accessibility

Publications for this product are offered in Adobe Portable Document Format (PDF) and XHTML through the z/OS Information Center, at http://publib.boulder.ibm.com/infocenter/zos/v2r1/index.jsp. If you experience difficulty with the accessibility of any z/OS information, send an email to mhvrcfs@us.ibm.com or write to:

IBM® Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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 such products 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 Vol 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.

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users accessing the Information Center using a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol can be used next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol giving information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you should refer to separate syntax fragment OP1.

The following words and symbols are used next to the dotted decimal numbers:
- ? means an optional syntax element. A dotted decimal number followed by the ? symbol indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that syntax elements NOTIFY and UPDATE are optional; that is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.
- ! means a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicates that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In this example, if you include the FILE keyword but do not specify an option, default option KEEP will be applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP only applies to the next

higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

- * means a syntax element that can be repeated 0 or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3*, 3 HOST, and 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

  Note:

  1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
  2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you could write HOST STATE, but you could not write HOST HOST.
  3. The * symbol is equivalent to a loop-back line in a railroad syntax diagram.
- + means a syntax element that must be included one or more times. A dotted decimal number followed by the + symbol indicates that this syntax element must be included one or more times; that is, it must be included at least once and can be repeated. For example, if you hear the line 6.1+ data area, you must include at least one data area. If you hear the lines 2+, 2 HOST, and 2 STATE, you know that you must include HOST, STATE, or both. Similar to the * symbol, the + symbol can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loop-back line in a railroad syntax diagram.

Notices

This information was developed for products and services offered in the U.S.A. or elsewhere.

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 character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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:

Site Counsel
IBM Corporation
2455 South Road
Poughkeepsie, NY 12601-5400
USA

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.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information might contain sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, HCD, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (http://www.ibm.com/software/support/systemsz/lifecycle/)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming interface information

This publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of DFSMS Object Access Method (OAM).

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml.

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.

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.

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*.

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.

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*.

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.

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.

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.

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.

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.

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*.

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.

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.

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 disk (DB2 table spaces or file system), 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.

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.

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.

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

Product Number: 5650-ZOS

Printed in USA

SC23-6865-00