

z/OS
3.2

*DFSMS Object Access Method Application
Programmer's Reference*



Note

Before using this information and the product it supports, read the information in [“Notices” on page 103.](#)

This edition applies to IBM® z/OS® 3.2 (5655-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-30

© **Copyright International Business Machines Corporation 1986, 2025.**

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

Contents

Figures.....	v
Tables.....	vii
About this book.....	ix
Major divisions of this book.....	ix
Required product knowledge.....	ix
z/OS information.....	ix
How to read syntax diagrams.....	x
How to provide feedback to IBM.....	xiii
Summary of changes.....	xv
Summary of changes for z/OS 3.2.....	xv
Summary of changes for z/OS 3.1.....	xv
Chapter 1. Understanding the Object Access Method.....	1
Understanding OAM components.....	2
Establishing a storage management policy.....	2
Understanding the OAM application programming interface.....	3
Choosing data types that work well with OAM.....	4
Retrieving a partial object.....	4
Coordinating Db2, OAM, and your application.....	5
Coordinating your application with OAM's object identification.....	5
Overriding management policy defaults.....	5
Separating objects.....	6
Deleting objects.....	6
Chapter 2. Application program interface for OAM.....	7
Using the OSREQ macro.....	7
What you can do with OSREQ.....	7
Choosing the form.....	8
Getting the code right.....	8
Implementing the functions.....	9
ACCESS—Initializing the OSREQ interface.....	9
CHANGE—Changing an object's management characteristics.....	11
DELETE—Deleting an existing object.....	14
QUERY—Obtaining object characteristics.....	15
RETRIEVE—Retrieving an existing object.....	18
Adding objects to the object storage hierarchy.....	21
STORE function.....	23
STOREBEG—Beginning a Store Sequence operation.....	27
STOREPRT—Storing an individual part in a Store Sequence operation.....	30
STOREEND—Ending a Store Sequence operation.....	32
UNACCESS—Ending the OSREQ interface.....	34
OSREQ keyword parameter descriptions.....	34
Usage considerations.....	44
Usage requirements.....	45
Restrictions and limitations.....	45

Programming notes.....	46
Register use.....	47
Expiration date processing.....	47
Messages and codes.....	48
OAM return codes and reason codes.....	48
Db2 SQL error reason codes.....	49
CBRIBUFL macro.....	49
CBRIQEL macro.....	51
Appendix A. Sample program for object storage.....	57
CBROSREQ.....	59
CBROSR2.....	67
CBROSR3.....	75
CBROSRSP.....	83
Appendix B. Performance considerations and object data reblocking.....	91
Performance considerations.....	91
Object data reblocking.....	91
Object storage.....	91
Object retrieval.....	92
Appendix C. Using the CBRUXSAE installation exit.....	93
Register contents on entry to CBRUXSAE.....	94
Programming the CBRUXSAE exit correctly.....	95
Sample CBRUXSAE installation exit.....	95
Appendix D. Accessibility.....	101
Notices.....	103
Terms and conditions for product documentation.....	104
IBM Online Privacy Statement.....	105
Policy for unsupported hardware.....	105
Minimum supported hardware.....	105
Programming interface information.....	106
Trademarks.....	106
Glossary.....	107
Index.....	113

Figures

- 1. Example of devices that application may use..... 4
- 2. Conceptual view of a Store Sequence operation.....28
- 3. Fields Described by CBRIBUFL..... 50
- 4. Data Buffer List Structure Diagram..... 51
- 5. Fields Described by CBRIQEL..... 53
- 6. Query Buffer List Structure Diagram.....55

Tables

1. IADDRESS parameter effects in various processing environments..... 11

2. Deciding which OAM store function to use..... 22

3. Valid Retention Periods for Expiration Date Processing..... 47

4. CBRUXSAE return codes..... 93

About this book

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

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

Major divisions of this book

This book contains the following major divisions:

- Chapter 1, “Understanding the Object Access Method,” on page 1 provides an overview of concepts relating to objects and the Object Access Method.
- Chapter 2, “Application program interface for OAM,” on page 7 contains detailed information about the OSREQ macro and how it is used by application programs.
- Appendix A, “Sample program for object storage,” on page 57 provides assembler source code for a sample object storage request interface.
- Appendix B, “Performance considerations and object data reblocking,” on page 91 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 C, “Using the CBRUXSAE installation exit,” on page 93 details how this exit is used to provide security checking for the OSREQ macro.
- “Glossary” on page 107 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.

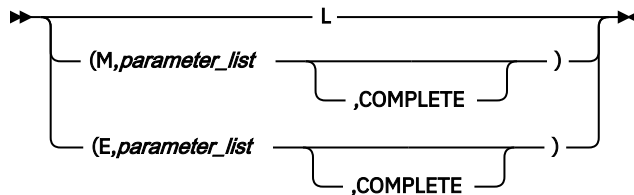
When possible, this information uses cross-document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

How to read syntax diagrams

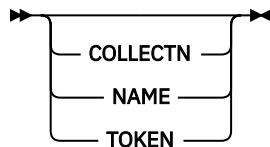
There is one basic rule for reading the syntax diagrams: Follow only one line at a time from the beginning to the end and code everything you encounter on that line.

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

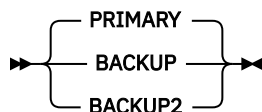
- Read the syntax diagrams from left to right and from top to bottom.
- Each syntax diagram begins with a double arrowhead (➤➤) and ends with opposing arrows (➤➤).
- An arrow (➤➤) at the end of a line indicates that the syntax continues on the next line. A continuation line begins with an arrow (➤➤).
- Commands, keywords, and macro invocations are shown in uppercase letters.
- Where you can choose from two or more keywords, the choices are stacked one above the other. If one choice within the stack lies on the main path, a keyword is required, and you must choose one. In the following example you must choose either L, M, or E.



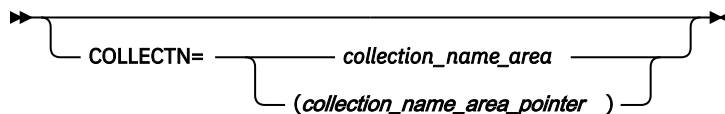
- If a stack is placed below the main path, a keyword is optional, and you can choose one or none. In the following example, TOKEN, COLLECTN, and NAME are optional keywords. You can choose any one of the three.



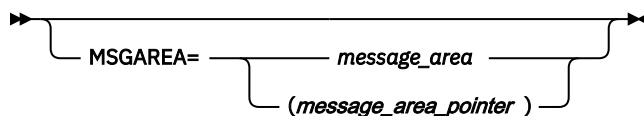
- Where you can choose from two or more keywords and one of the keywords appears above the main path, that keyword is the default. You may choose one or the other of the keywords, but if none is entered, the default keyword is automatically selected. In the following example you may choose either PRIMARY,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



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

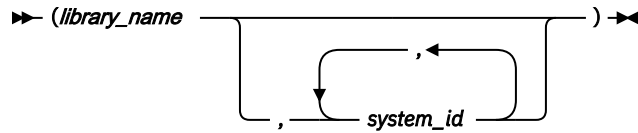


- You must provide all items enclosed in parentheses (). You must include the parentheses. In the following example, you must supply the volume serial number (*message_area_pointer*) and it must be enclosed in parentheses.



- The repeat symbol 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 provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. For more information, see [How to send feedback to IBM](#).

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Note: IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under [IBM z/OS Product Documentation Update Policy](http://www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy) (www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy).

Summary of changes for z/OS 3.2

The following content is new, changed, or no longer included in z/OS 3.2.

New

The following content is new.

September 2025 release

- None.

Changed

The following content is changed.

September 2025 release

- None.

Deleted

The following content is deleted.

September 2025 release

- None.

Summary of changes for z/OS 3.1

This information contains no technical changes for this release.

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
----------------	-------------

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, even though that 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, optical disk, or cloud 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, tape volumes, and cloud storage. Optical and tape volumes can be library-resident or shelf-resident. The primary copies of objects can be stored anywhere in the storage hierarchy while backup copies of objects can only be stored to optical or tape volumes, cloud, or file system. 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 OAM uses to determine where to place objects in the storage hierarchy (disk sub level 1 (Db2), disk sub level 2 (file system), optical, tape sublevel 1, tape sublevel 2, or cloud storage).

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 CBROAMxx 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 CBROAMxx 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 4 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

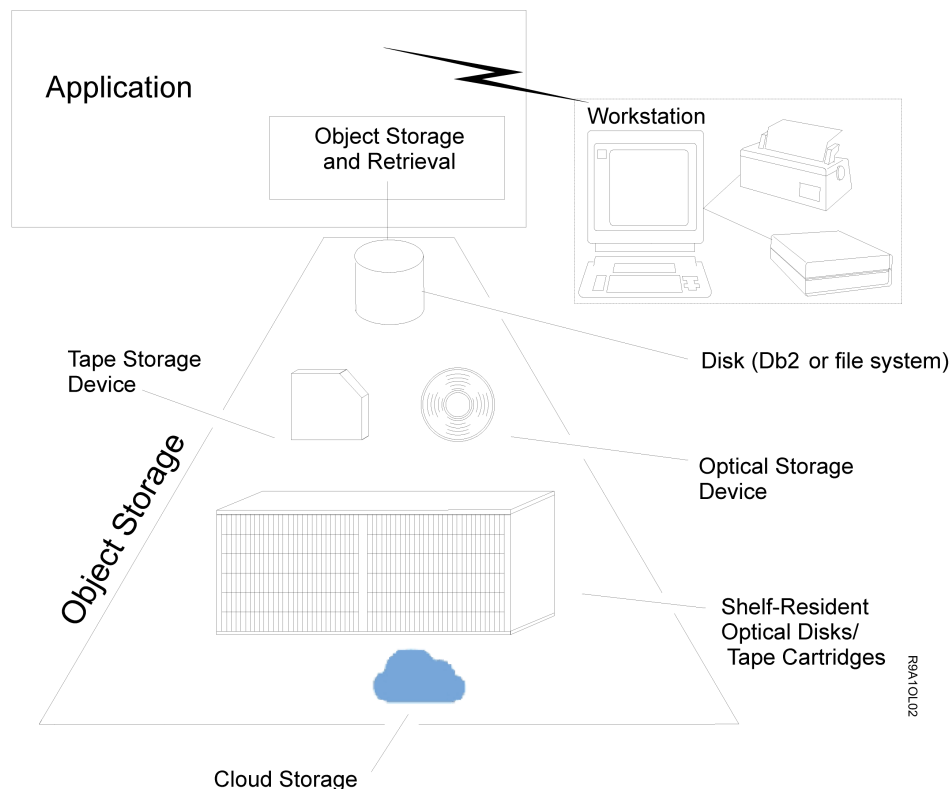


Figure 1. Example of devices that application may use

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 57 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 sub-assemblies. Storing the sub-assemblies 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 can be retrieved using a single OSREQ Retrieve only when using 64-bit buffers and the BUFFER64 option. To retrieve an object greater than 256 megabytes without using 64-bit buffers, 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 either the file system sublevel or the cloud level from an application program, the application must perform the Db2 "commit" within 24 hours of storing the object. Failure to do this ultimately results in loss of object data stored in the file system or cloud level.

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.

Also, understand the OAM configuration established by the system programmer. In a classic OAM configuration, there is only one instance of OAM (an OAM subsystem and, optionally, an OAM address space) and an association with a single Db2 subsystem. However, a multiple OAM configuration can have multiple instances of OAM used to perform object processing. Each object instance of OAM (an OAM subsystem and, optionally, an associated OAM Object address space) in a multiple OAM configuration is associated with a unique Db2 subsystem. Therefore, in a multiple OAM configuration, your application might, depending on the operating environment, need to explicitly identify the Db2 subsystem to be used to process the application request. This Db2 subsystem identification directs the application request to the appropriate OAM instance for object processing.

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 27](#).

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 B, [“Performance considerations and object data reblocking,”](#) on page 91 for performance considerations when you write the application program that interfaces with the OSREQ macro.

See Appendix C, [“Using the CBRUXSAE installation exit,”](#) on page 93 for information about the CBRUXSAE security authorization installation exit that is used at the OSREQ macro level. A sample installation exit is included.

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. In a multiple OAM configuration, identifies which OAM instance will process the set of requests.

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 triggering expiration criteria for an object in event-based-retention mode. A successful change could update the Last Reference Date and the Pending Action Date.

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.

When using 31-bit virtual storage buffers:

- For objects less than or equal to 256 megabytes you can use Store.
- For objects greater than 256 megabytes, you must use the Store Sequence functions (Storebeg, Storeprt, and Storeend).
- For objects greater than 50 megabytes and less than or equal to 256 megabytes, you can use either Store or the Store Sequence functions.

When using 64-bit virtual storage buffers, use Store for entire objects less than or equal to 2000M.

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 27](#) 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 30](#) 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 32](#) 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 9](#) contains detailed descriptions of the functions and their corresponding syntax diagrams.

Choosing the form

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

**MACRO FORM
DESCRIPTION****List (MF=L)**

Generates a parameter list that can be used with the other forms of the macro.

Modify (MF=M)

Updates the parameter list with new parameters (specified when the modify form is invoked).

Execute (MF=E)

Initiates execution of the actual object request; also updates the parameter list if new parameters are specified when the execute form is invoked.

Each form supports a variety of functions. These functions are described in [“What you can do with OSREQ” on page 7](#). Subsequent sections present detailed information about coding and invoking the macro to perform these functions. Use of the OSREQ macro must take into consideration both the programming language techniques and the environment in which the program executes. These issues are discussed in [“Usage considerations” on page 44](#).

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 34](#) 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 34](#).

- [“ACCESS—Initializing the OSREQ interface” on page 9](#)
- [“CHANGE—Changing an object's management characteristics” on page 11](#)
- [“DELETE—Deleting an existing object” on page 14](#)
- [“QUERY—Obtaining object characteristics” on page 15](#)
- [“RETRIEVE—Retrieving an existing object” on page 18](#)
- [“STORE function” on page 23](#)
- [“STOREBEG—Beginning a Store Sequence operation” on page 27](#)
- [“STOREPRT—Storing an individual part in a Store Sequence operation” on page 30](#)
- [“STOREEND—Ending a Store Sequence operation” on page 32](#)
- [“UNACCESS—Ending the OSREQ interface” on page 34](#)

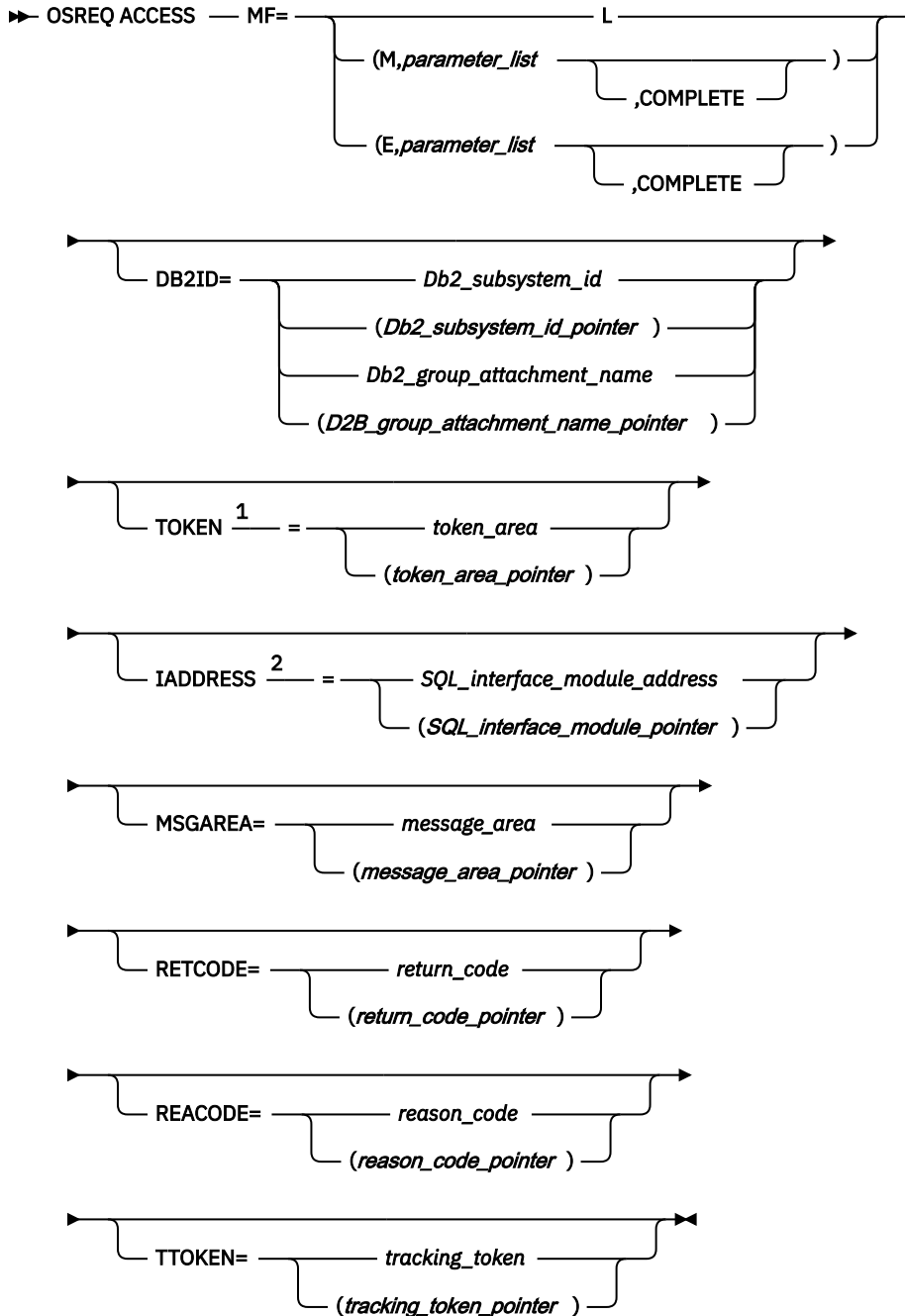
ACCESS—Initializing the OSREQ interface

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.

If in a multiple OAM configuration, OAM needs to establish a connection to Db2, the Db2 subsystem to be used for the request is specified with the DB2ID parameter. This specification of the Db2 subsystem identifies the OAM subsystem and any associated OAM address space for which the access is to be established.

The syntax diagram for the OSREQ ACCESS function follows.

Syntax for OSREQ ACCESS



Notes:

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

² This keyword indicates that a connection to Db2 already exists.

The OSREQ ACCESS function establishes the environmentally-dependent resources that are needed for other OSREQ function processing in the address space. In environments other than CICS, the DSN command processor, or Db2 stored procedures, 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 later in this topic.

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 or Db2 stored procedures), the IADDRESS= keyword must be used to specify the structured query language (SQL) interface module entry point address. This address is used for all SQL processing in the other OSREQ functions. See [Table 1 on page 11](#) for the effects of the IADDRESS parameter when used in various processing environments.

<i>Table 1. IADDRESS parameter effects in various processing environments</i>		
Processing environment	IADDRESS specified	IADDRESS not specified
IMS	Used	CAF error
MVS BATCH	Used (see note 1)	CAF success
CICS	Ignored	Not applicable
DSN Command Processor	Ignored	Not applicable
TSO	Used (see note 1)	CAF success
Db2 Stored Procedure	Used	RRSAF error

Notes:

1. If the Db2 CONNECT is not done by the application, a Db2 CONNECT and COMMIT is done for each SQL CALL.
2. Environments or invocations other than those listed in [Table 1 on page 11](#) have not been tested by IBM and the results can be unpredictable.

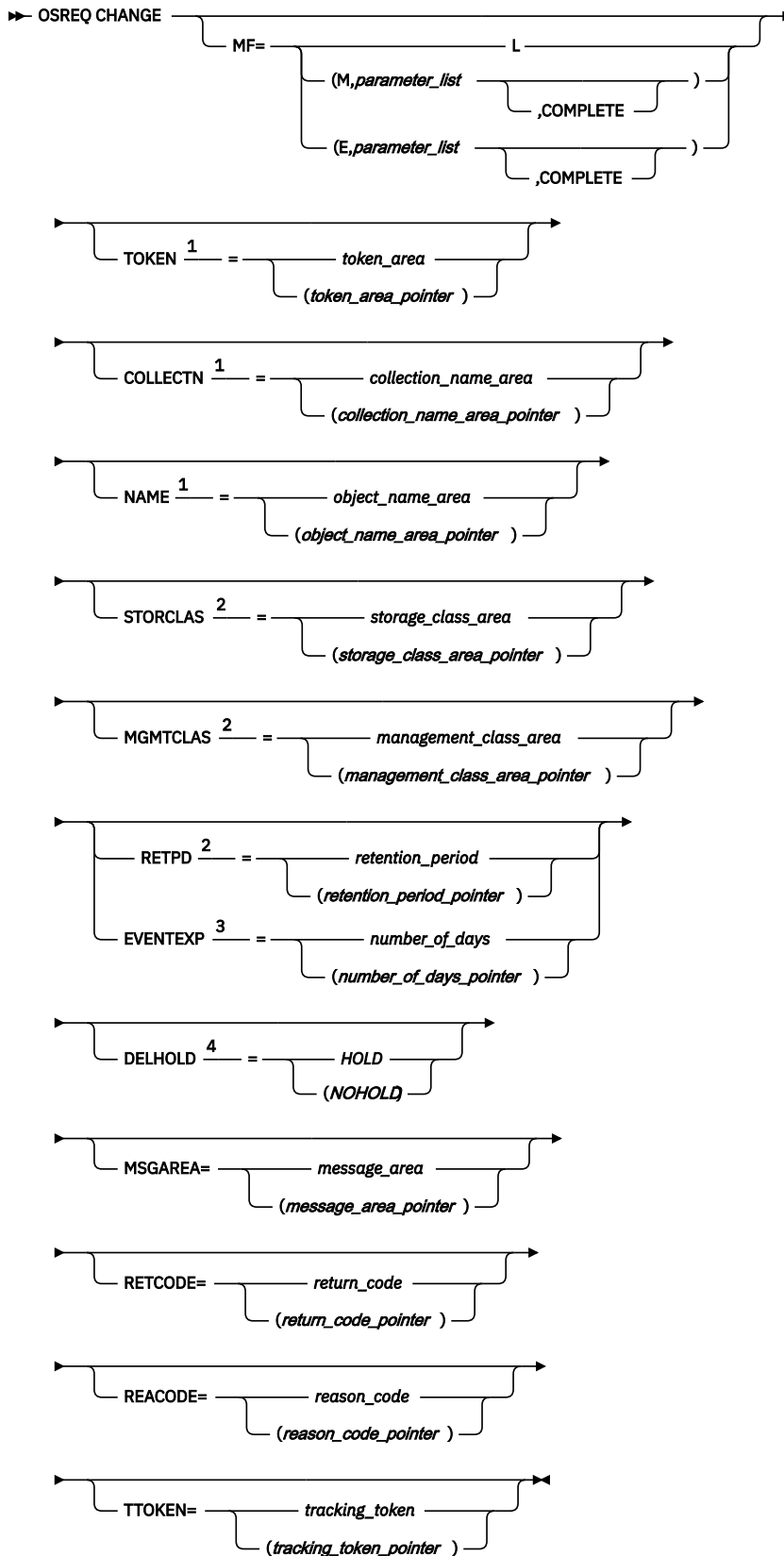
To limit the scope of database activities synchronized by the application, each application should issue its own ACCESS. The application must observe the Db2 restrictions regarding multiple threads from a single task as described in [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](#).

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

CHANGE—Changing an object's management characteristics

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

Syntax for OSREQ CHANGE



Notes:

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

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'FFFFFFFFE') 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'FFFFFFFFE'), 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 47](#) to see the rules in more detail.

- If a RETPD of X'7FFFFFFFF' (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 3 on page 47](#)).

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 ODRETD 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 47 for more information.

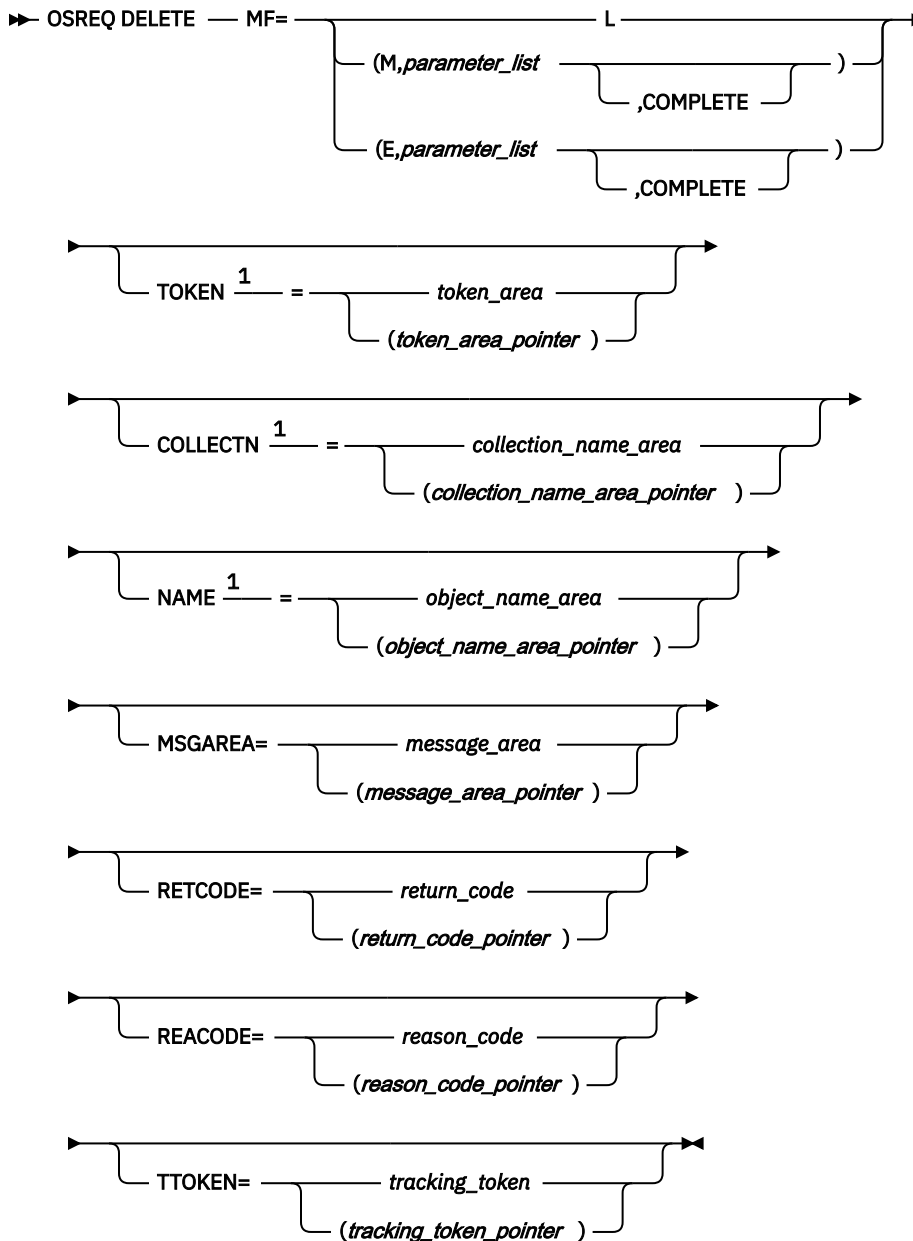
DELETE—Deleting an existing object

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 and backup copies 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



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 51.

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.

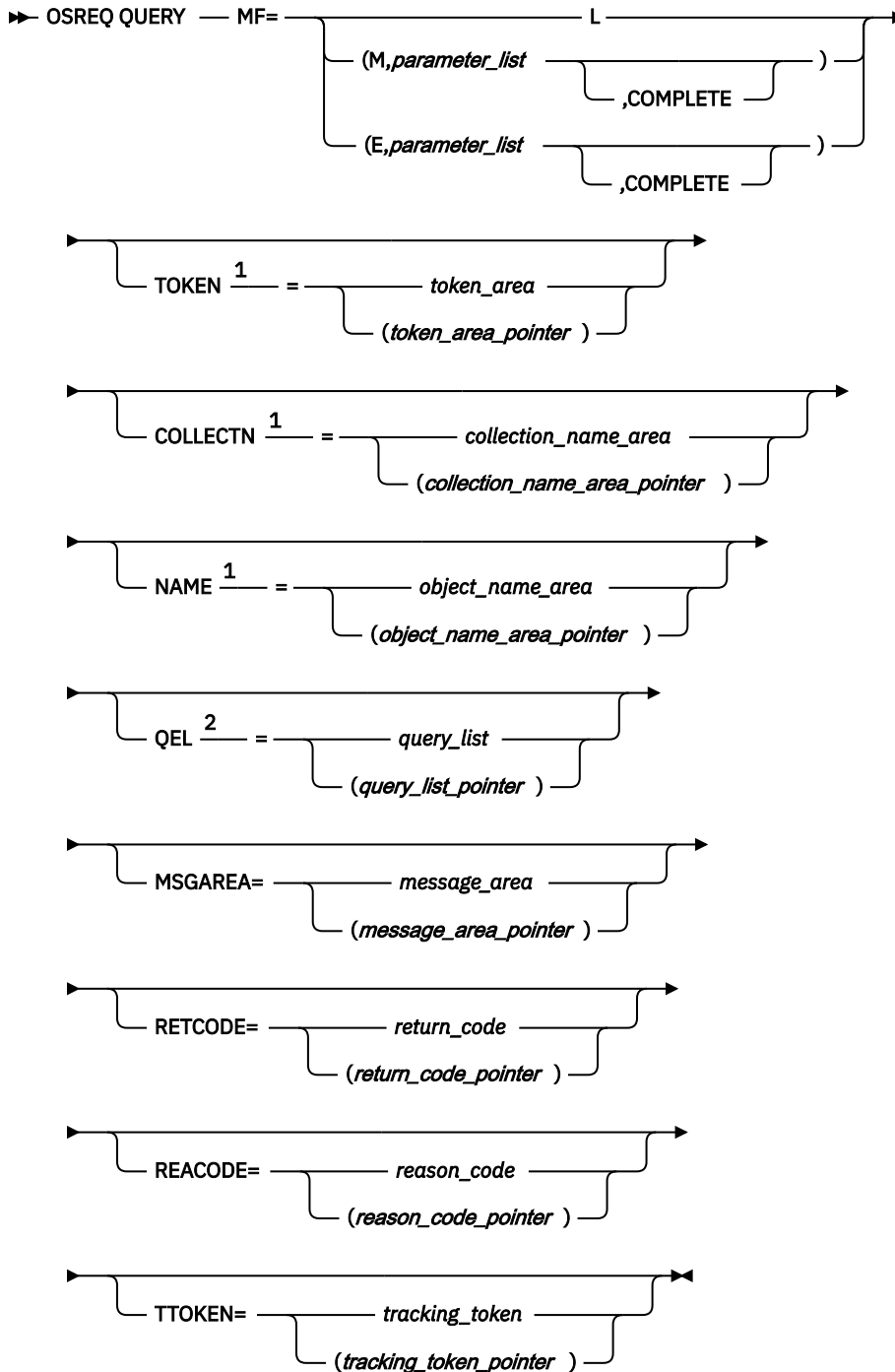
Request a generic search by one of the following methods:

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



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 “[#unique_40/unique_40_Connect_42_qelfi](#)” on page 39.

The output of a QUERY request can be used as input to a RETRIEVE request (see “[RETRIEVE—Retrieving an existing object](#)” on page 18).

RETRIEVE—Retrieving an existing object

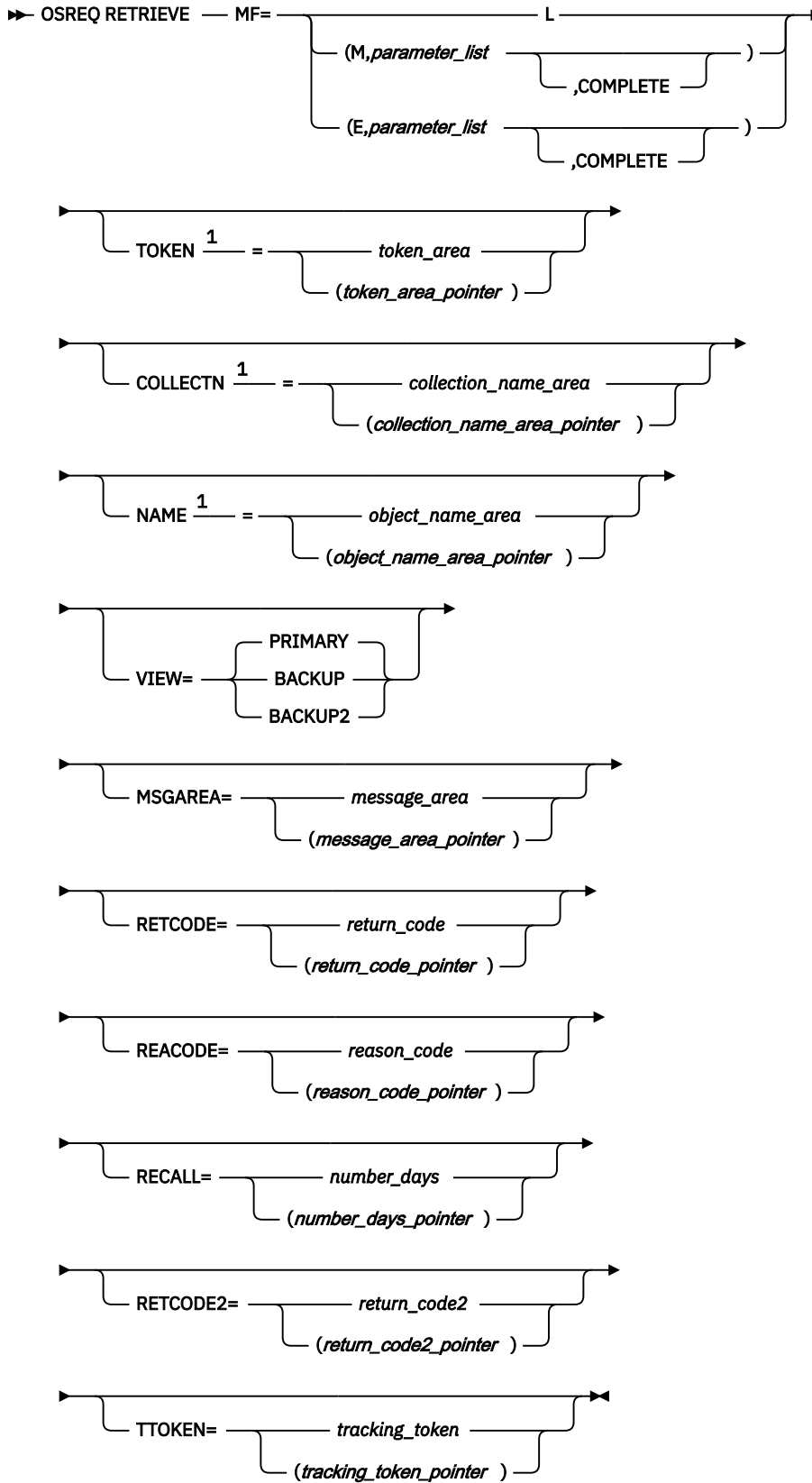
The RETRIEVE function locates the primary or backup copy of an object as specified by the COLLECTN, NAME, and VIEW keywords, and returns all or a specified portion of the object to the caller.

When using 31-bit addressable virtual storage buffers for object data, 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).

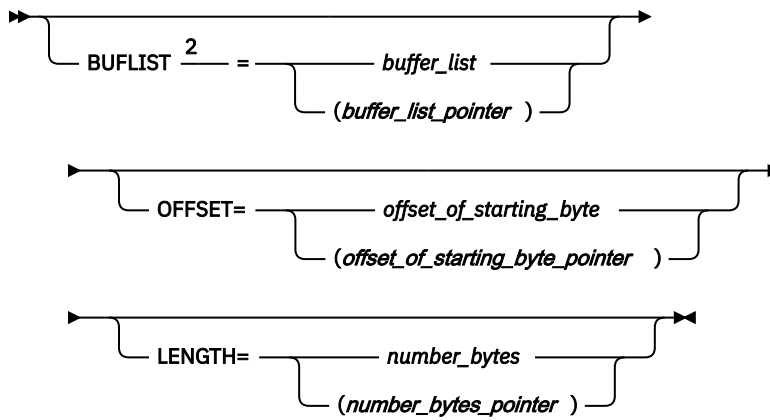
When using 64-bit addressable virtual storage buffers for object data, object data from 1 byte to the maximum object size (up to 2000M) can be retrieved using a single OSREQ RETRIEVE. Provide a 64-bit addressable virtual storage buffer equal to or larger than the length specified with the LENGTH64 parameter and optionally specify an offset using the OFFSET64 parameter.

The syntax diagram for the OSREQ RETRIEVE function follows.

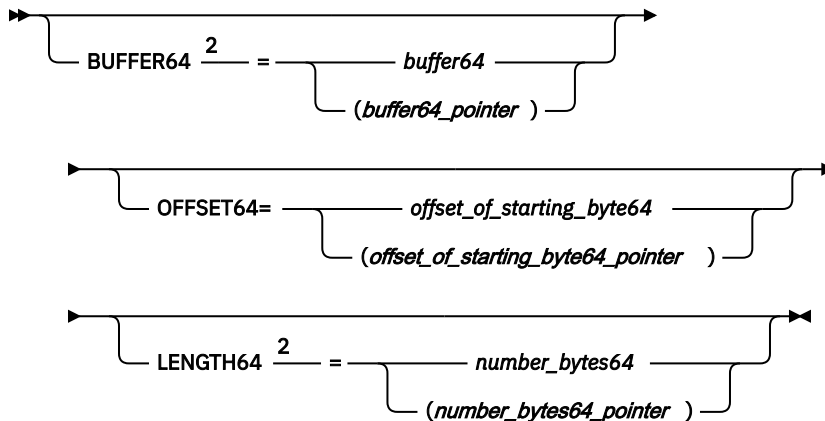
Syntax for OSREQ RETRIEVE



For 31-bit virtual address buffers



For 64-bit virtual address buffers



Notes:

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

² The keyword BUFLIST (when 31-bit addressable buffers are used) or keywords BUFFER64 and LENGTH64 (when 64-bit addressable buffers are used) 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 50](#).

You can retrieve a copy of the entire object (PRIMARY, BACKUP, or BACKUP2). Alternatively, you can retrieve a specified portion of the object by specifying a length and offset. 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).

The RECALL keyword can be used to explicitly recall a full copy of an object from optical, tape, or cloud to a Disk Sublevel (Db2 or File System) 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. RETRIEVE without implicit or explicit Recall might reset only the Last Reference Date. RETRIEVE with Recall, implicitly or explicitly invoked, could also set the Pending Action Date. Refer to [z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support](#) for more information on explicit and implicit recalls.

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. VIEW=PRIMARY is the default.

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. When VIEW=BACKUP is specified, OAM attempts to retrieve the first backup copy of the object from backup optical, tape, cloud, or file system. When VIEW=BACKUP2 is specified, OAM attempts to retrieve the second backup copy of the object from backup optical, tape, cloud, or file system.

If the specified VIEW function is requested but no object exists, return and reason codes reflect the error (see [OSREQ return and reason codes in z/OS DFSMSdfp Diagnosis](#)) and no data is retrieved into the user's buffers.

If OAM cannot successfully retrieve the primary copy of an object from the storage hierarchy and one or more backup copies exist, the Automatic Access Backup function can be used to automatically obtain a backup copy during an OSREQ RETRIEVE. The Automatic Access Backup function can be activated or deactivated by operator command or by SETOPT options in the CBROAMxx Parmlib member.

If there are problems retrieving the primary copy of an object, the OSMC Single Object Recovery utility, invoked by operator command, can be used to recreate a primary object copy from a backup copy. Upon successful completion of Single Object Recovery, you can again use OSREQ RETRIEVE to retrieve the primary copy of the object.

See [z/OS DFSMS OAM Planning, Installation, and Storage Administration Guide for Object Support](#) for more information on Automatic Access Backup or Single Object Recovery.

Adding objects to the object storage hierarchy

OAM provides multiple options for adding objects to the object storage hierarchy to address varied application virtual storage environments:

STORE function

The STORE function can be used with 31-bit addressable virtual storage buffers or 64-bit addressable virtual storage buffers. When using 31-bit addressable virtual storage buffers, the STORE function can be used for objects whose size is less than or equal to 256 megabytes (268,435,456 bytes) that are to be written to the disk, tape, optical, or cloud levels of the storage hierarchy. When using 64-bit addressable virtual storage buffers, the STORE function can be used for objects from 1 byte to 2000 megabytes (2,097,152,000 bytes) that are to be written to the disk, tape, or cloud levels of the storage hierarchy and for objects from 1 byte to 256 megabytes (268,435,456 bytes) that are to be written to the optical level of the storage hierarchy. STORE processing requires that the entire object exist in virtual storage. See [“STORE function” on page 23](#) for more information.

Store Sequence functions

Store Sequence can be used for objects whose size is greater than 50 megabytes (52,428,800 bytes) that are to be written to the disk, tape, or cloud (but not optical) levels of the storage hierarchy. Store Sequence processing handles objects in smaller parts, rather than having the entire object in storage (as required by STORE processing), which can reduce the virtual storage requirements for an application. See [“STOREBEG—Beginning a Store Sequence operation” on page 27](#), [“STOREPRT—Storing an individual part in a Store Sequence operation” on page 30](#), and [“STOREEND—Ending a Store Sequence operation” on page 32](#) for more information.

When storing objects in the Db2 sub-level, the LOB configuration must support the following cases when object data is always written to a LOB table:

- STORE with 64-bit addressable virtual storage buffers for objects greater than 256 megabytes (268,435,456 bytes) are always written to a LOB table
- Store Sequence processing always writes the objects to a LOB table.

Therefore, if LOB=N is specified on the OAMn entry in the IEFSSNxx parmliib member, or if a LOB storage structure does not exist for the target object storage group in these cases, then the attempt to do the STORE or Store Sequence to the Db2 sub-level fails.

[Table 2 on page 22](#) can be used to help decide which OAM store function is appropriate for the application virtual storage environment.

Table 2. Deciding which OAM store function to use

OSREQ Function	Virtual Storage Buffers	Minimum Object Size	Maximum Object Size	OAM Storage Hierarchy Location					When to Use
				DSL1 (Db2)	DSL2 (File System)	Tape	Optical	Cloud	
STORE with BUFLIST and SIZE	31-bit	1 byte	256M	Yes	Yes	Yes	Yes	Yes	Storing objects less than or equal to 256M in size and all object data can be made available in virtual storage at one time.
STORE with BUFFER64 and SIZE64	64-bit	1 byte	2000M ¹	Yes	Yes	Yes	Yes ¹	Yes	Storing objects greater than 256M, exploiting 64-bit virtual storage, and all object data is available in 64-bit virtual storage. Can also be used for objects less than or equal to 256M, but STORE with BUFLIST and SIZE is recommended for efficiency.

Table 2. Deciding which OAM store function to use (continued)									
OSREQ Function	Virtual Storage Buffers	Minimum Object Size	Maximum Object Size	OAM Storage Hierarchy Location					When to Use
				DSL1 (Db2)	DSL2 (File System)	Tape	Optical	Cloud	
STOREBEG, STOREPRT, and STOREEND	31-bit	Greater than 50M	2000M	Yes	Yes	Yes	No	Yes	Storing objects greater than 50M, but not exploiting 64-bit virtual storage and all object data can not be made available in virtual storage at one time.

Note: Although 2000M is the maximum size supported by OAM, the actual maximum is installation dependent and might be less than 2000M.

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.

When using 31-bit addressable virtual storage buffers for object data, 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 50 megabytes.

When using 64-bit addressable virtual storage buffers for object data, you can use STORE for objects from 1 byte to the maximum configured object size (up to 2000M).

Objects are stored on an object storage device based on storage class. Objects are removed from the object storage hierarchy based on management class expiration attributes or after their expiration date. 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](#).

When using 31-bit addressable virtual storage buffers for object data, the number of bytes specified in the SIZE parameter are written to an object storage device from the buffers specified in the BUFLIST parameter. When using 64-bit addressable virtual storage buffers, the number of bytes specified in the SIZE64 parameter are written to an object storage device from the buffer specified in the BUFFER64 parameter.

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:

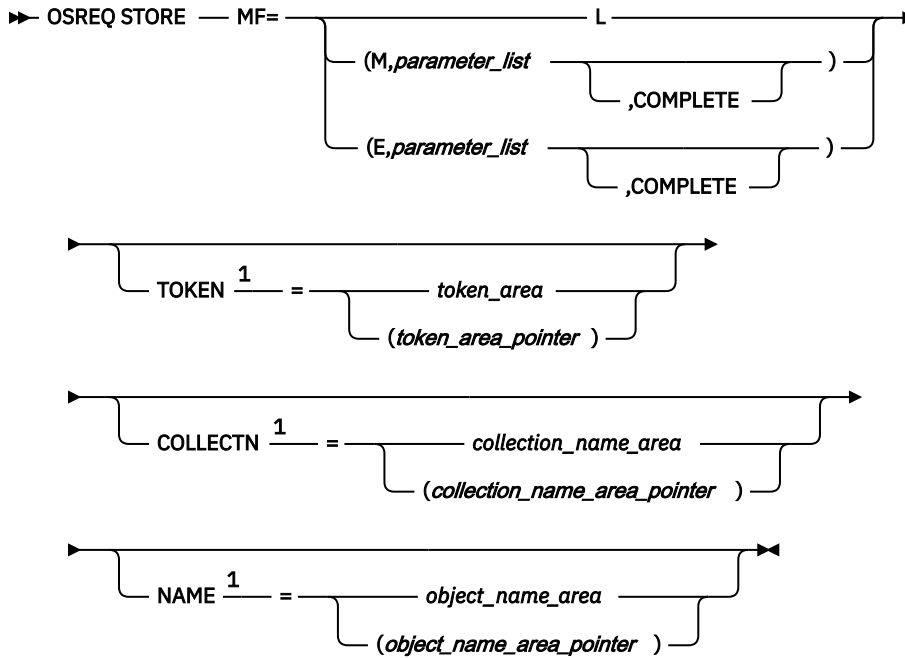
¹ The maximum size for optical is 256M

- 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'FFFFFFFFE'), 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 3 on page 47](#).
- 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 3 on page 47](#) 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.

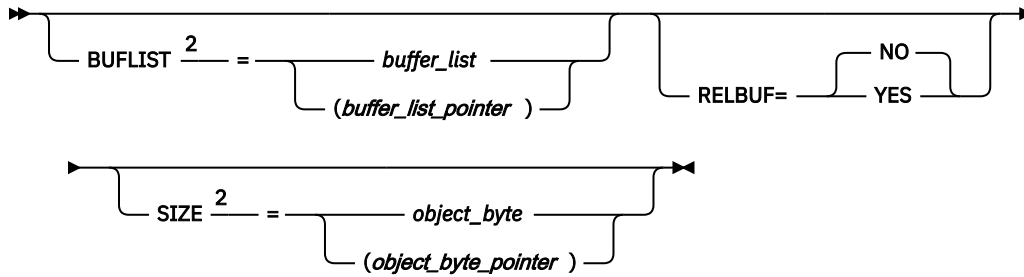
See [“Expiration date processing” on page 47](#) for more information.

- 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'FFFFFFFFE'), 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.

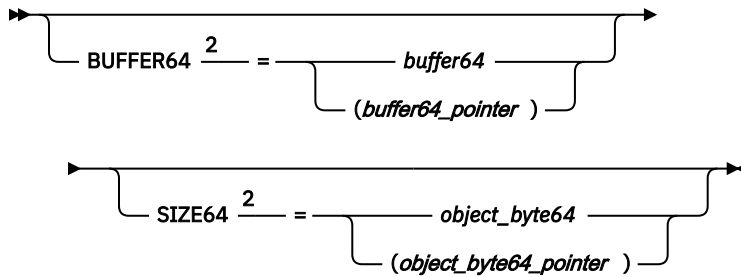
Syntax for OSREQ STORE

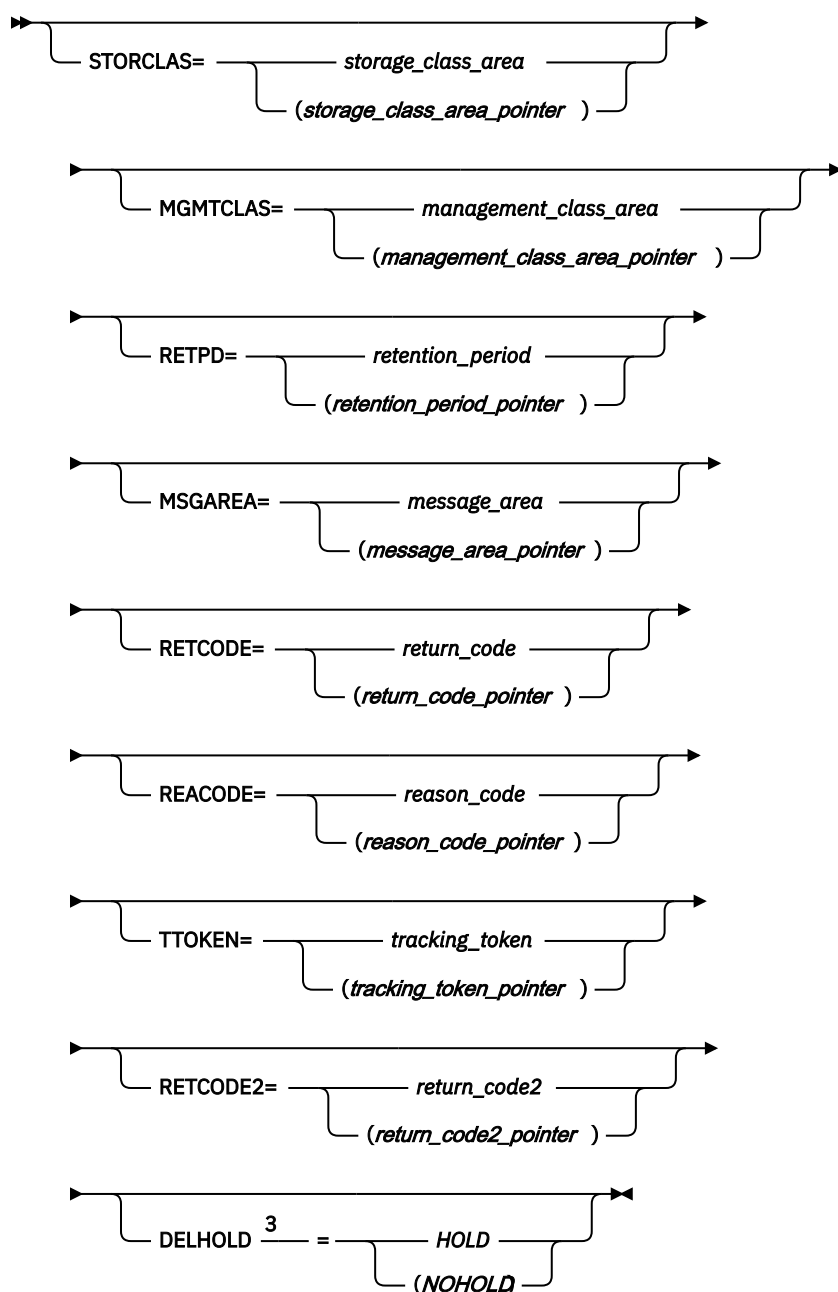


For 31-bit virtual address buffers:



For 64-bit virtual address buffers:





Notes:

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

² The keywords BUFLIST and SIZE (when 31-bit addressable buffers are used) or BUFFER64 and SIZE64 (when 64-bit addressable buffers are used) 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 50](#).

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 and the OSREQ STOREBEG request.

If the OSREQ Store request specifies a new collection name, a new collection entry in the Db2 Collection Name Table is created for the collection. The Db2 Collection Name Table entry contains the names of the

management and storage classes 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 determine 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 after the new collection name entry in the Db2 Collection Name Table 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 that are contained in the Db2 Collection Name Table 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 that are 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 that is 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 26](#) and [“Processing a store to an existing collection” on page 27](#). 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 Db2 Collection Name Table or ACS routines during OSREQ STOREPRT or OSREQ STOREEND requests.

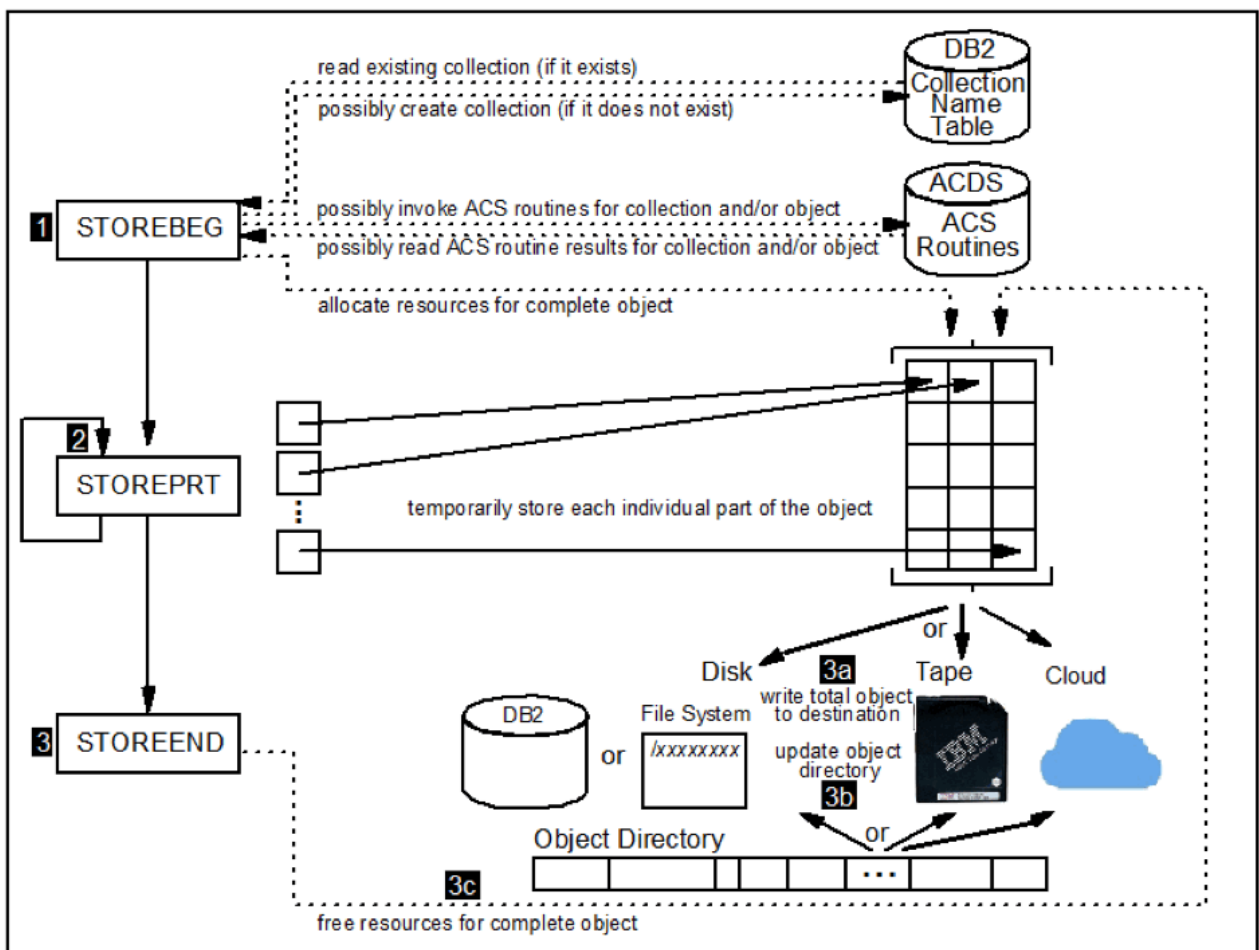
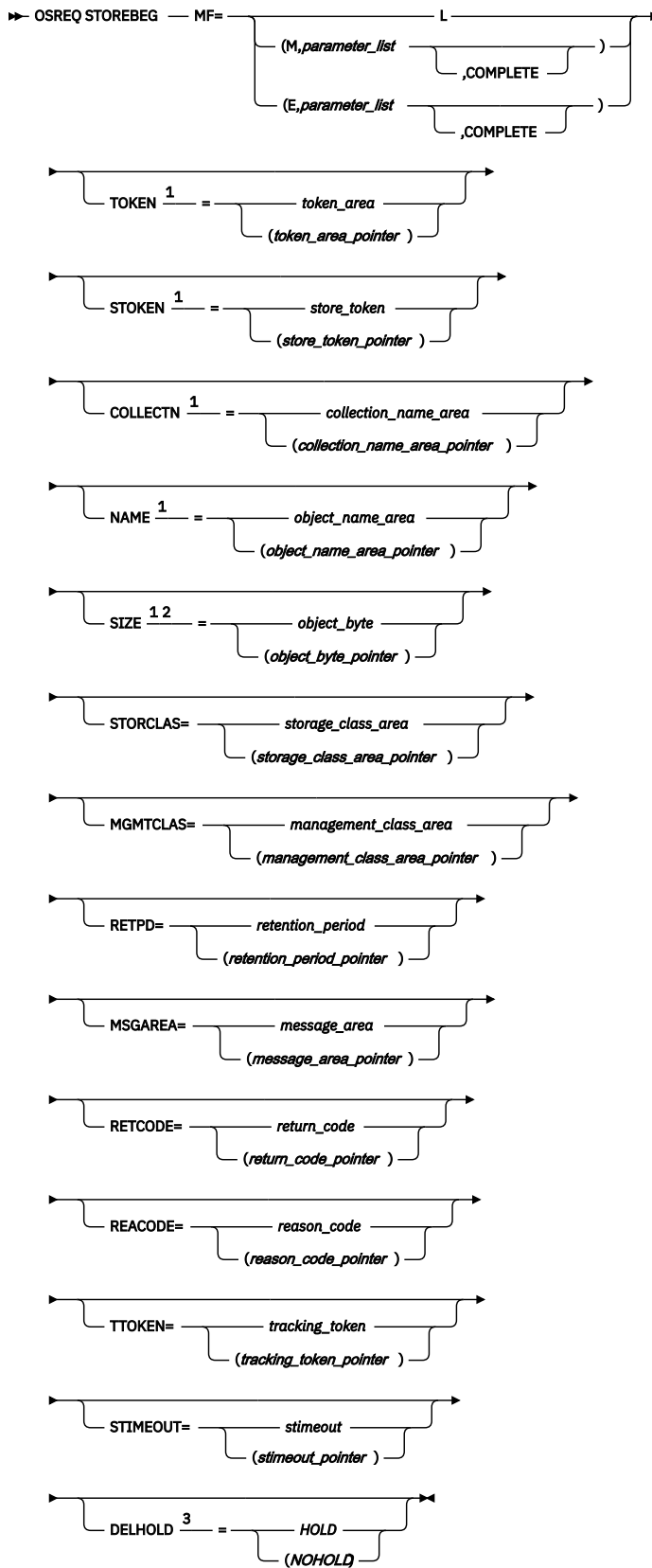


Figure 2. Conceptual view of a Store Sequence operation

Syntax for OSREQ STOREBEG



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 that is 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.

```
sequenceDiagram
    participant OSREQ as OSREQ
    participant STOREPRT as STOREPRT
    OSREQ->>STOREPRT: MF=
    STOREPRT-->>OSREQ: (M,parameter_list),COMPLETE
    STOREPRT-->>OSREQ: (E,parameter_list),COMPLETE
    OSREQ->>STOREPRT: TOKEN 1 =
    STOREPRT-->>OSREQ: token_area,token_area_pointer
    OSREQ->>STOREPRT: STOKEN 1 =
    STOREPRT-->>OSREQ: store_token,store_token_pointer
    OSREQ->>STOREPRT: SIZE 1 2 =
    STOREPRT-->>OSREQ: object_byte,object_byte_pointer
    OSREQ->>STOREPRT: OFFSET 1 3 =
    STOREPRT-->>OSREQ: offset_of_starting_byte,offset_of_starting_byte_pointer
    OSREQ->>STOREPRT: BUFLIST 1 4 =
    STOREPRT-->>OSREQ: buffer_list,buffer_list_pointer
    OSREQ->>STOREPRT: RELBUF=
    STOREPRT-->>OSREQ: YES,NO
    OSREQ->>STOREPRT: MSGAREA=
    STOREPRT-->>OSREQ: message_area,message_area_pointer
    OSREQ->>STOREPRT: RETCODE=
    STOREPRT-->>OSREQ: return_code,return_code_pointer
    OSREQ->>STOREPRT: REACODE=
    STOREPRT-->>OSREQ: reason_code,reason_code_pointer
    OSREQ->>STOREPRT: TTOKEN=
    STOREPRT-->>OSREQ: tracking_token,tracking_token_pointer
```

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.

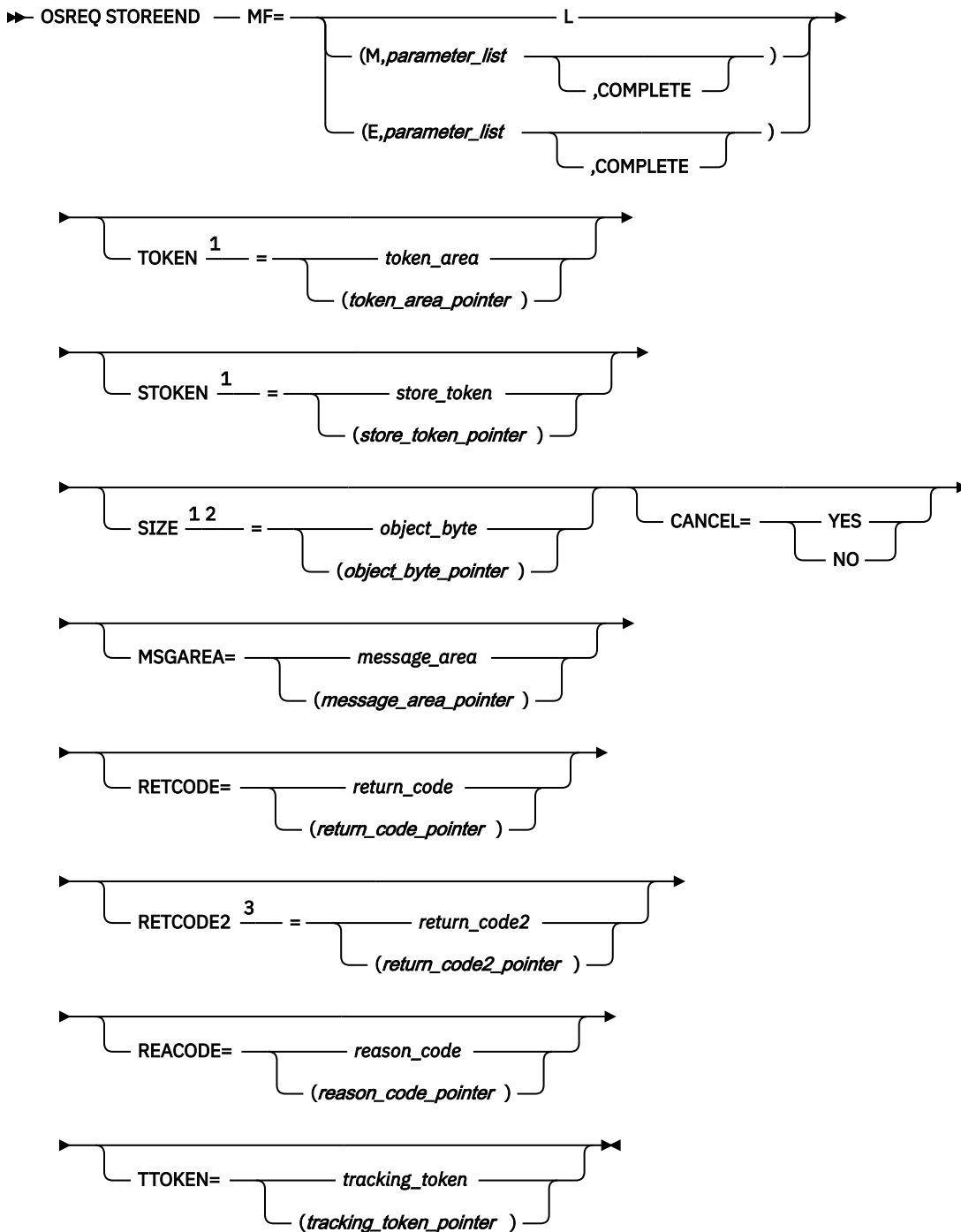
³ The offset must be zero for the first part stored for the object, and for each subsequent store you must identify the offset of the next byte immediately following the previous part stored for the object (that is, the sum of the offset and size for the previous part stored).

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



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.

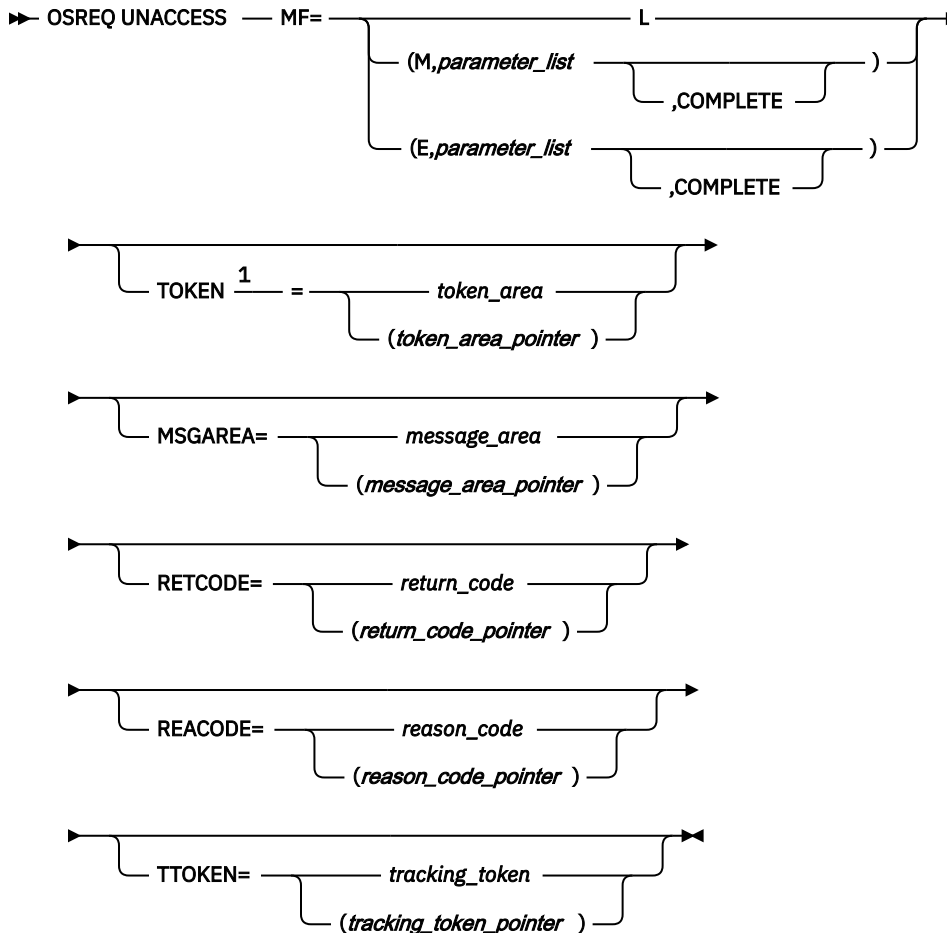
² The size specifies the total size of the object to be stored. Note that when specifying the total size of the object, it must match the total size specified on the STOREBEG and that exactly this amount of object data must have been previously provided with one or more STOREPRT requests for the object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



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.

BUFFER64=buffer64
BUFFER64=(buffer64_pointer)

The BUFFER64 keyword is used on STORE and RETRIEVE requests to specify a 64-bit virtual storage buffer address. *buffer64* is an 8-byte field that can be used to identify a 64-bit addressable virtual storage area above the 2-G bar. Although a 31-bit value can be specified for *buffer64*, this is not recommended as this will reduce the efficiency of the OSREQ request and instead the BUFLIST keyword should be used.

For an OSREQ STORE request, use of the BUFFER64 keyword requires the SIZE64 keyword. On an OSREQ STORE request, the BUFFER64 keyword is mutually exclusive with the BUFLIST, RELBUF, and SIZE keywords.

For an OSREQ RETRIEVE request, use of the BUFFER64 keyword requires the LENGTH64 keyword. When an optional offset is specified, the OFFSET64 keyword must be used. On an OSREQ RETRIEVE request, the BUFFER64 keyword is mutually exclusive with the BUFLIST, LENGTH, and OFFSET keywords.

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 49.

On an OSREQ STORE, the BUFLIST keyword is mutually exclusive with the BUFFER64 and SIZE64 keywords. On an OSREQ RETRIEVE, the BUFLIST keyword is mutually exclusive with the BUFFER64, LENGTH64, and OFFSET64 keywords.

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 canceled. 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 canceled, 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).

DB2ID=Db2_subsystem_id
DB2ID=(Db2_subsystem_id_pointer)

Db2_subsystem_id specifies a variable length area. This area contains the identification of a Db2 subsystem (Db2 SSID or Db2 group attachment name, if the Db2 subsystem is part of a data sharing group) to be used for processing this request. The first two bytes specify the number of characters that follow; the Db2 SSID or group attachment name value can be from 1-4 characters.

The DB2ID parameter is only intended for use in a multiple OAM configuration. Note that the system administrator must have configured an OAM subsystem and, when applicable, an OAM address space to interact with the specified Db2 subsystem in a multiple OAM configuration.

For OSREQ applications that run in the DSN environment and also when the IADDRESS keyword is specified DB2ID is not required; in all other cases in a multiple OAM configuration the specification of

DB2ID is required. If DB2ID is specified when running in the DSN environment or when the IADDRESS keyword is specified, it will be accepted if it matches the SSID or group attachment name of the already connected Db2 subsystem. If it does not match, the OSREQ request will fail.

For a Multiple OAM configuration with only one object subsystem defined, the DB2ID is not required. If a DB2ID is specified, it will be accepted if it matches the SSID or group attachment name of the already connected Db2 subsystem. If it does not match, the OSREQ request will fail.

For a Multiple OAM configuration with more than one object subsystem defined, the DB2ID is required.

If DB2ID is specified in a classic OAM configuration, it will be accepted if it matches the SSID or group attachment name of the Db2 subsystem specified with the DB2SSID keyword in the IGDSMSxx member of PARMLIB (or later by the operator if the DB2SSID keyword was not specified). If it does not match, the OSREQ request will fail.

Applications built on the current release that specify DB2ID can be used on earlier releases, but the DB2ID specification will not be used on any system running a release prior to V2R3.

The DB2ID keyword is used only on an ACCESS request.

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:

- ODETRDT is set to whichever is later; the newly calculated ODEXPDT or the current ODETRDT.
- ODEXPDT is set to whichever is later; the ODETRDT 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. The connection could have been made using either the Db2 SSID or Group Attachment Name.

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.

The LENGTH keyword is mutually exclusive with the BUFFER64, LENGTH64, and OFFSET64 keywords.

LENGTH64=*number_bytes64*

LENGTH64=(*number_bytes64_pointer*)

The LENGTH64 keyword is used on RETRIEVE requests and is required when the BUFFER64 keyword has been specified. *number_bytes64* is an 8 byte field to identify the length of the object data, in bytes, to be retrieved into the 64-bit addressable virtual storage buffer. LENGTH64 can be used with the OFFSET64 keyword to retrieve part of an object. The LENGTH64 keyword is mutually exclusive with the BUFLIST, LENGTH, and OFFSET keywords.

The minimum value for *number_bytes64* is 1. The maximum value for *number_bytes64* is the total size of the object. The total size of the object may be obtained with an OSREQ QUERY request. If the length specified is negative, or greater than the remaining portion of the object, or greater than 2000 megabytes (2 097 152 000 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).

The OFFSET keyword is mutually exclusive with the BUFFER64, LENGTH64, and OFFSET64 keywords.

OFFSET64=offset_of_starting_byte64
OFFSET64=(offset_of_starting_byte64_pointer)

The OFFSET64 keyword is used on RETRIEVE requests, is optional, and can only be specified when the BUFFER64 and LENGTH64 keywords have been specified. *offset_of_starting_byte64* is an 8 byte field to identify the offset of the first byte to be retrieved into the 64-bit addressable virtual storage buffer. The first byte of the object has an offset of zero, the second byte has an offset of one, and so on. If the OFFSET64 parameter is omitted on a RETRIEVE request that specifies the BUFFER64 and LENGTH64 keywords, the offset defaults to the beginning of the object (that is, OFFSET64=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. The OFFSET64 keyword is mutually exclusive with the BUFLIST, LENGTH, and OFFSET keywords.

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 51](#).

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 CBROAMxx parmlib member.
3. The MAXRECALLTASKS must be set to a non-zero value in a SETOSMC statement in the CBROAMxx 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.

The RELBUF keyword is mutually exclusive with the BUFFER64 and SIZE64 keywords.

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 CBROAMxx 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 CBROAMxx parmlib member
12	An attempt to schedule a RECALL was not successful because RECALLOFF(ON) was specified in the CBROAMxx parmlib member
14	An attempt to schedule a RECALL was not successful because of a scheduling error
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 3 on page 47](#) 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 IEFSSNxx 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 IEFSSNxx 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.

The SIZE keyword is mutually exclusive with the BUFFER64 and SIZE64 keywords.

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

SIZE64=object_byte64
SIZE64=(object_byte64_pointer)

The SIZE64 keyword is used on STORE requests and is required when the BUFFER64 keyword has been specified. *object_byte64* is an 8 byte field to identify the size of the object data, in bytes, within the 64-bit addressable virtual storage buffer. The SIZE64 keyword is mutually exclusive with the BUFLIST, RELBUF, SIZE keywords.

The minimum value for *object_byte64* is 1. The maximum value for *object_byte64* is installation dependent. The MOS=nnnn parameter in the IEFSSNxx 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 the MOS=nnnn parameter in the IEFSSNxx parmlib member for more information on object sizes greater than 50 megabytes.

STIMEOUT=timeout
STIMEOUT=(timeout_pointer)

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

The *timeout* 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.

TOKEN=*tracking_token*

TOKEN=(*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 TTKEN 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 [*OSREQ return and reason codes in z/OS DFSMSdfp Diagnosis*](#)); 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.
 - In the CICS environment, CICS APIs do not directly provide a 64-bit virtual storage address to CICS applications and because the use of non-CICS APIs (for example, to acquire 64-bit addressable virtual storage) is not supported by CICS, it is recommended that CICS applications do not implement the use of OSREQ 64-bit addressable virtual storage buffers for OSREQ STORE and OSREQ RETRIEVE functions.

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 [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims).
- In the CICS, DSN Command Processor, IMS and Db2 stored procedure 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 [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims)). In the TSO environment, synchronization is accomplished through the use of COMMIT and ROLLBACK functions, as described in [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims). In the IMS environment, synchronization is accomplished through the use of COMMIT and ROLLBACK functions (see [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims)), or by the use of SYNC and ROLL/B call to IMS. Synchronization in a Db2 stored procedure environment is accomplished through the use of RRSAF COMMIT and ROLLBACK functions, as described in the Db2 for z/OS reference, "Invoking the Resource Recovery Services attachment facility" section.
- 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.
- The OSREQ functionality is not backward compatible. Therefore, when new OSREQ functionality is exploited through the OSREQ macro it can be exploited only on a z/OS system with a version/release at or higher than the z/OS version/release in which the new OSREQ functionality is supported. For example, 64-bit address buffers are supported only on systems at z/OS V2R2 or higher.

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 11 in "ACCESS—Initializing the OSREQ interface" on page 9](#) have not been tested by IBM and the results may be unpredictable.

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 IEFSSNxx 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 might be relevant as you code your application interface:

- Optional input parameters on the OSREQ macro might 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 that is 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 34](#).
- If you do not specify a collection name on any function other than ACCESS, 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 Db2 Collection Name Table for any function other than STORE, STOREBEG, ACCESS, or UNACCESS, a return code and a reason code are generated.
- When a Db2 Collection Name Table 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 that are stored in the file system or cloud (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 that is 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 non addressable 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 words are not located in addressable storage, the return and reason codes are only found in general registers 15 and 0, 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'FFFFFFF') 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 3 on page 47 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 3. 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'7FFFFFFF' (-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.

Table 3. Valid Retention Periods for Expiration Date Processing (continued)

Specified RETPD Parameter Value	Requested Expiration Date STORE	Requested Expiration Date CHANGE
X'FFFFFFFFE' (-2)	Set expiration date to 0002-02-02 and set indicator in ODSSTATF 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= <i>number_of_days</i> .	Set expiration date to 0002-02-02 and set indicator in ODSSTATF 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= <i>number_of_days</i> .
1 to 93,000	<p>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 ODEXPD is set to sum of the object create date + RETPD value.</p> <p>For retention-protected objects:</p> <ul style="list-style-type: none"> the ODRETD is set to whichever is later; the newly calculated ODEXPD or the current ODRETD. the ODEXPD is set to whichever is later; the ODRETD or the ODEXPD. 	<p>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 ODEXPD is set to sum of the object create date + RETPD value.</p> <p>For retention-protected objects:</p> <ul style="list-style-type: none"> the ODRETD is set to whichever is later; the newly calculated ODEXPD or the current ODRETD. the ODEXPD is set to whichever is later; the ODRETD or the ODEXPD.
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 [OSREQ return and reason codes in z/OS DFSMSdfp Diagnosis](#) for a table of return codes and their associated reason codes. You can also use the following commands to display the OSREQ reason code information:

- The operator command D OAM,OSREQRC *wwwxyyzz*.

- The TSO IVP command OAMUTIL OSREQRC *wwwxyyzz*.

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 on page 50](#) and [Figure 4 on page 51](#) describe the contents of the DSECTs.

Figure 3. Fields Described by CBRIBUFL

OBL	DSECT	Data buffer list control block
+0 OBLID	DS 0F	Control block identifier ('OBL ')
+4 OBLSTL	DS CL4	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
+0 OBLBDATA	DS 0F	Object data area

Figure 3. Fields Described by CBRIBUFL

Figure 4 on page 51 is a structure diagram of the data buffer list (CBRIBUFL) pointed to by the BUFLIST keyword on an OSREQ STORE or OSREQ RETRIEVE macro.

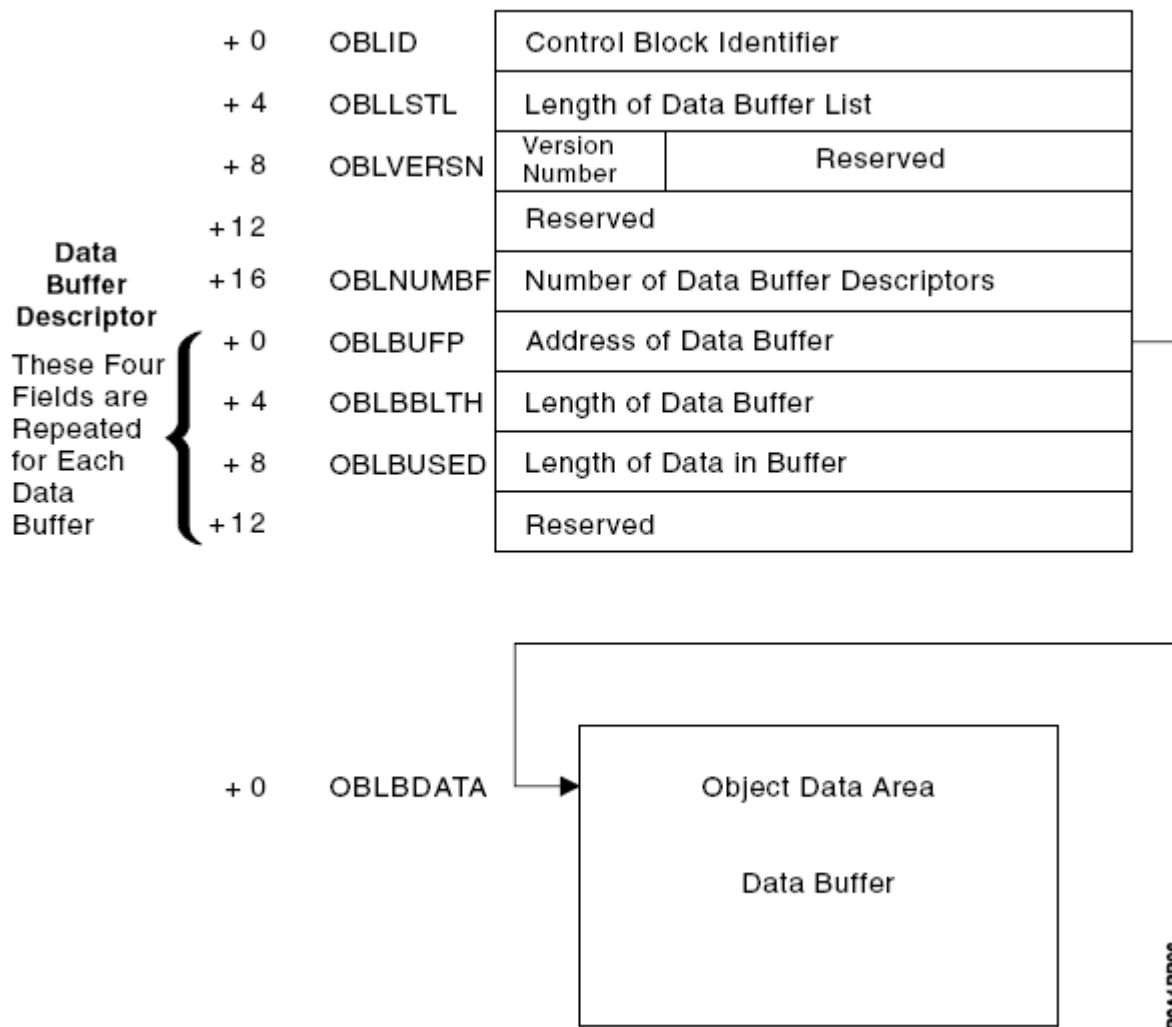


Figure 4. Data Buffer List Structure Diagram

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 that is 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 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 with the QELB DSECT to map one query element in the query buffer. [Figure 5 on page 53](#) and [Figure 4 on page 51](#) 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 might use the QELQB2OK field on the CBRIQEL mapping macro, which sorts objects before 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 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 contains binary zeros. Similarly, if a backup or secondary backup copy of the object does not exist or a backup or secondary backup copy of the object resides on cloud or file system, then the corresponding backup or second backup retrieval order key contains binary zeros. Also, if the QB= keyword in the IEFSSNxx parmlib member is set to QB=N, then the OAM address space is not invoked to obtain any existing backup retrieval order keys. This results in the backup and second backup retrieval order keys containing binary zeros.

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

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

QELBDESC	DSECT		Query buffer descriptor
+0 QELBUFP	DS A		Address of query buffer
+4 QELBBLTH	DS F		Length of query buffer
+8 QELBUSED	DS F		Number of bytes returned in query buffer
+12 QELBRVS1	DS F		Reserved, must be zero

Each query buffer is described as follows:

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

Each query element is described by the following:

QELQ	DSECT	Query element	
+0 QELQELE	DS H		QE length including this field
+2 QELQECD	DS CL10		Creation date (yyyy-mm-dd)
+12 QELQEDH	DS CL1		Set to '-'
+13 QELQECT	DS CL15		Creation time (hh.mm.ss.nnnnnn)
+28 QELQELD	DS CL10		Last referenced date (yyyy-mm-dd)
+38 QELQEED	DS CL10		Expiration date (yyyy-mm-dd)
+48 QELQESC	DS XL2,CL8		Storage class length and name
+48 QELQESCL	EQU QELQESC,2		Storage class length
+50 QELQESCN	EQU QELQESCL+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 QELQBKLOC	DS CL1		Backup1 location
+267 QELQBK2LOC	DS CL1		Backup2 location
+267 QELQB_TAPE_OPT	EQU C'M'		On removable media
+267 QELQB_CLOUD	EQU C'C'		On Cloud
+267 QELQB_FS	EQU C'E'		On File System
+267 QELQB_NONE	EQU C' '		No backup copy

Figure 5. Fields Described by CBRIQEL

The QELVERSN and QELQELE fields must be set by the user. 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, QELQESF, QELQELF, and QELQEDR fields. The backup retrieval order key fields contain binary zeros, and the backup1 and backup2 location fields contain blanks, 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 zeros 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 zeros 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, therefore, 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).

10 000

Object resides on cloud storage.

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 that is 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 cloud level retrieval response time can vary significantly and depends on many factors. Some of these include the size of the object, the underlying physical storage medium on which the object resides in the cloud, network bandwidth, and current usage level. The estimated retrieval response time is intended to provide only a comparative response time relative to the other OAM storage hierarchy targets for objects.

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 4 on page 51 is a structure diagram of the query buffer list (CBRIQEL) pointed to by the QEL keyword on an OSREQ QUERY macro:

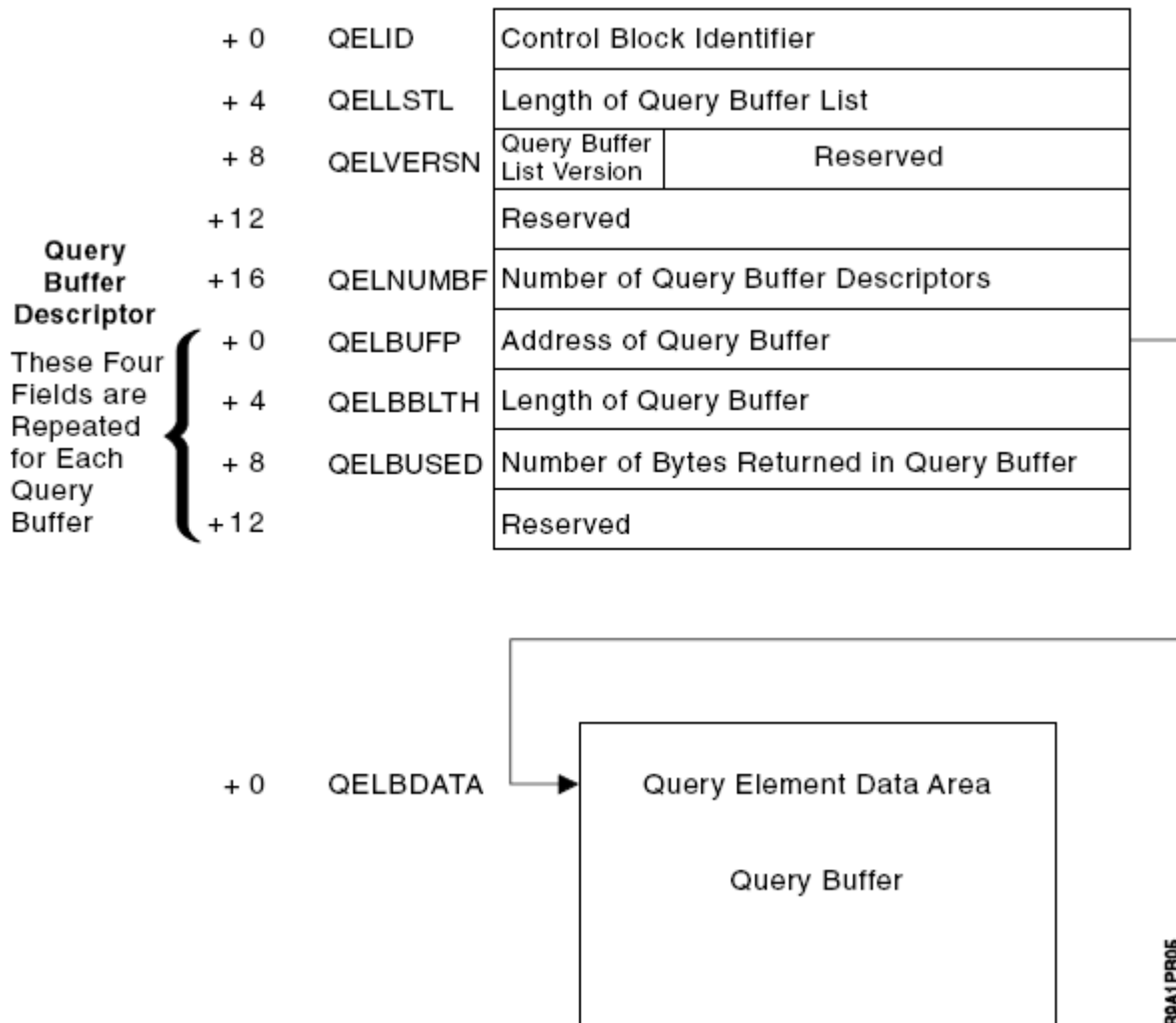


Figure 6. Query Buffer List Structure Diagram

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) might 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 that are 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 that is 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 retrieved or changed, or if the UPD=N parameter was specified on the CBRINIT statement of the IEFSSNxx parmlib member that was used during IPL, the last date that is referenced is "0001-01-01". A last date that is 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-align in the field that is next 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 sample programs that use the OSREQ macro for object manipulation. These programs are available as members CBROSREQ, CBROSR2, CBROSR3, and CBROSRSP in SAMPLIB.

General notes:

1. You can link-edit these members as part of the application load module. You do not need to issue a LOAD request before using the OSREQ calls.
2. These members are fully operational as shipped and can be used without modification to support application programs written in PL/1 or COBOL.
3. You can modify these members as necessary to support applications written in high-level languages other than PL/1 or COBOL.
4. 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 the CBROSREQ, CBROSR2, and CBROSR3 members. For example:

```
//ASSEMBLE EXEC PGM=ASMA90,PARM='RENT,DECK,SYSPARM(IADD)'
```

Note: The CBROSRSP member uses the IADDRESS parameter by default therefore the SYSPARM(IADD) is not used nor accepted.

5. ***Important*** All of these members expect the calling program to GETMAIN storage for the size of the DATAAREA mapping contained within each member. This DATAAREA must be mapped according to the member used and the appropriate fields filled in for the intended function requested. The address pointing to this DATAAREA must be loaded into register one before calling a member. This address is then used to map the GETMAINED storage to the DATAAREA mapping within the member for the data contained therein to become addressable to the member. For additional information regarding what fields to fill in within the DATAAREA pertaining to a function request, please refer to the comments contained within the member being used.

Note: Various fields within the DATAAREA must not be modified by the calling program as the DATAAREA storage passed to the member is intended to be reused across multiple invocations. Do not alter these fields. Please refer to the DATAAREA comments within each member for details on which fields should or should not be altered.

Notes regarding each member:

CBROSREQ “Generalized interface for the OSREQ macro”

1. This interface can issue basic OSREQ requests such as ACCESS, STORE, RETRIEVE, QUERY, CHANGE, DELETE, and UNACCESS.
2. This interface can also issue COMMITs or ROLLBACKs if needed.

Note: You must run the Db2 pre-compiler due to the EXEC SQL statement in the code for the CBROSREQ sample.

3. This sample uses the DSNHLI entry point for the OSREQ IADDRESS parameter if SYSPARM=IADD.

CBROSR2 “Generalized Interface for the OSREQ macro plus support for Store Sequence and Buffer 64”

1. This interface can issue all the OSREQ functions within CBROSREQ plus additional functions like STOREBEG, STOREPRT, STOREEND, and buffer 64 related OSREQ requests.
2. This interface does not issue COMMITs or ROLLBACKs nor executes any Db2 related functions (such as EXEC SQL), therefore a Db2 pre-compiler is not needed.
3. This sample uses the IADDRESS_PTR field located in the caller supplied DATAAREA for the OSREQ IADDRESS parameter.

CBROSR3 “Generalized Interface for the OSREQ macro plus support for the DB2ID keyword”

1. This interface can issue all the OSREQ functions within CBROSR2 plus supports the use of the DB2ID keyword on OSREQ ACCESS for Multiple OAM configurations.
2. This interface does not issue COMMITs or ROLLBACKs nor executes any Db2 related functions (such as EXEC SQL), therefore a Db2 pre-compiler is not needed.
3. This sample uses the IADDRESS_PTR field located in the caller supplied DATAAREA for the OSREQ IADDRESS parameter.

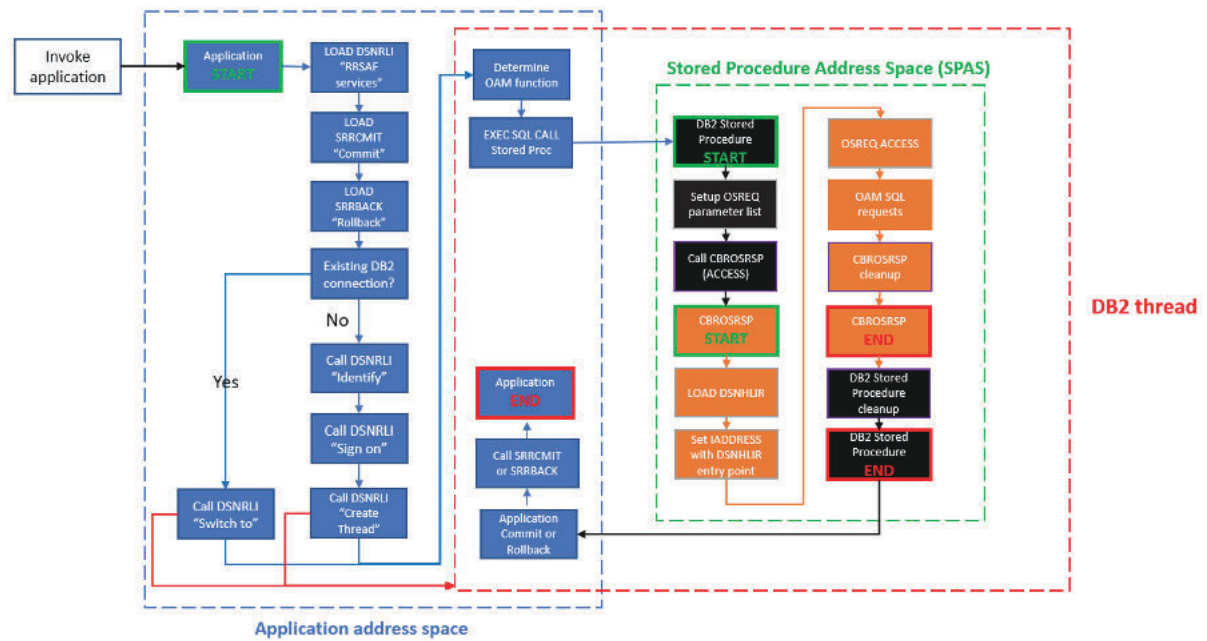
CBROSRSP “Generalized Interface for the OSREQ macro within a Db2 Stored Procedure environment”

1. This interface can issue all the OSREQ functions within CBROSR2 and exclusively runs within a Db2 stored procedure environment.
2. This interface does not issue COMMITs or ROLLBACKs nor executes any Db2 related functions (such as EXEC SQL), therefore a Db2 pre-compiler is not needed.
3. This sample uses the DSNHLIR entry point for the OSREQ IADDRESS parameter.

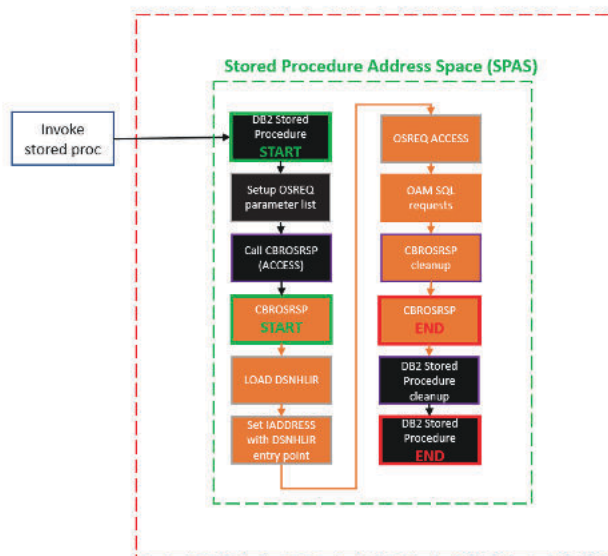
Note: Please refer to Db2 “Application Programming and SQL Guide” and “z/OS Stored Procedures: Through the Call and Beyond” for more information about latest RRSAF overall usage, restrictions and benefits to using Db2 stored procedure address spaces.

Example Db2 stored procedure environment flow of an OSREQ ACCESS for both explicit and implicit RRSAF connections:

Explicit RRSF Connection Example (OSREQ ACCESS)



Implicit RRSF Connection Example (OSREQ ACCESS)



Based on invocation and linkage conventions, DB2 will implicitly perform RRSF IDENTIFY and CREATE THREAD connection processing before stored procedure gains control.

OAM exhibits the same behavior as it does for an explicit connection.

CBROSREQ

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

```
***** 00050000
*      00100000
* DESCRIPTIVE NAME: Object Storage Request Sample interface 00150000
*      00150100
* Licensed Materials - Property of IBM 00150200
* 5650-ZOS 00150300
* COPYRIGHT IBM CORP. 1989, 2017 00150400
*      00200000
* FUNCTION: Provides a generalized interface for the Object Storage 00250000
* Request (OSREQ) macro. 00300000
*      00307100
```

```

*      This interface includes support to perform a DB2 CAF      00314200
*      sync (commit) or DB2 CAF abort (rollback).              @L1A 00321300
*                                                                00328400
*      This interface does not support specifying DB2ID on an @P2A 00328800
*      ACCESS request (needed in many cases for a multiple    @P2A 00333500
*      OAM configuration) and does not support the following  @P2A 00334000
*      OSREQ functions: STOREBEG, STOREPRT, and STOREEND.     @P2C 00339100
*      Please see sample job CBR0SR3 for support of the DB2ID @P2C 00342700
*      keyword on an ACCESS request and the OSREQ             @P2A 00344000
*      STOREBEG, STOREPRT, and STOREEND functions.            @L1A 00346300
*                                                                00350000
* OPERATION: This routine is called with a parameter area that 00400000
* defines the function and pointers necessary to invoke        00450000
* the OSREQ macro and/or synchronize the data bases that      00500000
* are connected to the current DB2 thread.                    00550000
* If it is determined that an OSREQ function is requested,    00600000
* then the OSREQ parameter list is filled in with an          00650000
* MF=M form of the macro. The function is executed via an     00700000
* MF=E form.                                                  00750000
* A call is made to an internal routine which will            00800000
* determine the need to synchronize the data bases.           00850000
* If sync has been requested and the value in the             00900000
* field pointed to by the RETURN_CODE_PTR                     00950000
* field is 0 or 4 then DB2 will be notified                   01000000
* to commit all changes made to the data bases                01050000
* since the last synchronization point.                       01100000
* If sync has been requested and the value in the             01150000
* field pointed to by the RETURN_CODE_PTR                     01200000
* field is greater than 4, DB2 will be                        01250000
* notified to rollback all changes made to the data           01300000
* bases since the last synchronization point.                 01350000
*                                                                01400000
* DB2 SYNC and ROLLBACK Notes:                                @L1A 01400900
* This sample is setup to assume the MVS batch environment.   01401800
* Changes related to executing the DB2 SYNC and ROLLBACK      01402700
* functions will need to be made for other environments.      01403600
* For example in a CICS environment, EXEC CICS SYNCPOINT      01404500
* would need to be performed instead of calling DSNALI to do  01405400
* a CAF CLOSE.                                                @L1A 01406300
*                                                                01407200
* If this sample is NOT compiled with the IADD SYSPARM or a   01407800
* a DB2 connection is not already established, then          01408400
* a DB2 connection or thread will be established by OAM       01409000
* performing a CAF OPEN during the OSREQ ACCESS request. If   01409900
* SYNC in the DATAAREA equals "YES", then a CAF CLOSE is used 01410800
* to perform either a DB2 sync or rollback. At this point the 01411700
* applications DB2 thread will be closed. To reopen this      01412600
* thread, this sample will perform a CAF OPEN. The values of   01413500
* the return and reason code for the CAF open is stored in the 01414400
* fields pointed to by CAFOPEN_RC_PTR and CAFOPEN_RS_PTR.     @L1A 01415300
*                                                                01416200
* If this sample IS compiled with the IADD SYSPARM, then a DB2 01417100
* connection and open thread is assumed and this sample will do 01418000
* an SQL COMMIT and SQL ROLLBACK instead of a CAF CLOSE to perform 01418900
* a DB2 sync or rollback. The CAFOPEN_RC_PTR and CAFOPEN_RS_PTR 01419800
* fields will not be set.                                     @L1A 01420700
*                                                                01421600
* If a DB2 sync or rollback is performed because the SYNC field 01422500
* in the DATAAREA equals "YES", then the return and reason code 01423400
* values of the commit or rollback will be stored in the fields 01424300
* pointed to by CAFCLOSE_RC_PTR and CAFCLOSE_RS_PTR. This sample 01425200
* uses a CAF CLOSE with SYNCH or a CAF CLOSE with ABRT for the 01426100
* MVS batch environment when the SYSPARM IADD is NOT specified 01427000
* and an SQL COMMIT or SQL ROLLBACK when IADD is specified.   @L1A 01427900
*                                                                01428800
*                                                                01435100
* Valid values for FUNCTION_REQUEST:                           @L1A 01436000
* "ACCESS " : OSREQ ACCESS                                     01436900
* "STORE " : OSREQ STORE                                       01437800
* "RETRIEVE" : OSREQ RETRIEVE                                   01438700
* "QUERY " : OSREQ QUERY                                        01439600
* "CHANGE " : OSREQ CHANGE                                       01440500
* "DELETE " : OSREQ DELETE                                       01441400
* "UNACCESS" : OSREQ UNACCESS                                   01442300
*                                                                01443200
*                                                                01444100
*                                                                01445000
* IADDRESS NOTE:                                              01445900
* NOTE: To generate the IADDRESS keyword in the OSREQ ACCESS  01450000
* function specify the SYSPARM value as IADD in the PARM      01500000
* field of the EXEC JCL statement. For example:              01550000
*                                                                01600000

```

```

*      //ASSEMBLE EXEC PGM=ASMA90,PARM='RENT,DECK,SYSPARM(IADD)' @L1C 01650000
*      01700000
* 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.
*      These fields will contain the reason and return
*      codes for DB2 sync & rollback function requests.
*      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.
*      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.
*      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
* $O1=OY29609 320 900219 TPCTGT: Remove unknown macro call
* $P0=KBE0022 331 911216 TUCTNN: IADDRESS support
* $O2=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
* $P2=129204 R23 160808 TUCDEW: Reference CBROSRS3 in comments
* $P3=129177 R23 160811 TUCAED: FMID update to HDZ2230
*****
OSRSAMPL CSECT ,
OSRSAMPL AMODE 31
OSRSAMPL RMODE ANY
      USING *,R15          USING to allow branchto STRTOSREQ
*
      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'HDZ2250 '  APAR LEVEL FOR THIS MODULE
      DROP R15
      SPACE 2
STRTOSREQ DS OH          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.                                04400000
* A pointer with a value of zero is equivalent to an omitted parameter.    04450000
*
L      R0,MESSAGE_AREA_PTR                                                04500000
L      R2,OBJECT_SIZE_PTR                                                 04550000
L      R3,STORAGE_CLASS_PTR                                               04600000
L      R4,MANAGEMENT_CLASS_PTR                                            04650000
L      R6,RETRIEVE_OFFSET_PTR                                             04700000
L      R7,RETRIEVE_LENGTH_PTR                                             04800000
L      R8,RETURN_CODE_PTR                                                 04850000
L      R9,REASON_CODE_PTR                                                 04900000
* Removed RETPD parm from this initial OSREQ invocation                    @L2D 04950000
  OSREQ (STORE),MF=(M,PARM_LIST),                                         X05000000
    MSGAREA=(R0),                DB2 error messages returned here X05050000
    TOKEN=TOKEN_AREA,           Contains logical OAM connection X05100000
    COLLECTN=COLLECTION_NAME,   X05150000
    NAME=OBJECT_NAME,          X05200000
    SIZE=(R2),                 X05250000
    STORCLAS=(R3),             X05300000
    MGMTCLAS=(R4),             X05350000
    OFFSET=(R6),               Starting byte for retrieve X05450000
    LENGTH=(R7),               Length of retrieve X05500000
    RETCODE=(R8),              Register 15 is stored here X05550000
    REACODE=(R9),              Register 0 is stored here 05600000
*
L      R0,TRACKING_TOKEN_PTR                                              @L1A 05603200
L      R2,RETURN_CODE2_PTR                                              @L1A 05604800
L      R3,RECALL_NUM_DAYS_PTR                                           @L1A 05606400
  OSREQ (STORE),MF=(M,PARM_LIST),                                         @L1AX05608000
    VIEW=PRIMARY,              Retrieve Primary Copy @L1AX05609600
    TTOKEN=(R0),               User Tracking Token @L1AX05611200
    RETCODE2=(R2),             Return Code 2 @L1AX05612800
    RECALL=(R3),               Recall Number of Days @L1A 05614400
*
L      R6,R6                    Zero Register @L1A 05616000
* if view=2, then set VIEW=BACKUP @L1A 05617600
  SLR R6,R6                    Load view into R6 @L1A 05619200
  L R6,VIEW                    @P1C 05620800
  LA R10,2                    Load value 2 into R10 @L1A 05622400
  CR R6,R10                   Does view = 2? @L1A 05624000
  BNE TRYVIEW3                No, then see if view = 3 @L1A 05625600
  OSREQ (STORE),MF=(M,PARM_LIST), @L1AX05627200
    VIEW=BACKUP                Retrieve First Backup Copy @L1A 05628800
    B TRYRELBUF                Skip test 'if view=3' @L1A 05630400
*
L      R6,R6                    05632000
* else if view=3, then set VIEW=BACKUP2 @L1A 05633600
  TRYVIEW3 DS 0H @L1A 05635200
  LA R10,3                    Load value 3 into R10 @L1A 05636800
  CR R6,R10                   Does view = 3? @L1A 05638400
  BNE TRYRELBUF                Nope, so leave VIEW=PRIMARY @L1A 05640000
  OSREQ (STORE),MF=(M,PARM_LIST), @L1AX05641600
    VIEW=BACKUP2                Retrieve First Backup Copy @L1A 05643200
*
L      R6,R6                    05644800
  TRYRELBUF DS 0H @L1A 05646400
  CLC RELEASE_BUFFER,=CL3'YES' 05650000
  BNE NORELBUF                05700000
  OSREQ (STORE),MF=(M,PARM_LIST), X05750000
    RELBUF=YES                  Will release pages after STORE 05800000
  NORELBUF DS 0H 05850000
*
L      R6,R6                    @L2A 05851300
* Set RETPD or EVENTEXP or both, based on caller's parm list. @L2A 05852600
*
L      R6,R6                    @L2A 05853900
* Note that a runtime error will occur if non-zero pointers are @L2A 05855200
* present for both RETPD and EVENTEXP. Supplying both RETPD and @L2A 05856500
* EVENTEXP is generally only useful for testing the error checking @L2A 05857800
* features of the OSREQ processing code. @L2A 05859100
*
L      R5,RETENTION_PERIOD_PTR @L2A 05860400
  OSREQ (STORE),MF=(M,PARM_LIST), @L2A 05861700
    RETPD=(R5) @L2AX05863000
*
L      R5,EVENTEXP_PTR @L2A 05864300
  OSREQ (CHANGE),MF=(M,PARM_LIST), @L2A 05865600
    EVENTEXP=(R5) @L2A 05866900
*
L      R5,EVENTEXP_PTR @L2AX05868200
  OSREQ (CHANGE),MF=(M,PARM_LIST), @L2A 05869500
    EVENTEXP=(R5) @L2A 05870800
*
L      R5,EVENTEXP_PTR @L2A 05872100
  OSREQ (CHANGE),MF=(M,PARM_LIST), @L2A 05873400
    EVENTEXP=(R5) @L2A 05874700
*
L      R5,EVENTEXP_PTR @L2A 05876000
  OSREQ (CHANGE),MF=(M,PARM_LIST), @L2A 05877300
    EVENTEXP=(R5) @L2A 05878600
*
L      R5,EVENTEXP_PTR @L2A 05879900
  OSREQ (CHANGE),MF=(M,PARM_LIST), @L2A 05881200
    EVENTEXP=(R5) @L2A 05882500

```

```

      B      DELHDONE                                @L2A 05882500
*
* DELHNO      DS      0H                                @L2A 05883800
      OSREQ  (STORE),MF=(M,PARM_LIST),                @L2A 05885100
      B      DELHDONE                                @L2AX05886400
      B      DELHDONE                                @L2A 05887700
*                                                    @L2A 05889000
*                                                    @L2A 05890300
* DELHYES     DS      0H                                @L2A 05891600
      OSREQ  (STORE),MF=(M,PARM_LIST),                @L2AX05892900
      B      DELHDONE                                @L2A 05894200
*                                                    @L2A 05895500
* DELHDONE    DS      0H                                05900000
      CLC FUNCTION_REQUEST,=CL8'ACCESS'              05950000
      BNE TRY_STORE                                06000000
*
* The logical connection to OAM is made here.        06050000
* If this is MVS batch, the Call Attach Facility will be used 06100000
* to connect to DB2, and a thread will be OPENed to Plan(CBRIDBS) 06150000
* otherwise, the connection is done by the environment in which 06200000
* this program is executing.                          06250000
* In all cases system control blocks will be created and/or modified 06300000
* to provide this access to OAM.                      06350000
*                                                    06400000
* To generate the IADDRESS keyword in the OSREQ ACCESS 06450000
* function, specify the SYSPARM value as IADD in the PARM 06500000
* field of the EXEC JCL statement. See NOTE in prolog. 06550000
*                                                    06600000
*      AIF ('&SYSPARM' EQ 'IADD').IA2                06650000
*      OSREQ ACCESS,MF=(E,PARM_LIST)                06700000
*      AGO    .SKIP1                                06750000
* .IA2        ANOP                                  06800000
* In this sample we use DSNHLI for SQL interface module to DB2 06850000
*      L      R2,=V(DSNHLI)                        06900000
*      OSREQ ACCESS,MF=(E,PARM_LIST),                X06950000
*      IADDRESS=(R2)                                07000000
* .SKIP1      ANOP                                  07050000
*                                                    07100000
* In the MVS batch environment, syncpoint processing may be desirable 07150000
* after ACCESS because the DB2 plan name can be changed at this time. 07200000
*                                                    07250000
* TRY_STORE    B      TRY_SYNC_POINT                07300000
*      DS      0H                                07350000
*      CLC FUNCTION_REQUEST,=CL8'STORE'              07400000
*      BNE TRY_CHANGE                              07450000
*                                                    07500000
* This will store an object in the DB2 object tables or on 07550000
* an optical disk, depending on the storage class specified. 07600000
*                                                    07650000
*      L      R10,STORE_BUFFER_PTR                  07700000
*      OSREQ STORE,MF=(E,PARM_LIST),                X07750000
*      BUFLIST=(R10)                                07800000
*      B      TRY_SYNC_POINT                        07850000
* TRY_CHANGE   DS      0H                                07900000
*      CLC FUNCTION_REQUEST,=CL8'CHANGE'            07950000
*      BNE TRY_QUERY                                08000000
*                                                    08050000
* This invocation of the OSREQ macro will change information in the 08100000
* directory that has been specified. A zero pointer in DATAAREA 08150000
* will result in no change for the respective information. All 08200000
* pointers zero result in no change.                 08250000
*                                                    08300000
*      OSREQ CHANGE,MF=(E,PARM_LIST)                08350000
*      B      TRY_SYNC_POINT                        08400000
* TRY_QUERY    DS      0H                                08450000
*      CLC FUNCTION_REQUEST,=CL8'QUERY'            08500000
*      BNE TRY_RETRIEVE                            08550000
*                                                    08600000
* Query the data base for the directory information that was stored. 08650000
* The size of the object can be extracted from this information so 08700000
* that a GETMAIN can be done for the area necessary for the 08750000
* retrieve operation.                                08800000
*                                                    08850000
*      L      R10,QUERY_BUFFER_PTR                  08900000
*      OSREQ QUERY,MF=(E,PARM_LIST),                X08950000
*      QEL=(R10)                                    09000000
*      B      TRY_SYNC_POINT                        09050000
* TRY_RETRIEVE DS      0H                                09100000
*      CLC FUNCTION_REQUEST,=CL8'RETRIEVE'          09150000
*      BNE TRY_DELETE                              09200000
*                                                    09250000
* A partial retrieve can be done to obtain the first xxx bytes of 09300000
* the object. In some cases the application may have some control 09350000
* information in this area to allow retrieval of still another part 09400000

```

```

* of the object, (which could be an image).                                09450000
*                                                                           09500000
                                                                           09525000
                                                                           09550000
        L      R10,RETRIEVE_BUFFER_PTR                                X09600000
    OSREQ RETRIEVE,MF=(E,PARM_LIST),                                09650000
        BUFLIST=(R10)                                09700000
        B      TRY_SYNC_POINT                                09750000
TRY_DELETE DS      0H                                09800000
        CLC FUNCTION_REQUEST,=CL8'DELETE'                    09850000
        BNE TRY_UNACCESS                                09900000
*                                                                           09950000
* This invocation will delete the object named from the object table    10000000
* and the directory.                                                    10050000
*                                                                           10100000
        OSREQ DELETE,MF=(E,PARM_LIST)                            10150000
        B      TRY_SYNC_POINT                                10200000
TRY_UNACCESS DS      0H                                10250000
        CLC FUNCTION_REQUEST,=CL8'UNACCESS'                    10300000
        BNE TRY_SYNC_POINT                                @L1C 10350000
*                                                                           10400000
* The logical connection to OAM should be broken before the TASK        10450000
* terminates so that OAM can remove any system control blocks          10500000
* that it built during ACCESS                                            10550000
*                                                                           10600000
        OSREQ UNACCESS,MF=(E,PARM_LIST)                          10650000
*                                                                           10700000
TRY_SYNC_POINT DS      0H                                10750000
*                                                                           10800000
* Save register 15 in the return code area and register 0 in the        10850000
* reason code area after return from OSREQ. This is recommended        10900000
* because, under certain error conditions, the return code and        10950000
* reason code areas may not be set by OSREQ.                            11000000
*                                                                           11050000
        ST      R15,0(,R8)      Save Return Code                11100000
        ST      R0,0(,R9)      Save Reason Code                  11150000
*                                                                           11200000
* Each function should be "committed" or "rolled back" depending      11250000
* on the return and reason codes from OAM.                              11300000
* This routine should issue:                                            11350000
*   SYNCPOINT with optional ROLLBACK in the CICS environment            11400000
* or SYNC or ROLL,ROLLB in the IMS environment                          11450000
* or COMMIT or ROLLBACK in the TSO environment                          11500000
* or CALL DSNALI to CLOSE and OPEN the thread to DB2 in the            11550000
*   MVS batch environment (which is shown here).                        11600000
*                                                                           11650000
        SR      R15,R15      Ensure return code 0 if             11700000
*                                                                           no syncpoint processing.
        CLC SYNC_POINT,=CL3'YES'                                    11750000
        BNE EXIT                                                    11800000
*                                                                           11850000
* A parameter list is constructed for the call to DSNALI                11900000
* to close the thread to commit or rollback changes.                    11950000
*                                                                           12000000
        LA      R10,=CL12'CLOSE'                                    12050000
        ST      R10,WORK_AREA1      Set function to close.        12100000
        LA      R10,=CL8'SYNC'      Prime for sync.                12150000
AIF ('&SYSPARM' EQ 'IADD').IA1                                          12200000
        L      R15,RETURN_CODE_PTR      Check OAM return code      12250000
        LA      R9,4                  to see if rollback should    12300000
        C      R9,0(R15)              be issued instead of sync.    12350000
        BNL SET_SYNC                  12400000
        LA      R10,=CL4'ABRT'        12450000
SET_SYNC ST      R10,WORK_AREA2      Set the action parameter.      12500000
        OI      WORK_AREA2,X'80'      Set end of parameter list    12550000
        BAL     R10,LOAD_DSNALI        This points R15 to DSNALI.    12600000
        LA      R1,WORK_AREA1          Point to parameter list.      12650000
        CALL    (15)                  Call DSNALI                    12700000
* Save CAF return code                                                  12705500
* Note: We already saved the rc for other functions (access,           12711000
* store, etc), so don't want to overwrite that rc w/ the               12716500
* commit/rollback rc                                                  @L1A 12722000
SAVE_CAFRC L      R8,CAF_CLOSE_RC_PTR                                @L1A 12727500
        L      R9,CAF_CLOSE_RS_PTR                                @L1A 12733000
        ST     R15,0(,R8)      Save CAF_CLOSE RETCODE              @L1A 12738500
        ST     R0,0(,R9)      Save CAF_CLOSE REASCODE              @L1A 12744000
        LTR    R15,R15      Check for good return                  12750000
        BNZ    EXIT          This routine has no                    12800000
*                                                                           recovery for bad returns
*                                                                           from CLOSE. The caller
*                                                                           should UNACCESS then ACCESS.
*                                                                           12900000
*                                                                           12950000
*                                                                           13000000
*                                                                           13050000
        AGO     .SKIP

```



```

.IA1 ANOP                                13100000
        LA      R8,SQLSTUFF                13150000
        USING   SQLDSECT,R8                13200000
        L       R15,RETURN_CODE_PTR        13250000
        LA      R9,4                        13300000
        C       R9,0(R15)                   13350000
        BNL     SET_SYNC                    13400000
        EXEC    SQL ROLLBACK                13450000
        B       SAVE_SQLCODES               @L1C 13500000
SET_SYNC EXEC    SQL COMMIT                 13550000
* Save SQL return codes                    13556600
* 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 13576400
SAVE_SQLCODES L      R8,CAFCLOSE_RC_PTR      @L1A 13583000
               L      R9,CAFCLOSE_RS_PTR      @L1A 13589600
               ST     R15,0(,R8)              @L1A 13596200
               ST     R0,0(,R9)              @L1A 13602800
               AGO     .SKIP2                 @L1C 13635800
                                           13642400
.SKIP ANOP                                13650000
*                                           13700000
* A parameter list is constructed for the call to DSNALI 13750000
* to open the thread to DB2. A new plan name could be specified
* or the same name (CBRIDBS) could be specified. 13800000
*                                           13850000
*                                           13900000
               LA      R10,=CL12'OPEN'        13950000
               ST     R10,WORK_AREA1          Set function to open. 14000000
               LA      R10,DB2_SUBSYS_ID       14050000
               ST     R10,WORK_AREA2          Set the ssid parameter. 14100000
               LA      R10,PLAN_NAME           14150000
               ST     R10,WORK_AREA3          Set the thread parameter. 14200000
               OI     WORK_AREA3,X'80'        Set end of parameter list 14250000
               BAL    R10,LOAD_DSNALI         This points R15 to DSNALI. 14300000
               LA      R1,WORK_AREA1          Point to parameter list. 14350000
               CALL   (15)                   Call DSNALI 14400000
               L       R8,CAFOPEN_RC_PTR       @L1A 14410000
               L       R9,CAFOPEN_RS_PTR       @L1A 14420000
               ST     R15,0(,R8)              @L1A 14430000
               ST     R0,0(,R9)              @L1A 14440000
               Save Return Code
               Save Reason Code
.SKIP2 ANOP                                14450000
EXIT DS      0H                            14500000
*                                           14550000
* Restore all registers except regs 15 and 0, then return to caller
*                                           14600000
*                                           14650000
               L       R13,SAVE_AREA+4        14700000
               L       R14,12(R13)            14750000
               LM      R1,R12,24(R13)         14800000
               BR      R14                    14850000
*                                           14900000
* This subroutine will determine if DSNALI is loaded. 14950000
* If it is, register 15 will be return with the address of DSNALI. 15000000
* If it is not, the module will be loaded and the address returned
* in register 15. 15100000
* 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. 15150000
*                                           15200000
*                                           15250000
*                                           15300000
LOAD_DSNALI DS      0H                    15350000
               L       R15,WORK_AREA4         DSNALI address is saved here. 15400000
               LTR     R15,R15                15450000
               BNZR    R10                    Return with address of DSNALI 15500000
               LOAD    EP=DSNALI              DB2 CAF MVS batch LI services 15533300
               ST     R0,WORK_AREA4           Save for future calls. 15600000
               LR      R15,R0                  Return address of DSNALI 15650000
               BR      R10                     to caller 15700000
*                                           15750000
* Register definitions 15800000
*                                           15850000
R0 EQU      0                                15900000
R1 EQU      1                                15950000
R2 EQU      2                                16000000
R3 EQU      3                                16050000
R4 EQU      4                                16100000
R5 EQU      5                                16150000
R6 EQU      6                                16200000
R7 EQU      7                                16250000
R8 EQU      8                                16300000
R9 EQU      9                                16350000
R10 EQU     10                               16400000
R11 EQU     11                               16450000
R12 EQU     12                               16500000

```

R13	EQU	13	16550000
R14	EQU	14	16600000
R15	EQU	15	16650000
*			16700000
* All literals will be included at this point.			16750000
*			16800000
LTORG			16850000
*			16900000
* This static parameter list will be used as a template for			16950000
* OSREQ invocations in the executable code.			17000000
*			17050000
STATIC_PARM_LIST OSREQ (STORE),MF=(L)			17100000
STATIC_LIST_END EQU *			17150000
*			17200000
* This area is provided by the caller of this routine			17250000
*			17300000
DATAAREA DSECT			17350000
*****			17400000
*			17450000
* This area must be obtained by the caller of OSRSAMPL and presented			17500000
* as a parameter to OSRSAMPL. It is expected that all subsequent calls			17550000
* will point to this same area. There is information in the area			17600000
* that will be used across calls.			17650000
*			17700000
*****			17750000
SAVE_AREA DS 18F Savearea for this module.			17800000
*****			17850000
* The following two named fields are set by the caller of OSRSAMPL.			17900000
* If the value in the field is not a valid value, the respective			17950000
* activity not be executed.			18000000
*****			18050000
FUNCTION_REQUEST	DS CL8	OSREQ function request value	18100000
*		ACCESS, STORE, etc. or other	18150000
SYNC_POINT	DS CL3	Syncpoint request, YES or other	18200000
	DS CL1	Reserved	18250000
*****			18300000
* The following five fields are set by OSRSAMPL and should not be			18350000
* altered by the caller. Subsequent calls to OSRSAMPL will rely			18400000
* on the information stored here.			18450000
*****			18500000
WORK_AREA1	DS A	Used	18550000
WORK_AREA2	DS A	for	18600000
WORK_AREA3	DS A	parameters.	18650000
WORK_AREA4	DS A	Holds address of DSNALI	18700000
TOKEN_AREA	DS 2F	OSREQ token, do not change it.	18750000
*****			18800000
* The following fields are set by the caller of OSRSAMPL			18850000
* The pointers are not altered by OSRSAMPL but the data that			18900000
* the pointers reference may be.			18950000
*****			19000000
RETURN_CODE_PTR	DS A	Pointer to OSREQ return code	19050000
*		The return code is referenced by	19100000
*		the syncpoint processing.	19150000
REASON_CODE_PTR	DS A	Pointer to OSREQ reason code	19200000
MESSAGE_AREA_PTR	DS A	Pointer to message area	19250000
RETENTION_PERIOD_PTR	DS A	Pointer to retention period	19300000
OBJECT_SIZE_PTR	DS A	Pointer to object size value	19350000
MANAGEMENT_CLASS_PTR	DS A	Pointer to management class parameter	19400000
STORAGE_CLASS_PTR	DS A	Pointer to storage class parameter	19450000
RETRIEVE_OFFSET_PTR	DS A	Pointer to offset value	19500000
RETRIEVE_LENGTH_PTR	DS A	Pointer to retrieve length value	19550000
RETRIEVE_BUFFER_PTR	DS A	Pointer to retrieve buffer list	19600000
STORE_BUFFER_PTR	DS A	Pointer to store buffer list	19650000
QUERY_BUFFER_PTR	DS A	Pointer to query buffer list	19700000
RELEASE_BUFFER	DS CL3	RELBUF value, YES or other	19750000
	DS CL1	Reserved	19800000
VIEW	DS F	Retrieve Object Copy @L1A	19804100
*		1 = PRIMARY @L1A	19808200
*		2 = First BACKUP Copy @L1A	19812300
*		3 = Second BACKUP Copy @L1A	19816400
TRACKING_TOKEN_PTR	DS A	User Tracking Token Pointer @L1A	19820500
RECALL_NUM_DAYS_PTR	DS A	Recall Number of Days Pointer @L1A	19824600
RETURN_CODE2_PTR	DS A	Return Code 2 Pointer @L1A	19828700
CAFOPEN_RC_PTR	DS A	Pointer to the OPEN CAF return code@L1A	19832800
CAFOPEN_RS_PTR	DS A	Pointer to the OPEN CAF reason code@L1A	19836900
CAFCLOSE_RC_PTR	DS A	Pointer to the CLOS CAF return code@L1A	19841000
CAFCLOSE_RS_PTR	DS A	Pointer to the CLOS CAF reason code@L1A	19845100
*			19850000
* Plan name and DB2 subsystem identification MUST be provided			19900000
* for MVS batch sync point processing.			19950000
*			20000000
PLAN_NAME	DS CL8	DB2 plan name for OPEN thread	20050000

```

DB2_SUBSYS_ID      DS CL4      Installation subsystem name for DB2.      20100000
*                                                           20150000
* Collection name and object name MUST be provided with every  20200000
* request for STORE, RETRIEVE, QUERY, CHANGE, and DELETE.      20250000
*                                                           20300000
COLLECTION_NAME    DS H        Length of collection name          20350000
                   DS CL44     Collection name character string    20400000
OBJECT_NAME        DS H        Length of object name             20450000
                   DS CL44     Object name character string        20500000
DELHOLD            DS CL8      DELHOLD= HOLD | NOHOLD | blank      @L2A 20516600
EVENTEXP_PTR       DS A        Pointer to EVENTEXP value          @L2A 20533200
*****                                                    20550000
* The following area is completely overlaid each time OSRSAMPL  20600000
* is called                                                    20650000
*****                                                    20700000
PARM_LIST DS CL(STATIC_LIST_END-STATIC_PARM_LIST) Dynamic parm list 20750000
          DS CL2528 DO NOT USE THIS AREA, BELONG TO CALLER          20800000
          EXEC SQL INCLUDE SQLCA                                     20850000
SQLSTUFF DS CL(SQLDLEN)                                           20900000
DATA_AREA_END EQU *                                               20950000
OSRSAMPL CSECT                                                    21000000
*                                                           @O1D 21050000
          END OSRSAMPL                                           21100000

```

CBROS2

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

```

***** 00050000
* 00100000
* DESCRIPTIVE NAME: Object Storage Request Sample interface #2 00150000
* 00200000
* Licensed Materials - Property of IBM 00210000
* 5650-ZOS 00220000
* COPYRIGHT IBM CORP. 2008, 2017 00230000
* 00240000
* FUNCTION: Provides a generalized interface for the Object Storage 00250000
* Request (OSREQ) macro. 00300000
* 00310000
* This interface does not support specifying DB2ID on an @P3A 00311000
* ACCESS request (needed in many cases for a multiple @P3A 00312000
* OAM configuration). @P3A 00313000
* Please see sample job CBROS3 for support of the DB2ID @P3A 00314000
* keyword on an ACCESS request. @P3A 00315000
* 00350000
* OPERATION: This routine is called with a parameter area that 00400000
* defines the function and pointers necessary to invoke 00450000
* the OSREQ macro. 00500000
* 00550000
* If it is determined that an OSREQ function is requested, 00600000
* then the OSREQ parameter list is filled in with an 00650000
* MF=M form of the macro. The function is executed via an 00700000
* MF=E form. 00750000
* 00800000
* 1. Validity check the DATAAREA Header. Exit if error. 00850000
* 2. Fill in the OSREQ PARM_LIST with all of the optional 00900000
* keywords using MF=M form of the macro. 00950000
* 3. If FUNCTION_REQUEST = "ACCESS " @P1C 01000000
* a. IF CBROS2 was compiled with IADD option, then 01050000
* set IADDRESS OSREQ macro keyword to the address of 01100000
* the DB2 library entry point DSNHLI using the MF=M 01150000
* form of the macro. @P1C 01175000
* b. ELSE set IADDRESS OSREQ macro keyword to 01200000
* IADDRESS_PTR using the MF=M form of the macro. @P1A 01225000
* 4. SELECT FUNCTION_REQUEST 01250000
* WHEN(ACCESS, STORE, RETRIEVE, QUERY, CHANGE, DELETE, 01300000
* UNACCESS, STOREBEG, STOREPRT, STOREEND) 01350000
* a. Set any function specific keywords 01400000
* b. Execute specified function using the MF=E form 01450000
* of the macro. 01500000
* c. Set R15 to 0, to indicate successful OSREQ 01550000
* macro invocation 01600000
* OTHERWISE: 01650000
* a. Set R15 to Invalid Function Request 01700000
* 5. Return to caller 01750000
* 01800000
* Valid values for FUNCTION_REQUEST: 01850000
* "ACCESS " : OSREQ ACCESS 01900000
* "STORE " : OSREQ STORE 01950000

```

*	"RETRIEVE" : OSREQ RETRIEVE	02000000
*	"QUERY " : OSREQ QUERY	02050000
*	"CHANGE " : OSREQ CHANGE	02100000
*	"DELETE " : OSREQ DELETE	02150000
*	"UNACCESS" : OSREQ UNACCESS	02200000
*	"STOREBEG" : OSREQ STOREBEG	02250000
*	"STOREPRT" : OSREQ STOREPRT	02300000
*	"STOREEND" : OSREQ STOREEND	02350000
*		02400000
*		02450000
*	IADDRESS NOTE:	02500000
*	To specify the default DSNHLI entry point for the	02540000
*	IADDRESS keyword in the OSREQ function, specify	02580000
*	the SYSPARM value as IADD in the PARM field of	02620000
*	the EXEC JCL statement. For example:	@P1C 02660000
*		02700000
*	//ASSEMBLE EXEC PGM=ASMA90,PARM='RENT,DECK,SYSPARM(IADD)'	02750000
*		02800000
*	REGISTER CONVENTIONS:	02850000
*	R0 - WORK REGISTER	02900000
*	R1 - STANDARD LINKAGE REGISTER	02950000
*	- PARAMETER LIST ADDRESS	03000000
*	R2 - WORK REGISTER	03050000
*	R3 - WORK REGISTER	03100000
*	R4 - WORK REGISTER	03150000
*	R5 - WORK REGISTER	03200000
*	R6 - WORK REGISTER	03250000
*	R7 - WORK REGISTER	03300000
*	R8 - WORK REGISTER	03350000
*	R9 - WORK REGISTER	03400000
*	R10 - WORK REGISTER	03450000
*	R11 - DATAAREA BASE REGISTER	03500000
*	R12 - OSR2SAMP BASE REGISTER	03550000
*	R13 - STANDARD LINKAGE REGISTER	03600000
*	- SAVE AREA ADDRESS	03650000
*	R14 - STANDARD LINKAGE REGISTER	03700000
*	- RETURN POINT ADDRESS	03750000
*	R15 - STANDARD LINKAGE REGISTER	03800000
*	- ENTRY POINT ADDRESS	03850000
*	- RETURN CODE	03900000
*		03950000
*	INPUT: Register 1 must point to a 4 byte field that contains	04000000
*	an address of an area that is described by	04050000
*	the dsect named DATAAREA in this program.	04100000
*	The DATAAREA must be filled in to indicate	04150000
*	the function requested and provide the proper	04200000
*	data for execution of the OSREQ macro.	04250000
*	Register 13 must point to a 72 byte area into which this	04300000
*	routine will save the registers at entry and	04350000
*	from which registers will be restore at exit.	04400000
*	Register 14 must point to the instruction address to which	04450000
*	this routine will return.	04500000
*	Register 15 must point to the entry point address of this	04550000
*	routine.	04600000
*	OUTPUT: Register 15 will contain the return code from DATAAREA	04650000
*	validity checking.	04700000
*	CODE MEANING	04750000
*	0 SUCCESS--OSREQ Function invoked	04800000
*	6 Invalid DATAAREA FUNCTION_REQUEST	04850000
*	8 Invalid DATAAREA hdr ID	04900000
*	10 Invalid DATAAREA hdr length	04950000
*	12 Invalid DATAAREA hdr version	05000000
*	14 Invalid DATAAREA hdr release	05050000
*		05100000
*	Fields pointed to by REASON_CODE_PTR and RETURN_CODE_PTR	05150000
*	will contain the reason and return codes returned	05200000
*	from OAM for OSREQ function requests.	05250000
*	Areas defined by the CBRIBUFL (for retrieve) and CBRIQEL	05300000
*	(for query) will be filled in when the respective	05350000
*	function is requested.	05400000
*		05450000
*	CHANGE-ACTIVITY:	05500000
*	\$L0=OAM2GB R1A 070316 TUCGPW: OAM2GB Phase 1	05550000
*	\$P0=K1A2012 R1A 080109 TUCGPW: Fixed loading VIEW into register	05575000
*	\$P1=K1A2309 R1A 080228 TUCGPW: Clarify how and when we set	05581200
*	the IADDRESS OSREQ function	05587400
*	keyword.	05593600
*	\$01=0A25764 R1A 080624 TUCGPW: Add backward compatibility	05595200
*	\$L1=OAMR1B R1B 080716 TUCDVH: OAMARE Archive retention	@L1A 05596800
*	\$P2=K1B0132 R1B 080721 TUCDVH: STIMEOUT support	@P2A 05598400
*	\$L2=OAMR22B R22 120828 TUCRGV: OAMR22 64BitBuffers	@L2A 05599000
*	\$L3=OAMR23M R23 160405 TUCAED: Multi OAM AS Stage 1	@L3A 05599100

```

*      $P3=129204 R23 160808 TUCDEW: Reference CBR0SR3 in comments @P3A 05599200
*
*****
OSR2SAMP CSECT , 05600000
OSR2SAMP AMODE 31 05650000
OSR2SAMP RMODE ANY 05700000
          USING *,R15          USING to allow branch to STRT0SR2 05750000
*
          SPACE 2 05800000
          B STRT0SR2          BRANCH TO ACTIVE PART OF MODULE 05900000
LENG0SR2 DC X'18'          LENGTH OF HEADER INFORMATION 05950000
NAME0SR2 DC CL8'CBR0SR2 ' 06000000
DATE0SR2 DC CL8'&SYSDATE' 06050000
APAR0SR2 DC CL8'HDZ2250 ' 06100000
          DROP R15          MODULE ASSEMBLY DATE 06150000
          SPACE 2          APAR LEVEL FOR THIS MODULE @L3C 06200000
STRT0SR2 DS 0H          START THE ACTIVE PART OF MODULE 06250000
*
          STM R14,R12,12(R13) 06300000
*
* Register 12 is the base for the code 06350000
*
          LR R12,R15 06400000
          USING OSR2SAMP,R12 06450000
*
* Register 11 is the base for the data area which is passed to this 06500000
* routine as a parameter. 06550000
*
          L R11,0(R1) 06600000
          USING DATAAREA,R11 06650000
          LA R15,SAVE_AREA 06700000
          ST R15,8(R13) 06750000
          ST R13,SAVE_AREA+4 06800000
          LR R13,R15 06850000
*
* The static OSREQ parameter list is copied into the work area 06900000
*
          MVC PARM_LIST,STATIC_PARM_LIST 06950000
*
* Do some DATAAREA Header Validity Checking 07000000
*
* Make sure the ID of the user's dataarea = current OSR2 ID 07050000
          LA R15,ERR_ID          Load ERR_ID into R15 07100000
          CLC DA_ID,=CL4'OSR2'  Does DA_ID == ID 07150000
          BNE EXIT              Exit if not equal 07200000
*
* Make sure the length of the user's dataarea = current OSR2 length 07250000
          LA R15,ERR_LEN          Load ERR_LEN into R15 07300000
          L R0,DA_LEN 07350000
          CFI R0,DATAAREA_LEN  Does DA_LEN = LENGTH 07400000
          BNE EXIT              Exit if not equal 07450000
*
* Make sure user's dataArea version is <= current OSR2 version @01C 07500000
          LA R15,ERR_VER          Load ERR_VER into R15 07550000
          SLR R2,R2              Zero Register 07600000
          IC R2,DA_VER          Load DA_VER into R2 07650000
          LA R3,OSR2_VER          Load VERSION into R3 07700000
          CR R3,R2              Does DA_VER = VERSION? @P0C 07750000
          BL EXIT              Exit w/ err if VERSION 07800000
*
* Make sure user's dataArea release is <= current OSR2 release @01C 07850000
          LA R15,ERR_REL          Load ERR_REL into R15 07900000
          SLR R2,R2              Zero Register 07950000
          IC R2,DA_REL          Load DA_REL into R2 08000000
          LA R3,OSR2_REL          Load RELEASE into R3 08050000
          CR R3,R2              Does DA_REL = RELEASE? @P0C 08100000
          BL EXIT              Exit w/ err if RELEASE 08150000
*
* Modify the parameter list to establish all the basic OSREQ function 08200000
* parameters. 08250000
*
* Note: A pointer with a value of zero is equivalent to an omitted parm 08300000
*
OSR_FUNC DS 0H 08350000
          L R0,COLLECTION_NAME_PTR 08400000
          L R2,MANAGEMENT_CLASS_PTR 08450000
          L R3,MESSAGE_AREA_PTR 08500000
          L R4,OBJECT_NAME_PTR 08550000

```

```

L      R5,OBJECT_SIZE_PTR                                09550000
L      R6,OFFSET_PTR                                    09600000
L      R7,REASON_CODE_PTR                               09650000
L      R8,RECALL_NUM_DAYS_PTR                           09700000
L      R10,RETRIEVE_LENGTH_PTR                          09850000
*
* Removed RETPD parm from this initial OSREQ invocation    @L1D 09883200
*
OSREQ (STORE),MF=(M,PARM_LIST),                          X09950000
    TOKEN=TOKEN_AREA,      Contains logical OAM connection X100000000
    COLLECTN=(R0),          X10050000
    MGMTCLAS=(R2),          X10100000
    MSGAREA=(R3),          DB2 error messages returned here X10150000
    NAME=(R4),              X10200000
    SIZE=(R5),              X10250000
    OFFSET=(R6),            Starting byte for retrieve       X10300000
    REACODE=(R7),           Register 0 is stored here        X10350000
    RECALL=(R8),            Recall Number of Days            X10400000
    LENGTH=(R10),           Length of retrieve                10500000
* Ran out of registers above -- add remaining PTRs        10550000
L      R0,RETURN_CODE_PTR                                10600000
L      R2,RETURN_CODE2_PTR                               10650000
L      R3,STORAGE_CLASS_PTR                              10700000
L      R4,TRACKING_TOKEN_PTR                             10750000
L      R5,OBJECT_SIZE64_PTR                              @L2A 10760000
L      R6,OFFSET64_PTR                                   @L2A 10770000
L      R7,RETRIEVE_LENGTH64_PTR                          @L2A 10780000
*
OSREQ (STORE),MF=(M,PARM_LIST),                          X10800000
    RETCODE=(R0),          Register 15 is stored here        X10850000
    RETCODE2=(R2),         Return Code 2                     X10900000
    STORCLAS=(R3),         X10950000
    TTOKEN=(R4),           User Tracking Token               @L2CX11050000
    SIZE64=(R5),           @L2AX11060000
    OFFSET64=(R6),         @L2AX11070000
    LENGTH64=(R7),         @L2A 11080000
*
* Set RELBUF=YES if DATAAREA RELEASE_BUFFER == "YES"    11100000
TRYRELBUF DS 0H                                           11150000
          CLC RELEASE_BUFFER,=CL8'YES'                    11200000
          BNE BUFDONE RELBUF=NO is default                 @L1C 11250000
OSREQ (STORE),MF=(M,PARM_LIST),                          X11350000
    RELBUF=YES            Will release pages after STORE    11400000
BUFDONE DS 0H                                           @L1A 11402700
*
* Set RETPD or EVENTEXP or both, based on caller's parm list. @L1A 11405400
*
* Note that a runtime error will occur if non-zero pointers are @L1A 11408100
* present for both RETPD and EVENTEXP. Supplying both RETPD and @L1A 11410800
* EVENTEXP is generally only useful for testing the error checking @L1A 11413500
* features of the OSREQ processing code.                  @L1A 11416200
*
L      R9,RETENTION_PERIOD_PTR                           @L1A 11418900
OSREQ (STORE),MF=(M,PARM_LIST),                          @L1A 11421600
    RETPD=(R9)                                                @L1A 11424300
*
L      R9,EVENTEXP_PTR                                   @L1A 11427000
OSREQ (CHANGE),MF=(M,PARM_LIST), EVENTEXP only on CHANGE @L1AX11429700
    EVENTEXP=(R9)                                             @L1A 11432400
*
* Set the DELHOLD parm or leave it off.                   @L1A 11435100
*
DELHCHK DS 0H                                           @L1A 11437800
          CLC DELHOLD,=CL8'HOLD'                            @L1AX11440500
          BE DELHYES                                         @L1A 11443200
*
          CLC DELHOLD,=CL8'NOHOLD'                          @L1A 11445900
          BE DELHNO                                           @L1A 11448600
          B DELHDONE                                         @L1A 11451300
*
          CLC DELHOLD,=CL8'NOHOLD'                          @L1A 11454000
          BE DELHNO                                           @L1A 11456700
          B DELHDONE                                         @L1A 11459400
*
          CLC DELHOLD,=CL8'NOHOLD'                          @L1A 11462100
          BE DELHNO                                           @L1A 11464800
          B DELHDONE                                         @L1A 11467500
*
          CLC DELHOLD,=CL8'NOHOLD'                          @L1A 11470200
          BE DELHNO                                           @L1A 11472900
          B DELHDONE                                         @L1A 11475600
*
DELHNO DS 0H                                           @L1AX11478300
OSREQ (STORE),MF=(M,PARM_LIST),                          @L1A 11481000
    DELHOLD=NOHOLD                                           @L1A 11483700
    B DELHDONE                                               @L1A 11486400
*
DELHYES DS 0H                                           @L1A 11489100
OSREQ (STORE),MF=(M,PARM_LIST),                          @L1AX11491800
    DELHOLD=HOLD                                             @L1A 11494500
DELHDONE DS 0H                                           @L1A 11497200
*
* Keep testing FUNCTION_REQUEST until an OSREQ FUNCTION match is found 11500000
* or no more functions are found                            11550000
*                                                           11600000

```

```

* If a match is found, then go ahead and execute that function      11650000
*                                                                    11700000
                                                                    11750000
* Execute ACCESS if FUNCTION_REQUEST == "ACCESS"                  11800000
TRY_ACCESS      DS      0H                                         11850000
                CLC FUNCTION_REQUEST,=CL8'ACCESS'                 11900000
                BNE TRY_STORE                                     11950000
*                                                                    12000000
* The logical connection to OAM is made here.                     12050000
* If this is MVS batch, the Call Attach Facility will be used     12100000
* to connect to DB2, and a thread will be OPENed to Plan(CBRIDBS) 12150000
* otherwise, the connection is done by the environment in which   12200000
* this program is executing.                                       12250000
* In all cases system control blocks will be created and/or modified 12300000
* to provide this access to OAM.                                    12350000
*                                                                    12400000
* To specify the default DSNHLI entry point for the               12440000
* IADDRESS keyword in the OSREQ function, specify                 12480000
* the SYSPARM value as IADD in the PARM field of                  12520000
* the EXEC JCL statement. See NOTE in prolog.                     12560000
*                                                                    12600000
                AIF ('&SYSPARM' EQ 'IADD').IA2                     12650000
*                                                                    12660000
* IADD not specified, so set IADDRESS OSREQ macro keyword to     12670000
* IADDRESS_PTR using the MF=M form of the macro.                  @P1A 12680000
*                                                                    12690000
                L      R2,IADDRESS_PTR      Load IADR from parmList 12700000
                OSREQ ACCESS,MF=(E,PARM_LIST),                     X12750000
                IADDRESS=(R2)                                       12800000
                AGO     .SKIP1                                     12850000
.IA2      ANOP                                                    12900000
* IADD was specified so set default entry point.                  @P1A 12925000
* In this sample we use DSNHLI for SQL interface module to DB2    12950000
*                                                                    12975000
                L      R2,=V(DSNHLI)                               13000000
                OSREQ ACCESS,MF=(E,PARM_LIST),                     X13050000
                IADDRESS=(R2)      GET THE ADDRESS OF THE INTERFACE 13100000
.SKIP1     ANOP                                                    13150000
                B      SAVE_RC                                     13200000
                                                                    13250000
* Execute STORE if FUNCTION_REQUEST == "STORE"                   13300000
TRY_STORE      DS      0H                                         13350000
                CLC FUNCTION_REQUEST,=CL8'STORE'                 13400000
                BNE TRY_RETRIEVE                                  13450000
*                                                                    13500000
* This will store an object in the DB2 object tables or on       13550000
* an optical disk, depending on the storage class specified.     13600000
*                                                                    13650000
                L      R9,BUFFER64_PTR_PTR                        @L2A 13660000
                L      R10,STORE_BUFFER_PTR                      13700000
                OSREQ STORE,MF=(E,PARM_LIST),                     X13750000
                BUFFER64=(R9),                                     @L2AX13760000
                BUFLIST=(R10)                                       13800000
                B      SAVE_RC                                     13850000
                                                                    13900000
* Execute RETRIEVE if FUNCTION_REQUEST == "RETRIEVE"             13950000
TRY_RETRIEVE   DS      0H                                         14000000
                CLC FUNCTION_REQUEST,=CL8'RETRIEVE'              14050000
                BNE TRY_QUERY                                       14100000
*                                                                    14150000
* A partial retrieve can be done to obtain the first xxx bytes of 14200000
* the object. In some cases the application may have some control 14250000
* information in this area to allow retrieval of still another part 14300000
* of the object, (which could be an image).                       14350000
*                                                                    14400000
                L      R9,BUFFER64_PTR_PTR                        @L2A 14410000
                L      R10,RETRIEVE_BUFFER_PTR                  14450000
                OSREQ (RETRIEVE),MF=(M,PARM_LIST),                X14500000
                VIEW=PRIMARY,      Retrieve Primary Copy          X14550000
                BUFFER64=(R9),                                     @L2AX14560000
                BUFLIST=(R10)                                       14600000
                                                                    14650000
*                                                                    14700000
* if view=2, the set VIEW=BACKUP                                  14750000
TRYVIEW2      DS      0H                                         14800000
                SLR     R6,R6      Zero Register                  14850000
                L      R6,VIEW      Load view into R6            @P0C 14900000
                LA     R10,2      Load value 2 into R10          14950000
                CR     R6,R10      Does view = 2?                14950000
                BNE    TRYVIEW3     No, then see if view = 3      15000000
                OSREQ (RETRIEVE),MF=(M,PARM_LIST),                X15050000
                VIEW=BACKUP      Retrieve First Backup Copy      15100000
                B      DO_RETRIEVE      Skip test 'if view=3'     15150000

```

*		15200000
* else if view=3, then set VIEW=BACKUP2		15250000
TRYVIEW3	DS 0H	15300000
	LA R10,3	15350000
	CR R6,R10	15400000
	BNE DO_RETRIEVE	15450000
	OSREQ (RETRIEVE),MF=(M,PARM_LIST),	X15500000
	VIEW=BACKUP2	15550000
	Retrieve First Backup Copy	15600000
* Execute the Retrieve		15650000
DO_RETRIEVE	DS 0H	15700000
	OSREQ RETRIEVE,MF=(E,PARM_LIST)	15750000
	B SAVE_RC	15800000
		15850000
* Execute QUERY if FUNCTION_REQUEST == "QUERY"		15900000
TRY_QUERY	DS 0H	15950000
	CLC FUNCTION_REQUEST,=CL8'QUERY'	16000000
	BNE TRY_CHANGE	16050000
*		16100000
* Query the data base for the directory information that was stored.		16150000
* The size of the object can be extracted from this information so		16200000
* that a GETMAIN can be done for the area necessary for the		16250000
* retrieve operation.		16300000
*		16350000
	L R10,QUERY_BUFFER_PTR	X16400000
	OSREQ QUERY,MF=(E,PARM_LIST),	16450000
	QEL=(R10)	16500000
	B SAVE_RC	16550000
		16600000
* Execute CHANGE if FUNCTION_REQUEST == "CHANGE"		16650000
TRY_CHANGE	DS 0H	16700000
	CLC FUNCTION_REQUEST,=CL8'CHANGE'	16750000
	BNE TRY_DELETE	16800000
*		16850000
* This invocation of the OSREQ macro will change information in the		16900000
* directory that has been specified. A zero pointer in DATAAREA		16950000
* will result in no change for the respective information. All		17000000
* pointers zero result in no change.		17050000
*		17100000
	OSREQ CHANGE,MF=(E,PARM_LIST)	17150000
	B SAVE_RC	17200000
*		17250000
* Execute DELETE if FUNCTION_REQUEST == "DELETE"		17300000
TRY_DELETE	DS 0H	17350000
	CLC FUNCTION_REQUEST,=CL8'DELETE'	17400000
	BNE TRY_UNACCESS	17450000
*		17500000
* This invocation will delete the object named from the object table		17550000
* and the directory.		17600000
*		17650000
	OSREQ DELETE,MF=(E,PARM_LIST)	17700000
	B SAVE_RC	17750000
*		17800000
* Execute UNACCESS if FUNCTION_REQUEST == "UNACCESS"		17850000
TRY_UNACCESS	DS 0H	17900000
	CLC FUNCTION_REQUEST,=CL8'UNACCESS'	17950000
	BNE TRY_STOREBEG	18000000
*		18050000
* The logical connection to OAM should be broken before the TASK		18100000
* terminates so that OAM can remove any system control blocks		18150000
* that it built during ACCESS		18200000
*		18250000
	OSREQ UNACCESS,MF=(E,PARM_LIST)	18300000
	B SAVE_RC	18350000
*		18400000
* Execute STOREBEG if FUNCTION_REQUEST == "STOREBEG"		18450000
TRY_STOREBEG	DS 0H	18500000
	CLC FUNCTION_REQUEST,=CL8'STOREBEG'	18550000
	BNE TRY_STOREPRT	1856200
*		18562400
	ICM R9,15,STIMEOUT_PTR	@P2A 18568600
	BZ DO_STOREBEG	@P2A 18574800
	No	@P2AX18581000
*		@P2A 18587200
	OSREQ STOREBEG,MF=(M,PARM_LIST),	@P2A 18593400
	STIMEOUT=(R9)	18600000
DO_STOREBEG	DS 0H	X18650000
* Begin the sequential storage of an object in parts.		18700000
	OSREQ STOREBEG,MF=(E,PARM_LIST),	18750000
	STOKEN=STOKEN_AREA	18800000
	B SAVE_RC	18850000
*		18900000
* Execute STOREPRT if FUNCTION_REQUEST == "STOREPRT"		18900000
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 STORPRT,MF=(E,PARM_LIST),
        BUFLIST=(R9),
        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),
        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),
        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        3
OSR2_REL      EQU        0
*
* 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	23050000
		23100000
*		23150000
* All literals will be included at this point.		23200000
*		23250000
LTORG		23300000
*		23350000
* This static parameter list will be used as a template for		23400000
* OSREQ invocations in the executable code.		23450000
*		23500000
STATIC_PARM_LIST OSREQ (STORE),MF=(L)		23550000
STATIC_LIST_END EQU *		23600000
*		23650000
* This area is provided by the caller of this routine		23700000
*		23750000
DATAAREA DSECT		23800000
*****		23850000
*		23900000
* Th DATAAREA must be obtained by the caller of OSR2 and presented		23950000
* as a parameter (R1) to OSR2. It is expected that all subsequent		24000000
* calls will point to this same area. There is information in the		24050000
* area that will be used across calls.		24100000
*		24150000
*****		24200000
*		24250000
* DATAAREA Header		24300000
DA_ID DS CL4 x0 identifier		24350000
DA_LEN DS F x4 DATAAREA length--x280 (640) @01C		24400000
DA_VER DS X x8 DATAAREA version		24450000
DA_REL DS X x9 DATAAREA release		24500000
DS CL6 xA Reserved		24550000
		24600000
*****		24650000
* The following two named fields are set by the caller of OSR2.		24700000
* If the value in the field is not a valid value, the respective		24750000
* activity cannot be executed.		24800000
*****		24850000
FUNCTION_REQUEST DS CL8 x10 OSREQ function request value		24900000
*		24950000
ACCESS, STORE, etc. or other		25000000
DS CL8 x18 Reserved		25050000
*****		25100000
* The following fields are set by OSR2 and should not be		25150000
* altered by the caller. Subsequent calls to OSR2 will rely		25200000
* on the information stored here.		25250000
*		25300000
* STOKEN NOTE: The STOKEN must be kept on a DOUBLE WORD boundary		25350000
*****		25400000
TOKEN_AREA DS 2F x20 OSREQ token, do not change it.		25450000
STOKEN_AREA DS 4F x28 OSREQ stoken, do not change it.		25500000
DS 8F x38 Reserved		25550000
*****		25600000
* The following fields are set by the caller of OSR2.		25650000
* The pointers are not altered by OSR2 but the data that		25700000
* the pointers reference may be.		25750000
*****		25800000
*		25850000
CANCEL DS CL3 x58 CANCEL value, YES or other		25900000
DS CL1 x5B Reserved		25950000
COLLECTION_NAME_PTR DS A x5C Pointer to collection name		26000000
IADDRESS_PTR DS A x60 Reserved for IADDRESS_PTR		26050000
MANAGEMENT_CLASS_PTR DS A x64 Pointer to management class parm		26100000
MESSAGE_AREA_PTR DS A x68 Pointer to message area		26150000
OBJECT_NAME_PTR DS A x6C Pointer to object name		26200000
OBJECT_SIZE_PTR DS A x70 Pointer to object size value		26250000
OFFSET_PTR DS A x74 Pointer to offset value		26300000
QUERY_BUFFER_PTR DS A x78 Pointer to query buffer list		26350000
REASON_CODE_PTR DS A x7C Pointer to OSREQ reason code		26400000
RECALL_NUM_DAYS_PTR DS A x80 Recall Number of Days Pointer		26450000
RELEASE_BUFFER DS CL3 x84 RELBUF value, YES or other		26500000
DS CL1 x87 Reserved		26550000
RETENTION_PERIOD_PTR DS A x88 Pointer to retention period		26600000
RETRIEVE_LENGTH_PTR DS A x8C Pointer to retrieve length value		26650000
RETRIEVE_BUFFER_PTR DS A x90 Pointer to retrieve buffer list		26700000
RETURN_CODE_PTR DS A x94 Pointer to OSREQ return code		26750000
RETURN_CODE2_PTR DS A x98 Return Code 2 Pointer		26800000
STIMEOUT_PTR DS A x9C Store Timeout Pointer @P2C		26850000
STORE_BUFFER_PTR DS A xA0 Pointer to store buffer list		26900000
STORAGE_CLASS_PTR DS A xA4 Pointer to storage class parameter		26950000
TRACKING_TOKEN_PTR DS A xA8 User Tracking Token Pointer		27000000
VIEW DS F xAC Retrieve Object Copy		27050000
*	1 = PRIMARY	27100000
*	2 = First BACKUP Copy	

```

*                               3 = Second BACKUP Copy                27150000
DELHOLD                        DS CL8    xB0 DELHOLD= HOLD | NOHOLD | blank @L1A 27180000
EVENTEXP_PTR                  DS A      xB8 Pointer to EVENTEXP           @L1A 27210000
OBJECT_SIZE64_PTR             DS A      xBC Pointer to 64-bit object size  @L2A 27211000
OFFSET64_PTR                  DS A      xC0 Pointer to 64-bit offset       @L2A 27212000
RETRIEVE_LENGTH64_PTR         DS A      xC4 Pointer to 64-bit retrieve len  @L2A 27213000
BUFFER64_PTR_PTR              DS A      xC8 Pointer to 64-bit buffer addr  @L2A 27214000
                               DS CL108  xCC Reserved for future keywords    @L2C 27240000
*                               27270000
* Register Save Area
SAVE_AREA                      DS 18F    x138 Savearea for this module.    @01C 27300000
*                               27350000
*                               27400000
*****                          27450000
* The following area is completely overlaid each time OSR2
* is called                      27500000
*****                          27550000
*****                          27600000
PARAM_LIST DS CL(STATIC_LIST_END-STATIC_PARAM_LIST) x180 Dynamic
*                               parm list @01C 27650000
*                               27700000
*                               27750000
*                               @L3D 27750100
* Dynamic reserved storage calculation for DATAAREA
USEDLEN EQU *-DATAAREA          @L3A 27750200
RESERVED DS CL(640-USEDLEN)     @L3A 27750300
DATAAREA_LEN EQU *-DATAAREA     @L3A 27850000
OSR2SAMP CSECT                  27900000
*                               27950000
*                               28000000
*                               28050000
END OSR2SAMP

```

CBROS3

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

```

***** 00050000
* 00100000
* DESCRIPTIVE NAME: Object Storage Request Sample Interface #3 for 00150000
* Multiple OAM Address Spaces 00200000
* 00200001
* (Modeled after CBROS2 V2R2) 00200002
* 00200003
* Licensed Materials - Property of IBM 00200005
* 5650-ZOS 00200050
* COPYRIGHT IBM CORP. 2017 00200500
* 00205000
* FUNCTION: Provides a generalized interface for the Object Storage 00250000
* Request (OSREQ) macro in a Multiple OAM environment. 00300000
* 00350000
* OPERATION: This routine is called with a parameter area that 00400000
* defines the function and pointers necessary to invoke 00450000
* the OSREQ macro. 00500000
* 00550000
* If it is determined that an OSREQ function is requested, 00600000
* then the OSREQ parameter list is filled in with an 00650000
* MF=M form of the macro. The function is executed via an 00700000
* MF=E form. 00750000
* 00800000
* 1. Validity check the DATAAREA Header. Exit if error. 00850000
* 2. Fill in the OSREQ PARAM_LIST with all of the optional 00900000
* keywords using MF=M form of the macro. 00950000
* 3. If FUNCTION_REQUEST = "ACCESS " 01000000
* a. IF CBROS3 was compiled with IADD option, then 01050000
* set IADDRESS OSREQ macro keyword to the address of 01100000
* the DB2 library entry point DSNHLI using the MF=M 01150000
* form of the macro. 01175000
* b. ELSE set IADDRESS OSREQ macro keyword to 01200000
* IADDRESS_PTR and set DB2ID OSREQ macro keyword 01225000
* to DB2ID_PTR using the MF=M form of the macro 01225001
* 4. SELECT FUNCTION_REQUEST 01250000
* WHEN(ACCESS, STORE, RETRIEVE, QUERY, CHANGE, DELETE, 01300000
* UNACCESS, STOREBEG, STOREPRT, STOREEND) 01350000
* a. Set any function specific keywords 01400000
* b. Execute specified function using the MF=E form 01450000
* of the macro. 01500000
* c. Set R15 to 0, to indicate successful OSREQ 01550000
* macro invocation 01600000
* OTHERWISE: 01650000
* a. Set R15 to Invalid Function Request 01700000
* 5. Return to caller 01750000
* 01800000
* Valid values for FUNCTION_REQUEST: 01850000

```

```

* "ACCESS " : OSREQ ACCESS 01900000
* "STORE " : OSREQ STORE 01950000
* "RETRIEVE" : OSREQ RETRIEVE 02000000
* "QUERY " : OSREQ QUERY 02050000
* "CHANGE " : OSREQ CHANGE 02100000
* "DELETE " : OSREQ DELETE 02150000
* "UNACCESS" : OSREQ UNACCESS 02200000
* "STOREBEG" : OSREQ STOREBEG 02250000
* "STOREPRT" : OSREQ STOREPRT 02300000
* "STOREEND" : OSREQ STOREEND 02350000
* 02400000
* 02450000
* IADDRESS NOTE: 02500000
* To specify the default DSNHLI entry point for the 02540000
* IADDRESS keyword in the OSREQ function, specify 02580000
* the SYSPARM value as IADD in the PARM field of 02620000
* the EXEC JCL statement. For example: 02660000
* 02700000
* //ASSEMBLE EXEC PGM=ASMA90,PARM='RENT,DECK,SYSPARM(IADD)' 02750000
* 02800000
* REGISTER CONVENTIONS: 02850000
* R0 - WORK REGISTER 02900000
* R1 - STANDARD LINKAGE REGISTER 02950000
* - PARAMETER LIST ADDRESS 03000000
* R2 - WORK REGISTER 03050000
* R3 - WORK REGISTER 03100000
* R4 - WORK REGISTER 03150000
* R5 - WORK REGISTER 03200000
* R6 - WORK REGISTER 03250000
* R7 - WORK REGISTER 03300000
* R8 - WORK REGISTER 03350000
* R9 - WORK REGISTER 03400000
* R10 - WORK REGISTER 03450000
* R11 - DATAAREA BASE REGISTER 03500000
* R12 - CBROSR3 BASE REGISTER 03550000
* R13 - STANDARD LINKAGE REGISTER 03600000
* - SAVE AREA ADDRESS 03650000
* R14 - STANDARD LINKAGE REGISTER 03700000
* - RETURN POINT ADDRESS 03750000
* R15 - STANDARD LINKAGE REGISTER 03800000
* - ENTRY POINT ADDRESS 03850000
* - RETURN CODE 03900000
* 03950000
* INPUT: Register 1 must point to a 4 byte field that contains 04000000
* an address of an area that is described by 04050000
* the dsect named DATAAREA in this program. 04100000
* The DATAAREA must be filled in to indicate 04150000
* the function requested and provide the proper 04200000
* data for execution of the OSREQ macro. 04250000
* Register 13 must point to a 72 byte area into which this 04300000
* routine will save the registers at entry and 04350000
* from which registers will be restore at exit. 04400000
* Register 14 must point to the instruction address to which 04450000
* this routine will return. 04500000
* Register 15 must point to the entry point address of this 04550000
* routine. 04600000
* OUTPUT: Register 15 will contain the return code from DATAAREA 04650000
* validity checking. 04700000
* CODE MEANING 04750000
* 0 SUCCESS--OSREQ Function invoked 04800000
* 6 Invalid DATAAREA FUNCTION_REQUEST 04850000
* 8 Invalid DATAAREA hdr ID 04900000
* 10 Invalid DATAAREA hdr length 04950000
* 12 Invalid DATAAREA hdr version 05000000
* 14 Invalid DATAAREA hdr release 05050000
* 05100000
* Fields pointed to by REASON_CODE_PTR and RETURN_CODE_PTR 05150000
* will contain the reason and return codes returned 05200000
* from OAM for OSREQ function requests. 05250000
* Areas defined by the CBRIBUFL (for retrieve) and CBRIQEL 05300000
* (for query) will be filled in when the respective 05350000
* function is requested. 05400000
* 05450000
* CHANGE-ACTIVITY: 05500000
* $L0=OAMR23M R23 150914 TUCAED: Multi OAM AS STAGE 1 @L0A 05598801
* 05600000
* ***** 05650000
* CBROSR3 CSECT , 05700000
* CBROSR3 AMODE 31 05750000
* CBROSR3 RMODE ANY 05800000
* USING *,R15 USING to allow branch to 05850000
* STRTOSR3 05900000

```

	SPACE 2		05950000
	B	STRTOSR3	06000000
LENGOSR3	DC	X'18'	06050000
NAMEOSR3	DC	CL8'CBROS3 '	06100000
DATEOSR3	DC	CL8'&SYSDATE'	06150000
APAROSR3	DC	CL8'HDZ2250 '	06200000
	DROP	R15	06250000
	SPACE 2		06300000
STRTOSR3	DS	0H	06350000
*		START THE ACTIVE PART OF MOD	06400000
	STM	R14,R12,12(R13)	06450000
*			06500000
*		Register 12 is the base for the code	06550000
*			06600000
	LR	R12,R15	06650000
	USING	CBROS3,R12	06700000
*			06750000
*		Register 11 is the base for the data area which is passed to this	06800000
*		routine as a parameter.	06850000
*			06900000
	L	R11,0(R1)	06950000
	USING	DATAAREA,R11	07000000
	LA	R15,SAVE_AREA	07050000
	ST	R15,8(R13)	07100000
	ST	R13,SAVE_AREA+4	07150000
	LR	R13,R15	07200000
*			07250000
*		The static OSREQ parameter list is copied into the work area	07300000
*			07350000
	MVC	PARM_LIST,STATIC_PARM_LIST	07400000
			07450000
*			07500000
*		Do some DATAAREA Header Validity Checking	07550000
*			07600000
			07650000
*		Make sure the ID of the user's dataarea = current OSR3 ID	07700000
	LA	R15,ERR_ID	07750000
	CLC	DA_ID,=CL4'OSR3'	07800000
	BNE	EXIT	07850000
		Exit if not equal	07900000
*		Make sure the length of the user's dataarea = current OSR3 length	07950000
	LA	R15,ERR_LEN	08000000
	L	R0,DA_LEN	08050000
	CFI	R0,DATAAREA_LEN	08100000
	BNE	EXIT	08150000
		Exit if not equal	08200000
*		Make sure user's dataArea version is <= current OSR3 version	08250000
	LA	R15,ERR_VER	08300000
	SLR	R2,R2	08350000
	IC	R2,DA_VER	08400000
	LA	R3,OSR3_VER	08450000
	CR	R3,R2	08500000
	BL	EXIT	08533300
*		Exit w/ err if VERSION	08566600
		< DA_VER	08600000
*		Make sure user's dataArea release is <= current OSR3 release	08650000
	LA	R15,ERR_REL	08700000
	SLR	R2,R2	08750000
	IC	R2,DA_REL	08800000
	LA	R3,OSR3_REL	08850000
	CR	R3,R2	08900000
	BL	EXIT	08933300
*		Exit w/ err if RELEASE	08966600
*		< DA_REL	09000000
*		Modify the parameter list to establish all the basic OSREQ function	09050000
*		parameters.	09100000
*			09150000
*		Note: A pointer with a value of zero is equivalent to an omitted parm	09200000
*			09250000
OSR_FUNC	DS	0H	09300000
	L	R0, COLLECTION_NAME_PTR	09350000
	L	R2, MANAGEMENT_CLASS_PTR	09400000
	L	R3, MESSAGE_AREA_PTR	09450000
	L	R4, OBJECT_NAME_PTR	09500000
	L	R5, OBJECT_SIZE_PTR	09550000
	L	R6, OFFSET_PTR	09600000
	L	R7, REASON_CODE_PTR	09650000
	L	R8, RECALL_NUM_DAYS_PTR	09700000
	L	R10, RETRIEVE_LENGTH_PTR	09850000
*			09900000
	OSREQ (STORE),MF=(M,PARM_LIST),		X09950000
	TOKEN=TOKEN_AREA,	Contains logical OAM connection	X10000000

COLLECTN=(R0),		X10050000
MGMTCLAS=(R2),		X10100000
MSGAREA=(R3),	DB2 error messages returned here	X10150000
NAME=(R4),		X10200000
SIZE=(R5),		X10250000
OFFSET=(R6),	Starting byte for retrieve	X10300000
REACODE=(R7),	Register 0 is stored here	X10350000
RECALL=(R8),	Recall Number of Days	X10400000
LENGTH=(R10)	Length of retrieve	10500000
* Ran out of registers above -- add remaining PTRs		10550000
L R0,RETURN_CODE_PTR		10600000
L R2,RETURN_CODE2_PTR		10650000
L R3,STORAGE_CLASS_PTR		10700000
L R4,TRACKING_TOKEN_PTR		10750000
L R5,OBJECT_SIZE64_PTR		10750050
L R6,OFFSET64_PTR		10750500
L R7,RETRIEVE_LENGTH64_PTR		10755000
*		10800000
OSREQ (STORE),MF=(M,PARM_LIST),		X10850000
RETCODE=(R0),	Register 15 is stored here	X10900000
RETCODE2=(R2),	Return Code 2	X10950000
STORCLAS=(R3),		X11000000
TOKEN=(R4),	User Tracking Token	X11050000
SIZE64=(R5),		X11050050
OFFSET64=(R6),		X11050500
LENGTH64=(R7)		11055000
*		11100000
* Set RELBUF=YES if DATAAREA RELEASE_BUFFER == "YES"		11150000
TRYRELBUFF DS 0H		11200000
CLC RELEASE_BUFFER,=CL3'YES'		11250000
BNE BUFDONE	RELBUFF=NO is default	11300000
OSREQ (STORE),MF=(M,PARM_LIST),		X11350000
RELBUFF=YES	Will release pages after STORE	11400000
BUFDONE DS 0H		11402700
*		11405400
* Set RETPD or EVENTEXP or both, based on caller's parm list.		11408100
*		11410800
* Note that a runtime error will occur if non-zero pointers are		11413500
* present for both RETPD and EVENTEXP. Supplying both RETPD and		11416200
* EVENTEXP is generally only useful for testing the error checking		11418900
* features of the OSREQ processing code.		11421600
*		11424300
L R9,RETENTION_PERIOD_PTR		11427000
OSREQ (STORE),MF=(M,PARM_LIST),		X11429700
RETPD=(R9)		11432400
*		11435100
L R9,EVENTEXP_PTR		11437800
OSREQ (CHANGE),MF=(M,PARM_LIST),	EVENTEXP only on CHANGE	X11440500
EVENTEXP=(R9)		11443200
*		11445900
* Set the DELHOLD parm or leave it off.		11448600
*		11451300
DELHCHK DS 0H		11454000
CLC DELHOLD,=CL8'HOLD'		11456700
BE DELHYES		11459400
*		11462100
CLC DELHOLD,=CL8'NOHOLD'		11464800
BE DELHNO		11467500
B DELHDONE		11470200
*		11472900
DELHNO DS 0H		11475600
OSREQ (STORE),MF=(M,PARM_LIST),		X11478300
DELHOLD=NOHOLD		11481000
B DELHDONE		11483700
*		11486400
DELHYES DS 0H		11489100
OSREQ (STORE),MF=(M,PARM_LIST),		X11491800
DELHOLD=HOLD		11494500
DELHDONE DS 0H		11497200
*		11500000
* Keep testing FUNCTION_REQUEST until an OSREQ FUNCTION match is found		11550000
* or no more functions are found		11600000
* If a match is found, then go ahead and execute that function		11650000
*		11700000
		11750000
* Execute ACCESS if FUNCTION_REQUEST == "ACCESS"		11800000
TRY_ACCESS DS 0H		11850000
CLC FUNCTION_REQUEST,=CL8'ACCESS'		11900000
BNE TRY_STORE		11950000
*		12000000
* The logical connection to OAM is made here.		12050000
* If this is MVS batch, the Call Attach Facility will be used		12100000

```

* to connect to DB2, and a thread will be OPENed to Plan(CBRIDBS) 12150000
* otherwise, the connection is done by the environment in which 12200000
* this program is executing. 12250000
* In all cases system control blocks will be created and/or modified 12300000
* to provide this access to OAM. 12350000
* 12400000
* To specify the default DSNHLI entry point for the 12440000
* IADDRESS keyword in the OSREQ function, specify 12480000
* the SYSPARM value as IADD in the PARM field of 12520000
* the EXEC JCL statement. See NOTE in prolog. 12560000
* 12600000
* AIF ('&SYSPARM' EQ 'IADD').IA2 12650000
* 12660000
* IADD not specified, so set IADDRESS OSREQ macro keyword to 12670000
* IADDRESS_PTR using the MF=M form of the macro. 12680000
* 12690000
* L R2,IADDRESS_PTR Load IADR from parmList 12700000
* L R3,DB2ID_PTR Load DB2ID from parmList 12705000
* OSREQ ACCESS,MF=(E,PARM_LIST), X12750000
* IADDRESS=(R2), X12800000
* DB2ID=(R3) 12805000
* AGO .SKIP1 12850000
.IA2 ANOP 12900000
* IADD was specified so set default entry point. 12925000
* In this sample we use DSNHLI for SQL interface module to DB2 12950000
* 12975000
* L R2,=V(DSNHLI) 13000000
* OSREQ ACCESS,MF=(E,PARM_LIST), X13050000
* IADDRESS=(R2) 13100000
.SKIP1 ANOP 13150000
* B SAVE_RC 13200000
* 13250000
* Execute STORE if FUNCTION_REQUEST == "STORE" 13300000
TRY_STORE DS 0H 13350000
CLC FUNCTION_REQUEST,=CL8'STORE' 13400000
BNE TRY_RETRIEVE 13450000
* 13500000
* This will store an object in the DB2 object tables or on 13550000
* an optical disk, depending on the storage class specified. 13600000
* 13650000
* L R9,BUFFER64_PTR_PTR 13655000
* L R10,STORE_BUFFER_PTR 13700000
* OSREQ STORE,MF=(E,PARM_LIST), X13750000
* BUFFER64=(R9), X13755000
* BUFLIST=(R10) 13800000
* B SAVE_RC 13850000
* 13900000
* Execute RETRIEVE if FUNCTION_REQUEST == "RETRIEVE" 13950000
TRY_RETRIEVE DS 0H 14000000
CLC FUNCTION_REQUEST,=CL8'RETRIEVE' 14050000
BNE TRY_QUERY 14100000
* 14150000
* A partial retrieve can be done to obtain the first xxx bytes of 14200000
* the object. In some cases the application may have some control 14250000
* information in this area to allow retrieval of still another part 14300000
* of the object, (which could be an image). 14350000
* 14400000
* L R9,BUFFER64_PTR_PTR 14450000
* L R10,RETRIEVE_BUFFER_PTR 14455000
* OSREQ (RETRIEVE),MF=(M,PARM_LIST), X14500000
* VIEW=PRIMARY, Retrieve Primary Copy X14550000
* BUFFER64=(R9), X14555000
* BUFLIST=(R10) 14600000
* 14650000
* if view=2, the set VIEW=BACKUP 14700000
TRYVIEW2 DS 0H 14750000
SLR R6,R6 Zero Register 14800000
L R6,VIEW Load view into R6 14850000
LA R10,2 Load value 2 into R10 14900000
CR R6,R10 Does view = 2? 14950000
BNE TRYVIEW3 No, then see if view = 3 15000000
* OSREQ (RETRIEVE),MF=(M,PARM_LIST), X15050000
* VIEW=BACKUP Retrieve First Backup Copy 15100000
* B DO_RETRIEVE Skip test 'if view=3' 15150000
* 15200000
* else if view=3, then set VIEW=BACKUP2 15250000
TRYVIEW3 DS 0H 15300000
LA R10,3 Load value 3 into R10 15350000
CR R6,R10 Does view = 3? 15400000
BNE DO_RETRIEVE Nope, so leave VIEW=PRIMARY 15450000
* OSREQ (RETRIEVE),MF=(M,PARM_LIST), X15500000
* VIEW=BACKUP2 Retrieve First Backup Copy 15550000

```

* Execute the Retrieve		15600000
DO_RETRIEVE	DS 0H	15650000
OSREQ RETRIEVE,MF=(E,PARM_LIST)		15700000
B SAVE_RC		15750000
		15800000
* Execute QUERY if FUNCTION_REQUEST == "QUERY"		15850000
TRY_QUERY	DS 0H	15900000
CLC FUNCTION_REQUEST,=CL8'QUERY'		15950000
BNE TRY_CHANGE		16000000
		16050000
* Query the data base for the directory information that was stored.		16100000
* The size of the object can be extracted from this information so		16150000
* that a GETMAIN can be done for the area necessary for the		16200000
* retrieve operation.		16250000
		16300000
		16350000
L R10,QUERY_BUFFER_PTR		16350000
OSREQ QUERY,MF=(E,PARM_LIST),		X16400000
QEL=(R10)		16450000
B SAVE_RC		16500000
		16550000
* Execute CHANGE if FUNCTION_REQUEST == "CHANGE"		16600000
TRY_CHANGE	DS 0H	16650000
CLC FUNCTION_REQUEST,=CL8'CHANGE'		16700000
BNE TRY_DELETE		16750000
		16800000
* This invocation of the OSREQ macro will change information in the		16850000
* directory that has been specified. A zero pointer in DATAAREA		16900000
* will result in no change for the respective information. All		16950000
* pointers zero result in no change.		17000000
		17050000
OSREQ CHANGE,MF=(E,PARM_LIST)		17100000
B SAVE_RC		17150000
		17200000
* Execute DELETE if FUNCTION_REQUEST == "DELETE"		17250000
TRY_DELETE	DS 0H	17300000
CLC FUNCTION_REQUEST,=CL8'DELETE'		17350000
BNE TRY_UNACCESS		17400000
		17450000
* This invocation will delete the object named from the object table		17500000
* and the directory.		17550000
		17600000
OSREQ DELETE,MF=(E,PARM_LIST)		17650000
B SAVE_RC		17700000
		17750000
* Execute UNACCESS if FUNCTION_REQUEST == "UNACCESS"		17800000
TRY_UNACCESS	DS 0H	17850000
CLC FUNCTION_REQUEST,=CL8'UNACCESS'		17900000
BNE TRY_STOREBEG		17950000
		18000000
* The logical connection to OAM should be broken before the TASK		18050000
* terminates so that OAM can remove any system control blocks		18100000
* that it built during ACCESS		18150000
		18200000
OSREQ UNACCESS,MF=(E,PARM_LIST)		18250000
B SAVE_RC		18300000
		18350000
* Execute STOREBEG if FUNCTION_REQUEST == "STOREBEG"		18400000
TRY_STOREBEG	DS 0H	18450000
CLC FUNCTION_REQUEST,=CL8'STOREBEG'		18500000
BNE TRY_STOREPRT		18550000
		18556200
ICM R9,15,STIMEOUT_PTR Any STIMEOUT value?		18562400
BZ DO_STOREBEG No		18568600
		18574800
OSREQ STOREBEG,MF=(M,PARM_LIST),		X18581000
STIMEOUT=(R9)		18587200
DO_STOREBEG	DS 0H	18593400
* Begin the sequential storage of an object in parts.		18600000
OSREQ STOREBEG,MF=(E,PARM_LIST),		X18650000
STOKEN=STOKEN_ARÉA		18700000
B SAVE_RC		18750000
		18800000
* Execute STOREPRT if FUNCTION_REQUEST == "STOREPRT"		18850000
TRY_STOREPRT	DS 0H	18900000
CLC FUNCTION_REQUEST,=CL8'STOREPRT'		18950000
BNE TRY_CANCEL		19000000
* Store the next sequential contiguous part of an object		19050000
L R9,STORE_BUFFER_PTR		19100000
OSREQ STOREPRT,MF=(E,PARM_LIST),		X19150000
BUFLIST=(R9),		X19200000
STOKEN=STOKEN_ARÉA		19250000
B SAVE_RC		19300000


```

*
19350000
19400000
* Set CANCEL=YES if DATAAREA CANCEL == "YES"
19450000
TRY_CANCEL DS 0H
19500000
CLC CANCEL,=CL3'YES'
19550000
BNE TRY_STOREEND CANCEL=NO is default
19600000
OSREQ (STOREEND),MF=(M,PARM_LIST),
X19650000
CANCEL=YES Will CANCEL Store Sequence
19700000
* Execute STOREEND if FUNCTION_REQUEST == "STOREEND"
19750000
TRY_STOREEND DS 0H
19800000
CLC FUNCTION_REQUEST,=CL8'STOREEND'
19850000
BNE INVALID_FUNC
19900000
* End the sequential storage of an object in parts.
19950000
* L R10,CANCEL
20000000
OSREQ STOREEND,MF=(E,PARM_LIST),
X20050000
STOKEN=STOKEN_AREA
20100000
B SAVE_RC
20150000
*
20200000
* None of the OSREQ functions matched FUNCTION_REQUEST, so set error
20250000
INVALID_FUNC DS 0H
20300000
LA R15,ERR_FUNC Set invalid function request
20350000
B EXIT
20400000
*
20450000
* Save register 15 in the return code area and register 0 in the
20500000
* reason code area after return from OSREQ. This is recommended
20550000
* because, under certain error conditions, the return code and
20600000
* reason code areas may not be set by OSREQ.
20650000
*
20700000
SAVE_RC DS 0H
20750000
L R2,RETURN_CODE_PTR
20800000
L R3,REASON_CODE_PTR
20850000
ST R15,0(,R2) Save Return Code to RETURN_CODE_PTR
20900000
ST R0,0(,R3) Save Reason Code to REASON_CODE_PTR
20950000
LA R15,0 Reset R15 back to zero to indicate
21000000
that the osreq function was
21050000
invoked
21100000
*
21150000
* Restore all registers except regs 15 and 0, then return to caller
21200000
EXIT DS 0H
21250000
L R13,SAVE_AREA+4
21300000
L R14,12(R13)
21350000
LM R1,R12,24(R13)
21400000
BR R14
21450000
*
21500000
* Register definitions
21550000
*
21600000
R0 EQU 0
21650000
R1 EQU 1
21700000
R2 EQU 2
21750000
R3 EQU 3
21800000
R4 EQU 4
21850000
R5 EQU 5
21900000
R6 EQU 6
21950000
R7 EQU 7
22000000
R8 EQU 8
22050000
R9 EQU 9
22100000
R10 EQU 10
22150000
R11 EQU 11
22200000
R12 EQU 12
22250000
R13 EQU 13
22300000
R14 EQU 14
22350000
R15 EQU 15
22400000
*
22450000
* Header Constants
22500000
*
22550000
*OSR3_ID EQU "OSR3"
22600000
OSR3_VER EQU 1
22650000
OSR3_REL EQU 0
22700000
*
22750000
* Header Validity Checking Error Codes
22800000
ERR_FUNC EQU 6 Invalid Function Request
22850000
ERR_ID EQU 8 Invalid Header ID
22900000
ERR_LEN EQU 10 Invalid Header Length
22950000
ERR_VER EQU 12 Invalid Header Version
23000000
ERR_REL EQU 14 Invalid Header Release
23050000
*
23100000
*
23150000
* All literals will be included at this point.
23200000
*
23250000
LTOrg
23300000
*
23350000
* This static parameter list will be used as a template for
23400000

```

* OSREQ invocations in the executable code.		23450000
* STATIC_PARM_LIST OSREQ (STORE),MF=(L)		23500000
STATIC_LIST_END EQU *		23550000
* This area is provided by the caller of this routine		23600000
* DATAAREA DSECT		23650000
*****		23700000
* The DATAAREA must be obtained by the caller of OSR3 and presented		23750000
* as a parameter (R1) to OSR3. It is expected that all subsequent		23800000
* calls will point to this same area. There is information in the		23850000
* area that will be used across calls.		23900000
*****		23950000
* DATAAREA Header		24000000
DA_ID DS CL4 x0 identifier		24050000
DA_LEN DS F x4 DATAAREA length--x280 (640)		24100000
DA_VER DS X x8 DATAAREA version		24150000
DA_REL DS X x9 DATAAREA release		24200000
DS CL6 xA Reserved		24250000
*****		24300000
* The following two named fields are set by the caller of OSR3.		24350000
* If the value in the field is not a valid value, the respective		24400000
* activity cannot be executed.		24450000
*****		24500000
FUNCTION_REQUEST DS CL8 x10 OSREQ function request value		24550000
* ACCESS, STORE, etc. or other		24600000
DS CL8 x18 Reserved		24650000
*****		24700000
* The following fields are set by OSR3 and should not be		24750000
* altered by the caller. Subsequent calls to OSR3 will rely		24800000
* on the information stored here.		24850000
* STOKEN NOTE: The STOKEN must be kept on a DOUBLE WORD boundary		24900000
*****		24950000
TOKEN_AREA DS 2F x20 OSREQ token, do not change it.		25000000
STOKEN_AREA DS 4F x28 OSREQ token, do not change it.		25050000
DS 8F x38 Reserved		25100000
*****		25150000
* The following fields are set by the caller of OSR3.		25200000
* The pointers are not altered by OSR3 but the data that		25250000
* the pointers reference may be.		25300000
*****		25350000
* CANCEL DS CL3 x58 CANCEL value, YES or other		25400000
DS CL1 x5B Reserved		25450000
COLLECTION_NAME_PTR DS A x5C Pointer to collection name		25500000
IADDRESS_PTR DS A x60 Reserved for IADDRESS_PTR		25550000
MANAGEMENT_CLASS_PTR DS A x64 Pointer to management class parm		25600000
MESSAGE_AREA_PTR DS A x68 Pointer to message area		25650000
OBJECT_NAME_PTR DS A x6C Pointer to object name		25700000
OBJECT_SIZE_PTR DS A x70 Pointer to object size value		25750000
OFFSET_PTR DS A x74 Pointer to offset value		25800000
QUERY_BUFFER_PTR DS A x78 Pointer to query buffer list		25850000
REASON_CODE_PTR DS A x7C Pointer to OSREQ reason code		25900000
RECALL_NUM_DAYS_PTR DS A x80 Recall Number of Days Pointer		25950000
RELEASE_BUFFER DS CL3 x84 RELBUF value, YES or other		26000000
DS CL1 x87 Reserved		26050000
RETENTION_PERIOD_PTR DS A x88 Pointer to retention period		26100000
RETRIEVE_LENGTH_PTR DS A x8C Pointer to retrieve length value		26150000
RETRIEVE_BUFFER_PTR DS A x90 Pointer to retrieve buffer list		26200000
RETURN_CODE_PTR DS A x94 Pointer to OSREQ return code		26250000
RETURN_CODE2_PTR DS A x98 Return Code 2 Pointer		26300000
STIMEOUT_PTR DS A x9C Store Timeout Pointer		26350000
STORE_BUFFER_PTR DS A xA0 Pointer to store buffer list		26400000
STORAGE_CLASS_PTR DS A xA4 Pointer to storage class parameter		26450000
TRACKING_TOKEN_PTR DS A xA8 User Tracking Token Pointer		26500000
VIEW DS F xAC Retrieve Object Copy		26550000
* 1 = PRIMARY		26600000
* 2 = First BACKUP Copy		26650000
* 3 = Second BACKUP Copy		26700000
DELHOLD DS CL8 xB0 DELHOLD= HOLD NOHOLD blank		26750000
EVENEXP_PTR DS A xB8 Pointer to EVENEXP		26800000
OBJECT_SIZE64_PTR DS A xBC Pointer to 64-bit object size		26850000
OFFSET64_PTR DS A xC0 Pointer to 64-bit offset		26900000
RETRIEVE_LENGTH64_PTR DS A xC4 Pointer to 64-bit retrieve len		26950000
BUFFER64_PTR_PTR DS A xC8 Pointer to 64-bit buffer addr		27000000
DB2ID_PTR DS A xCC Pointer to DB2ID keyword value		27050000
		27100000
		27150000
		27180000
		27210000
		27215000
		27220000
		27225000
		27230000
		27235000

DS CL104	xD0	Reserved for future keywords	27240000
*			27270000
* Register Save Area			27300000
SAVE_AREA	DS 18F	x138 Savearea for this module.	27350000
*			27400000
*****			27450000
* The following area is completely overlaid each time OSR3			27500000
* is called			27550000
*****			27600000
PARM_LIST	DS CL(STATIC_LIST_END-STATIC_PARM_LIST)	x180 Dynamic	27650000
*		parm list	27700000
*Dynamic reserved storage calculation for DATAAREA			27750000
USEDLEN	EQU *-DATAAREA		27770000
RESERVED	DS CL(640-USEDLEN)		27770001
DATAAREA_LEN	EQU *-DATAAREA		27900000
CBROSR3 CSECT			27950000
*			28000000
END	CBROSR3		28050000

CBROSRSP

Sample program for an object storage request using the OSREQ macro within a Db2 stored procedure environment:

*****	00000100
* CBROSRSP	00000200
*	00000300
* DESCRIPTIVE NAME: Object Storage Request Sample Interface for DB2	00000400
* Stored Procedure Environment (or other RRSAP	00000500
* managed DB2 connections).	00000600
*	00000700
* Licensed Materials - Property of IBM	00000800
* 5650-ZOS	00000900
* COPYRIGHT IBM CORP. 2021	00001000
*	00001100
* FUNCTION: Provides a generalized interface to facilitate invocation	00001200
* of the Object Storage Request (OSREQ) macro within a DB2	00001300
* Stored Procedure environment utilizing an RRSAP connection.	00001400
*	00001500
* DSNHLIR (RRSAP DB2 SQL module) is loaded and then primed	00001600
* to the IADDRESS parameter on an ACCESS request.	00001700
*	00001800
* RESTRICTIONS: Calling application responsible for establishing a	00001900
* RRSAP connection prior to invoking this sample. OAM	00002000
* will use the caller's RRSAP connection.	00002100
*	00002200
* Calling application responsible for COMMITS/ROLLBACKS.	00002300
*	00002400
* Read DB2 references "Application Programming and SQL	00002500
* Guide" and "z/OS Stored Procedures: Through the CALL	00002600
* and Beyond" for information on latest RRSAP	00002700
* restrictions and overall usage.	00002800
*	00002900
* INPUT: Register 1 : must point to a 4 byte field that contains	00003000
* an address to an area that is described by	00003100
* the dsect named DATAAREA in this program.	00003200
* The DATAAREA must be filled in to indicate	00003300
* the function requested and provide the proper	00003400
* data for execution of the OSREQ macro.	00003500
* (See "DATAAREA NOTES" for additional detail)	00003600
*	00003700
* Register 13: must point to a 72 byte area into which this	00003800
* routine will save the registers at entry and	00003900
* from which registers will be restored at exit.	00004000
*	00004100
* Register 14: must point to the instruction address to which	00004200
* this routine will return.	00004300
*	00004400
* Register 15: must point to the entry point address of this	00004500
* routine.	00004600
*	00004700
* OUTPUT: Register 15: will contain the return code from DATAAREA	00004800
* validity checking or will indicate that a	00004900
* function was processed.	00005000
*	00005100
* CODE MEANING	00005200
* 0 Function invoked	00005300
* 6 Invalid DATAAREA FUNCTION_REQUEST	00005400

```

*          8      Invalid DATAAREA hdr ID          00005500
*          10     Invalid DATAAREA hdr length      00005600
*          12     Invalid DATAAREA hdr version     00005700
*
*          00005800
*          00005900
* REGISTER CONVENTIONS: R0      : WORK REGISTER      00006000
*          R1      : STANDARD LINKAGE REGISTER      00006100
*                  PARAMETER LIST ADDRESS          00006200
*          R2-R10   : WORK REGISTER      00006300
*          R11      : DATAAREA BASE REGISTER      00006400
*          R12      : CBROSRSP BASE REGISTER      00006500
*          R13      : STANDARD LINKAGE REGISTER      00006600
*                  : SAVE AREA ADDRESS            00006700
*          R14      : STANDARD LINKAGE REGISTER      00006800
*                  : RETURN POINT ADDRESS          00006900
*          R15      : STANDARD LINKAGE REGISTER      00007000
*                  ENTRY POINT ADDRESS            00007100
*                  RETURN CODE                    00007200
*
*          00007300
* DATAAREA NOTES: 1. Valid values for FUNCTION_REQUEST: 00007400
*
*          "ACCESS  " : OSREQ ACCESS              00007500
*          "STORE   " : OSREQ STORE               00007600
*          "STOREBEG" : OSREQ STOREBEG            00007700
*          "STOREPRT" : OSREQ STOREPRT            00007800
*          "STOREEND" : OSREQ STOREEND            00007900
*          "RETRIEVE" : OSREQ RETRIEVE            00008000
*          "QUERY    " : OSREQ QUERY              00008100
*          "CHANGE   " : OSREQ CHANGE             00008200
*          "DELETE   " : OSREQ DELETE             00008300
*          "UNACCESS" : OSREQ UNACCESS            00008400
*
*          00008500
*          00008600
*          2. It is expected that the same DATAAREA will be 00008700
*             reused across multiple calls of CBROSRSP, therefore 00008800
*             caller must ensure that the DATAAREA integrity 00008900
*             remain intact. Some fields within the DATAAREA 00009000
*             should not be altered in between calls. (Please 00009100
*             see DATAAREA comments for additional detail) 00009200
*
*          00009300
*          3. Areas defined by CBRIBUFL and CBRIQEL will be 00009400
*             filled in when the respective function is 00009500
*             is requested. 00009600
*
*          00009700
*          4. DSNHLIR_EP has been added to store the entry point 00009800
*             address for SQL module DSNHLIR. This field MUST BE 00009900
*             initialized to binary zeroes before first 00010000
*             invocation in order for this sample to issue a 00010100
*             LOAD. 00010200
*
*          00010300
* LOGICFLOW: 1. Validity check the DATAAREA Header. Exit if error. 00010400
*          2. Initialize variables. 00010500
*          3. LOAD DSNHLIR. 00010600
*          4. Fill in the OSREQ PARM_LIST with all of the optional 00010700
*             keywords using MF=M form of the macro. 00010800
*          5. Validate FUNCTION_REQUEST 00010900
*             If function is supported, then 00011000
*             a. Set any function specific keywords 00011100
*             b. Execute specified function 00011200
*             c. Save OSREQ return and reason codes 00011300
*             c. Set R15 to zero to indicate function performed 00011400
*             Else 00011500
*             a. Set R15 to indicate invalid function 00011600
*          6. Restore registers from save area and return to caller 00011700
*
*          00011800
* CHANGE-ACTIVITY: 00011900
*
*          00012000
*          $L0=0A57837 R23 20200224 TUCAED: Initial release 00012100
*          ***** 00012200
* CBROSRSP CSECT , 00012300
* CBROSRSP AMODE 31 00012400
* CBROSRSP RMODE ANY 00012500
*          USING *,R15 Temp basing to branch over eyecat 00012600
*          B STRTOSRSP Branch to start of module 00012700
* LENGOSR DC X'20' Length of eyecatcher information 00012800
* NAMEOSR DC CL8'CBROSRSP' Module name 00012900
* DATEOSR DC CL8'&SYSDATE' Module assembly date 00013000
* RELEOSR DC CL8'HDZ2250 ' Module release 00013100
* APAROSR DC CL8' ' Module maintenance level 00013200
*          DROP R15 Drop temp basing 00013300
*
*          00013400
*          *****
*          MAINLINE START 00013500
*          ***** 00013600

```

```

* Save area management, addressability to code/data area and          00013700
* initialization of RRSAP return/reason codes.                        00013800
*****                                                                00013900
STRTOSRSP DS      0H          Start of mainline                      00014000
          STM    R14,R12,12(R13) Save caller's regs to save area  00014100
          LR     R12,R15       Save entry point addr to R12       00014200
          USING  CBROSRSRSP,R12 Establish basing to mainline       00014300
          L      R11,0(R1)     Load parameter list to R11        00014400
          USING  DATAAREA,R11 Establish basing to parameters     00014500
          LA     R15,SAVE_AREA Load local save area addr to R15  00014600
          ST     R15,8(R13)    Store to save area chain           00014700
          ST     R13,SAVE_AREA+4 Save save area chain to local    00014800
          LR     R13,R15       Establish basing to local SA       00014900
*****                                                                00015000
* DATAAREA header validity checking                                00015100
*****                                                                00015200
          LA     R15,ERR_ID   Load ERR_ID into R15              00015300
          CLC    DA_ID,OSRS_ID "OSRS" eyecatcher in user area?    00015400
          BNE    EXIT        If not present, then exit           00015500
          LA     R15,ERR_LEN   Load ERR_LEN into R15            00015600
          L      R0,DA_LEN     Load expected length to R1         00015700
          CFI    R0,DATAAREA_LEN Length of user area match?      00015800
          BNE    EXIT        If not, then exit                   00015900
          LA     R15,ERR_VER   Load ERR_VER into R15            00016000
          SLR    R2,R2         Zero Register                      00016100
          IC     R2,DA_VER     Load DA_VER into R2               00016200
          LA     R3,OSRS_VER   Load VERSION into R3             00016300
          CR     R3,R2         Does DA_VER = VERSION?            00016400
          BL     EXIT        Exit w/ err if VERSION < DA_VER      00016500
*****                                                                00016600
* Load required RRSAP environment services:                        00016700
*                                                                    00016800
*      DSNHLIR:  Handles SQL calls.                                00016900
*                                                                    00017000
* Note: If services can't be loaded an 806 abend will occur, so be sure 00017100
*       that there is a JOBLIB or STEPLIB pointing to the library that 00017200
*       contains the DSNHLIR load module.                          00017300
*                                                                    00017400
*****                                                                00017500
DSNHLIR DS      0H          Handle DSNHLIR addressability        00017600
          L      R15,DSNHLIR_EP Load user entry point for DSNHLIR 00017700
          LTR    R15,R15      Is it zero?                        00017800
          BNZ    LOADDONE     If EP present, go to LOADDONE       00017900
          LOAD   EP=DSNHLIR   LOAD DSNHLIR                      00018000
          ST     R0,DSNHLIR_EP Save DSNHLIR EP to dataarea       00018100
LOADDONE DS      0H          Done loading RRSAP environment      00018200
*****                                                                00018300
* The OSREQ parameter list is MODIFIED to establish all of the basic  00018400
* parameters for all of the OSREQ functions.                        00018500
*                                                                    00018600
* Note: A pointer with a value of zero is equivalent to an omitted    00018700
*       parameter.                                                  00018800
* Note: Modify for STORE broken up into two parts due to number of    00018900
*       work registers available.                                    00019000
* Note: A runtime error will occur if non-zero pointers are present for 00019100
*       both RETPD and EVENTEXP. Supplying both RETPD and EVENTEXP is 00019200
*       generally only useful for testing the error checking features  00019300
*       of the OSREQ processing code.                                00019400
*****                                                                00019500
          MVC    PARM_LIST,STATIC_PARM_LIST Copy parmlist into work area 00019600
STORE1 DS      0H          First part of OSREQ modify            00019700
          L      R0,COLLECTION_NAME_PTR Load collection name      00019800
          L      R2,MANAGEMENT_CLASS_PTR Load management class    00019900
          L      R3,MESSAGE_AREA_PTR   Load message area        00020000
          L      R4,OBJECT_NAME_PTR    Load object name          00020100
          L      R5,OBJECT_SIZE_PTR    Load object size          00020200
          L      R6,OFFSET_PTR         Load offset                00020300
          L      R7,REASON_CODE_PTR    Load reason code          00020400
          L      R8,RECALL_NUM_DAYS_PTR Load recall number of days 00020500
          L      R10,RETRIEVE_LENGTH_PTR Load retrieve length     00020600
          OSREQ  (STORE),MF=(M,PARM_LIST),                                X00020700
                  TOKEN=TOKEN_AREA,                                    X00020800
                  COLLECTN=(R0),                                       X00020900
                  MGMTCLAS=(R2),                                       X00021000
                  MSGAREA=(R3),                                       X00021100
                  NAME=(R4),                                           X00021200
                  SIZE=(R5),                                           X00021300
                  OFFSET=(R6),                                         X00021400
                  REACODE=(R7),                                       X00021500
                  RECALL=(R8),                                         X00021600
                  LENGTH=(R10)                                         00021700
STORE2 DS      0H          Second part of OSREQ modify          00021800

```

	L	R0,RETURN_CODE_PTR	Load return code	00021900
	L	R2,RETURN_CODE2_PTR	Load return code2	00022000
	L	R3,STORAGE_CLASS_PTR	Load storage class	00022100
	L	R4,TRACKING_TOKEN_PTR	Load tracking token	00022200
	L	R5,OBJECT_SIZE64_PTR	Load object size 64 bit	00022300
	L	R6,OFFSET64_PTR	Load offset 64 bit	00022400
	L	R7,RETRIEVE_LENGTH64_PTR	Load retrieve length 64 bit	00022500
OSREQ	(STORE),MF=(M,PARAM_LIST),			X00022600
		RETCODE=(R0),		X00022700
		RETCODE2=(R2),		X00022800
		STORCLAS=(R3),		X00022900
		TOKEN=(R4),		X00023000
		SIZE64=(R5),		X00023100
		OFFSET64=(R6),		X00023200
		LENGTH64=(R7)		00023300
RELBUF	DS	0H	Check release buffer	00023400
	CLC	RELEASE_BUFFER,=CL3'YES'	RELBUF requested?	00023500
	BNE	RETPER	If not, default is taken	00023600
OSREQ	(STORE),MF=(M,PARAM_LIST),			X00023700
		RELBUF=YES		00023800
RETPER	DS	0H	Set retention period	00023900
	L	R9,RETENTION_PERIOD_PTR	Load retention period	00024000
OSREQ	(STORE),MF=(M,PARAM_LIST),			X00024100
		RETPD=(R9)		00024200
EVTEXP	DS	0H	Set event expiration	00024300
	L	R9,EVENTEXP_PTR	Load event expiration	00024400
OSREQ	(CHANGE),MF=(M,PARAM_LIST),			X00024500
		EVENTEXP=(R9)		00024600
DELHLD	DS	0H	Check deletion hold	00024700
	CLC	DELHOLD,=CL8'HOLD'	HOLD requested?	00024800
	BE	DELHYES	If so, go to DELHYES	00024900
	CLC	DELHOLD,=CL8'NOHOLD'	Otherwise, NOHOLD requested?	00025000
	BE	DELHNO	If so, go to DELHNO	00025100
	B	FUNCTION	Otherwise, go to FUNCTION	00025200
DELHNO	DS	0H	Set deletion hold "NOHOLD"	00025300
OSREQ	(STORE),MF=(M,PARAM_LIST),			X00025400
		DELHOLD=NOHOLD		00025500
	B	FUNCTION	Go to FUNCTION	00025600
DELHYES	DS	0H	Set deletion hold "HOLD"	00025700
OSREQ	(STORE),MF=(M,PARAM_LIST),			X00025800
		DELHOLD=HOLD		00025900
*****				00026000
* Start of Function select logic. Check FUNCTION_REQUEST for valid				00026100
* function. If valid function is found, then execute. Otherwise, set				00026200
* error and exit.				00026300
*****				00026400
FUNCTION	DS	0H		00026500
	CLC	FUNCTION_REQUEST,=CL8'ACCESS'	Is this an ACCESS?	00026600
	BNE	STORE	If not, then try STORE	00026700
*****				00026800
* ACCESS function				00026900
*				00027000
* The logical connection to OAM is made here. In this sample we use				00027100
* the DSNHLIR entry point passed in to the IADDRESS parameter. DSNHLIR				00027200
* is the entry point responsible for SQL related requests within a				00027300
* stored procedure environment.				00027400
*****				00027500
ACCESS	DS	0H		00027600
	L	R2,DSNHLIR_EP	Load DSNHLIR EP into R2	00027700
OSREQ	ACCESS,MF=(E,PARAM_LIST),			X00027800
		IADDRESS=(R2)		00027900
	B	SAVE_RC	Done, go to save codes	00028000
*****				00028100
* STORE function				00028200
*				00028300
* This will store an object within a supported storage tier depending				00028400
* on the storage class assigned to the object.				00028500
*****				00028600
STORE	DS	0H		00028700
	CLC	FUNCTION_REQUEST,=CL8'STORE'	STORE requested?	00028800
	BNE	CHANGE	If not, then try CHANGE	00028900
	L	R9,BUFFER64_PTR_PTR	Load 64 bit buffer	00029000
	L	R10,STORE_BUFFER_PTR	Load store buffer	00029100
OSREQ	STORE,MF=(E,PARAM_LIST),			X00029200
		BUFFER64=(R9),		X00029300
		BUFLIST=(R10)		00029400
	B	SAVE_RC	Done, go to save codes	00029500
*****				00029600
* CHANGE function				00029700
*				00029800
* This invocation of the OSREQ macro will change information in the				00029900
* directory that has been specified. A zero pointer in DATAAREA				00030000

```

* will result in no change for the respective information. All 00030100
* pointers zero result in no change. 00030200
***** 00030300
CHANGE DS 0H 00030400
        CLC FUNCTION_REQUEST,=CL8'CHANGE' CHANGE requested? 00030500
        BNE QUERY If not, then try QUERY 00030600
        OSREQ CHANGE,MF=(E,PARM_LIST) 00030700
        B SAVE_RC Done, go to save codes 00030800
***** 00030900
* QUERY function 00031000
* 00031100
* Query the data base for the directory information that was stored. 00031200
* The size of the object can be extracted from this information so 00031300
* that a GETMAIN can be done for the area necessary for the 00031400
* retrieve operation. 00031500
***** 00031600
QUERY DS 0H 00031700
        CLC FUNCTION_REQUEST,=CL8'QUERY' QUERY requested? 00031800
        BNE RETRIEVE If not, then try RETRIEVE 00031900
        L R10,QUERY_BUFFER_PTR Load query buffer 00032000
        OSREQ QUERY,MF=(E,PARM_LIST), 00032100
        QEL=(R10) X00032200
        B SAVE_RC Done, go to save codes 00032300
***** 00032400
* RETRIEVE function 00032500
* 00032600
* A partial retrieve can be done to obtain the first xxx bytes of 00032700
* the object. In some cases the application may have some control 00032800
* information in this area to allow retrieval of still another part 00032900
* of the object, (which could be an image). 00033000
***** 00033100
RETRIEVE DS 0H 00033200
        CLC FUNCTION_REQUEST,=CL8'RETRIEVE' RETRIEVE requested? 00033300
        BNE DELETE If not, then try DELETE 00033400
        L R9,BUFFER64_PTR_PTR Load 64 bit buffer 00033500
        L R10,RETRIEVE_BUFFER_PTR Load retrieve buffer 00033600
        OSREQ RETRIEVE,MF=(M,PARM_LIST), 00033700
        VIEW=PRIMARY, X00033800
        BUFFER64=(R9), X00033900
        BUFLIST=(R10) 00034000
VIEW2 DS 0H 00034100
        SLR R6,R6 Check for VIEW=BACKUP 00034200
        L R6,VIEW Zero Register 00034300
        LA R10,2 Load value 2 into R10 00034400
        CR R6,R10 Load value 2 into R10 00034500
        BNE VIEW3 Does view = 2? 00034600
        OSREQ (RETRIEVE),MF=(M,PARM_LIST), No, then check VIEW=BACKUP2 00034700
        VIEW=BACKUP X00034800
        B EXECRET Go to, execute retrieve 00034900
VIEW3 DS 0H 00035000
        LA R10,3 Check for VIEW=BACKUP2 00035100
        CR R6,R10 Load value 3 into R10 00035200
        BNE EXECRET Does view = 3? 00035300
        OSREQ (RETRIEVE),MF=(M,PARM_LIST), No, then use VIEW=PRIMARY 00035400
        VIEW=BACKUP2 X00035500
EXECRET DS 0H 00035600
        OSREQ RETRIEVE,MF=(E,PARM_LIST) Execute retrieve 00035700
        B SAVE_RC Done, go to save codes 00035800
***** 00035900
* DELETE function 00036000
* 00036100
* This invocation will delete the object named from the object table 00036200
* and the directory. 00036300
***** 00036400
DELETE DS 0H 00036500
        CLC FUNCTION_REQUEST,=CL8'DELETE' DELETE requested? 00036600
        BNE UNACCESS If not, then try UNACCESS 00036700
        OSREQ DELETE,MF=(E,PARM_LIST) 00036800
        B SAVE_RC Done, go to save codes 00036900
***** 00037000
* UNACCESS function 00037100
* 00037200
* The logical connection to OAM should be broken before the TASK 00037300
* terminates so that OAM can remove any system control blocks 00037400
* that it built during ACCESS 00037500
***** 00037600
UNACCESS DS 0H 00037700
        CLC FUNCTION_REQUEST,=CL8'UNACCESS' UNACCESS requested? 00037800
        BNE STOREBEG If not, then try STOREBEG 00037900
        OSREQ UNACCESS,MF=(E,PARM_LIST) 00038000
        B SAVE_RC Done, go to save codes 00038100
***** 00038200

```

```

* STOREBEG function                                00038300
*                                                    00038400
* Begin the sequential storage of an object in parts. 00038500
***** 00038600
STOREBEG DS 0H
          CLC FUNCTION_REQUEST,=CL8'STOREBEG' STOREBEG requested? 00038700
          BNE STOREPRT If not, then try STOREPRT 00038800
          ICM R9,15,STIMEOUT_PTR Any STIMEOUT value? 00038900
          BZ EXSTBEG No, then execute STOREBEG 00039000
          OSREQ STOREBEG,MF=(M,PARM_LIST), 00039100
              STIMEOUT=(R9) X00039200
EXSTBEG DS 0H Execute STOREBEG 00039300
          OSREQ STOREBEG,MF=(E,PARM_LIST), X00039400
              STOKEN=STOKEN_AREA 00039500
          B SAVE_RC Done, go to save codes 00039600
***** 00039700
* STOREPRT function                                00039800
*                                                    00039900
* Store the next sequential contiguous part of an object 00040000
***** 00040100
STOREPRT DS 0H
          CLC FUNCTION_REQUEST,=CL8'STOREPRT' STOREPRT requested? 00040200
          BNE STOREEND If not, then try CANCEL 00040300
          L R9,STORE_BUFFER_PTR Load store buffer 00040400
          L R8,BUFFER64_PTR_PTR Load 64 bit buffer 00040500
          OSREQ STOREPRT,MF=(E,PARM_LIST), X00040600
              BUFLIST=(R9), X00040700
              BUFFER64=(R8), X00040800
              STOKEN=STOKEN_AREA X00040900
          B SAVE_RC Done, go to save codes 00041000
***** 00041100
* STOREEND function                                00041200
*                                                    00041300
* End the sequential storage of an object in parts. 00041400
***** 00041500
STOREEND DS 0H
          CLC FUNCTION_REQUEST,=CL8'STOREEND' STOREEND requested? 00041600
          BNE BADFUNC If not, then set invalid rc 00041700
          CLC CANCEL,=CL3'YES' CANCEL requested? 00041800
          BNE NOCANCEL No, then go to NOCANCEL 00041900
          OSREQ (STOREEND),MF=(M,PARM_LIST), X00042000
              CANCEL=YES 00042100
          B SAVE_RC Done, go to save codes 00042200
NOCANCEL DS 0H Execute STOREEND 00042300
          OSREQ STOREEND,MF=(E,PARM_LIST), X00042400
              STOKEN=STOKEN_AREA 00042500
          B SAVE_RC Done, go to save codes 00042600
***** 00042700
* Invalid function requested, set error 00042800
***** 00042900
BADFUNC DS 0H
          LA R15,ERR_FUNC Set function error 00043000
          B EXIT Go to exit 00043100
***** 00043200
* Save register 15 in the return code area and register 0 in the 00043300
* reason code area after return from OSREQ. This is recommended 00043400
* because, under certain error conditions, the return code and 00043500
* reason code areas may not be set by OSREQ. 00043600
***** 00043700
SAVE_RC DS 0H
          L R8,RETURN_CODE_PTR Load OSREQ return code 00043800
          L R9,REASON_CODE_PTR Load OSREQ reason code 00043900
          ST R15,0(,R8) Store return code 00044000
          ST R0,0(,R9) Store reason code 00044100
***** 00044200
* EXIT program 00044300
*                                                    00044400
* Restore all registers except regs 15 and 0, then return to caller 00044500
***** 00044600
EXIT DS 0H
          L R13,SAVE_AREA+4 Reload save area from chain 00044700
          L R14,12(R13) Reload return address 00044800
          LM R1,R12,24(R13) Reload callers registers 00044900
          BR R14 Branch back to caller 00045000
* MAINLINE END 00045100
***** 00045200
* Register definitions 00045300
***** 00045400
R0 EQU 0 00045500
R1 EQU 1 00045600
R2 EQU 2 00045700

```



```

R3      EQU 3      00046500
R4      EQU 4      00046600
R5      EQU 5      00046700
R6      EQU 6      00046800
R7      EQU 7      00046900
R8      EQU 8      00047000
R9      EQU 9      00047100
R10     EQU 10     00047200
R11     EQU 11     00047300
R12     EQU 12     00047400
R13     EQU 13     00047500
R14     EQU 14     00047600
R15     EQU 15     00047700
***** 00047800
* Constants 00047900
***** 00048000
OSRS_ID DC C'OSRS' Header eyecatcher 00048100
OSRS_VER EQU 1 Header version 00048200
ERR_FUNC EQU 6 Invalid Function Request 00048300
ERR_ID EQU 8 Invalid Header ID 00048400
ERR_LEN EQU 10 Invalid Header Length 00048500
ERR_VER EQU 12 Invalid Header Version 00048600
LTORG 00048700
***** 00048800
* This static parameter list will be used as a template for 00048900
* OSREQ invocations in the executable code. 00049000
***** 00049100
STATIC_PARM_LIST OSREQ (STORE),MF=(L) 00049200
STATIC_LIST_END EQU * 00049300
***** 00049400
* 00049500
* The DATAAREA must be obtained by the caller of OSRSP and presented 00049600
* as a parameter (R1) to OSRSP. It is expected that all subsequent 00049700
* calls will point to this same area. There is information in the 00049800
* area that will be used across calls. 00049900
* 00050000
***** 00050100
DATAAREA DSECT 00050200
DA_ID DS CL4 x0 Identifier 00050300
DA_LEN DS F x4 DATAAREA length--x280 (640) 00050400
DA_VER DS X x8 DATAAREA version 00050500
DS CL7 x9 Reserved 00050600
* 00050700
* The following two named fields are set by the caller of CBROSRSRSP. 00050800
* If the value in the field is not a valid value, the respective 00050900
* activity not be executed. 00051000
* 00051100
FUNCTION_REQUEST DS CL8 x10 OSREQ function request value 00051200
* ACCESS, STORE, etc. or other 00051300
DS CL8 x18 Reserved 00051400
* 00051500
* The following field is set by CBROSRSRSP and should not be altered 00051600
* by the caller. Subsequent calls to CBROSRSRSP will rely on this 00051700
* value. 00051800
* 00051900
TOKEN_AREA DS 2F x20 OSREQ token, do not change it. 00052000
STOKEN_AREA DS 4F x28 Area where STOREBEG stores a value 00052100
DS 8F x38 Reserved 00052200
* 00052300
* The following fields are set by the caller of CBROSRSRSP 00052400
* The pointers are not altered by CBROSRSRSP but the data that 00052500
* the pointers reference may be. 00052600
* 00052700
CANCEL DS CL3 x58 Cancels store sequence (YES|NO) 00052800
DS CL1 x5B Reserved 00052900
COLLECTION_NAME_PTR DS A x5C Pointer to collection name 00053000
DS A x60 Reserved 00053100
MANAGEMENT_CLASS_PTR DS A x64 Pointer to management class 00053200
MESSAGE_AREA_PTR DS A x68 Pointer to message area 00053300
OBJECT_NAME_PTR DS A x6C Pointer to object name 00053400
OBJECT_SIZE_PTR DS A x70 Pointer to object size value 00053500
OFFSET_PTR DS A x74 Pointer to offset 00053600
QUERY_BUFFER_PTR DS A x78 Pointer to query buffer list 00053700
REASON_CODE_PTR DS A x7C Pointer to OSREQ reason code 00053800
RECALL_NUM_DAYS_PTR DS A x80 Pointer to recall number of days 00053900
RELEASE_BUFFER DS CL3 x84 Release buffer (YES|NO) 00054000
DS CL1 x87 Reserved 00054100
RETENTION_PERIOD_PTR DS A x88 Pointer to retention period 00054200
RETRIEVE_LENGTH_PTR DS A x8C Pointer to retrieve length value 00054300
RETRIEVE_BUFFER_PTR DS A x90 Pointer to retrieve buffer list 00054400
RETURN_CODE_PTR DS A x94 Pointer to OSREQ return code 00054500
RETURN_CODE2_PTR DS A x98 Pointer to OSREQ return code 2 00054600

```

STIMEOUT_PTR	DS A	x9C	Pointer to STIMETOUT	00054700
STORE_BUFFER_PTR	DS A	xA0	Pointer to store buffer list	00054800
STORAGE_CLASS_PTR	DS A	xA4	Pointer to storage class parameter	00054900
TRACKING_TOKEN_PTR	DS A	xA8	Pointer to tracking token	00055000
VIEW	DS F	xAC	Retrieve Object Copy	00055100
*			1 = PRIMARY	00055200
*			2 = First BACKUP Copy	00055300
*			3 = Second BACKUP Copy	00055400
DELHOLD	DS CL8	xB0	Deletion hold (HOLD NOHOLD blank)	00055500
EVENTEXP_PTR	DS A	xB8	Pointer to EVENTEXP value	00055600
OBJECT_SIZE64_PTR	DS A	xBC	Pointer to 64 bit object size	00055700
OFFSET64_PTR	DS A	xC0	Pointer to 64 bit offset	00055800
RETRIEVE_LENGTH64_PTR	DS A	xC4	Pointer to 64 bit retrieve length	00055900
BUFFER64_PTR_PTR	DS A	xC8	Pointer to 64 bit buffer	00056000
	DS CL12	xCC	Reserved	00056100
*				00056200
* The following field is set by CBROSRSP upon first invocation. This				00056300
* field should not be altered after it is set.				00056400
*				00056500
* Note: This field NEEDS to be initialized to binary zeroes before				00056600
* first invocation.				00056700
*				00056800
DSNHLIR_EP	DS A	xD8	DSNHLIR entry point address	00056900
*				00057000
* Save area				00057100
*				00057200
SAVE_AREA	DS 18F	xDC	Savearea for this module.	00057300
*				00057400
* The following area is completely overlaid each time CBROSRSP				00057500
* is called				00057600
*				00057700
* PARM_LIST DS CL(STATIC_LIST_END-STATIC_PARM_LIST) x124 Dyn. parm list				00057800
*				00057900
* Dyname reserved storage calculation for DATAAREA				00058000
*				00058100
USEDLEN	EQU *-DATAAREA		USEDLEN = x1BC (444)	00058200
RESERVED	DS CL(640-USEDLEN)		RESERVED = xC4 (196)	00058300
DATAAREA_LEN	EQU *-DATAAREA		DATAAREA_LEN = x280 (640)	00058400
CBROSRSP CSECT				00058500
END CBROSRSP				00058600

Appendix B. 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, tape media, or cloud storage, 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, tape media, or cloud storage, 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, tape, or cloud 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 C. 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	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) 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) Store to existing collection only. <ul style="list-style-type: none">• 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.

Table 4. CBRUXSAE return codes (continued)	
Return Code	Description
254	<p>BYPASS CURRENT FUNCTION (IF STORE, RESTRICTED)</p> <p>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.</p> <p>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.</p> <p>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.</p>
255	<p>BYPASS CURRENT FUNCTION (IF STORE, NOT RESTRICTED)</p> <p>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.</p> <p>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.</p> <p>Note: Return codes 254 and 255 have the same meaning for all functions except the store functions (STORE and STORE BEGIN).</p>
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"](#) on page 95 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'      00050000
CBRUXSAE   START 0                      SAMPLE OSREQ TX AUTH INST EXIT    00100000
          SPACE 2                      00150000
**** START OF SPECIFICATIONS ***** 00200000
*                                                * 00250000
*  MODULE NAME:      CBRUXSAE                * 00300000
*                                                * 00350000
```

```

*      DESCRIPTIVE NAME:  SAMPLE OSREQ TRANSACTION SECURITY          * 00400000
*      AUTHORIZATION INSTALLATION EXIT                             * 00450000
*                                                                  * 00456200
*PROPRIETARY V3 STATEMENT                                         * 00462400
*LICENSED MATERIALS - PROPERTY OF IBM                             * 00468600
*"RESTRICTED MATERIALS OF IBM"                                    * 00468601
*5650-ZOS                                                         * 00476800
*COPYRIGHT IBM CORP. 1996, 2009                                   * 00485000
*END PROPRIETARY V3 STATEMENT                                     * 00493400
*                                                                  * 00500000
*      Function:                                                  @L1C* 00512000
*      Module CBRUXSAE is invoked each time a request is made to @L1C* 00524000
*      OAM via the OSREQ interface. CBRUXSAE may refuse to allow @L1C* 00536000
*      the user to perform the requested transaction by returning @L1C* 00548000
*      an appropriate return code in register 15 (described in    @L1C* 00560000
*      the OUTPUT section below).                                @L1A* 00572000
*                                                                  * 00584000
*      Starting with z/OS V1R11, more granular return codes have @L1A* 00596000
*      been implemented to allow bypassing the exit for each of the @L1A* 00608000
*      individual OSREQ functions in addition to the ability to    @L1A* 00620000
*      restrict STOREs to existing collections only.              @L1A* 00632000
*      The additional return codes enable an installation to bypass @L1A* 00644000
*      the exit for any combination of functions. For example, the @L1A* 00656000
*      exit can be bypassed for OSREQ QUERY and RETRIEVE requests @L1A* 00668000
*      but active for OSREQ STORE, and DELETE requests.          @L1A* 00680000
*                                                                  * 00692000
***** !! WARNING !! ***** @L1A* 00704000
*      WARNING: Prior to z/OS V1R11, ANY non-zero return code (except * 00716000
*      RC 16 for BYPASS) meant "authorization failed". Starting with * 00728000
*      z/OS V1R11, return codes 253, 254, and 255 have new meaning * 00740000
*      as described in the OUTPUT section below. If you used 253, * 00752000
*      254, or 255 in a pre-V1R11 version of CBRUXSAE, please review * 00764000
*      the new meanings and modify your exit appropriately.       * 00776000
***** !! WARNING !! ***** @L1A* 00788000
*                                                                  * 00800000
*      THE INSTALLATION CAN PERFORM AUTHORIZATION CHECKING BY ANY * 00850000
*      MEANS IT DEEMS REASONABLE. FOR EXAMPLE:                   * 00900000
*      1. INVOKE RACF VIA THE SAF RACROUTE MACRO                  * 00950000
*      2. USE A TABLE-DRIVEN METHOD OF AUTHORIZATION CHECKING, * 01000000
*      USING A DATASET OF USERIDS AND THE COLLECTIONS/OBJECTS * 01050000
*      A USER IS AUTHORIZED TO PERFORM FUNCTIONS AGAINST.       * 01100000
*      THE AUTHORIZATION CHECKING MAY BE AT THE GRANULARITY THAT * 01150000
*      THE INSTALLATION DECIDES IS NECESSARY, USING THE VALUES * 01200000
*      PASSED IN TO THIS EXIT.                                    * 01250000
*                                                                  * 01300000
*      NOTES:                                                     * 01350000
*      THIS SAMPLE RETURNS WITH A RETURN CODE OF 16, TELLING OAM @03C* 01400000
*      TO CONTINUE PROCESSING.                                     * 01450000
*                                                                  * 01500000
*      DEPENDENCIES:          MVS/SP VERSION 4.3.0                * 01550000
*                              DFSMS/MVS 1.2.0                     * 01600000
*                                                                  * 01650000
*      CHARACTER CODE:        EBCDIC                               * 01700000
*                                                                  * 01750000
*      RESTRICTIONS:          NONE                                  * 01800000
*                                                                  * 01850000
*      REGISTER CONVENTIONS:                                       * 01900000
*      R0 - UNPREDICTABLE                                           * 01950000
*      R1 - STANDARD LINKAGE REGISTER                               * 02000000
*      R2 - UNPREDICTABLE                                           * 02050000
*      R3 - UNPREDICTABLE                                           * 02100000
*      R4 - UNPREDICTABLE                                           * 02150000
*      R5 - UNPREDICTABLE                                           * 02200000
*      R6 - UNPREDICTABLE                                           * 02250000
*      R7 - UNPREDICTABLE                                           * 02300000
*      R8 - UNPREDICTABLE                                           * 02350000
*      R9 - ADDRESS OF AUTODATA AREA FOR EXIT                       * 02400000
*      R10 - UNPREDICTABLE                                           * 02450000
*      R11 - INPUT BASE REGISTER                                     * 02500000
*      R12 - CBRUXSAE BASE REGISTER                                 * 02550000
*      R13 - STANDARD LINKAGE REGISTER                              * 02600000
*      - SAVE AREA ADDRESS                                          * 02650000
*      R14 - STANDARD LINKAGE REGISTER                              * 02700000
*      - RETURN POINT ADDRESS                                       * 02750000
*      R15 - STANDARD LINKAGE REGISTER                              * 02800000
*      - ENTRY POINT ADDRESS                                        * 02850000
*      - RETURN CODE                                                * 02900000
*                                                                  * 02950000
*      MODULE TYPE:          CONTROL SECTION                        * 03000000
*                                                                  * 03050000
*      PROCESSOR:            ASSEMBLER H                           * 03100000
*                                                                  * 03150000

```



```

* ATTRIBUTES: * 03200000
* * 03250000
* LOCATION: LINKLIB * 03300000
* STATE: PROBLEM OR SUPERVISOR (CALLER) * 03350000
* AMODE: 31 * 03400000
* RMODE: ANY * 03450000
* KEY: KEY OF CALLER * 03500000
* MODE: TASK * 03550000
* SERIALIZATION: UNLOCKED * 03600000
* TYPE: REENTRANT, REUSABLE, REFRESHABLE * 03650000
* AUTHORIZATION: NONE * 03700000
* * 03750000
* LINKAGE: STANDARD LINKAGE CONVENTIONS * 03800000
* * 03850000
* CALLING SEQUENCE: * 03900000
* CBRUXSAE IS INVOKED IN THE USER'S ADDRESS SPACE USING THE * 03950000
* MVS LINK MACRO * 04000000
* * 04050000
* INPUT: * 04100000
* REGISTER 1 WILL CONTAIN THE ADDRESS OF A PARAMETER LIST * 04150000
* WHICH WILL CONTAIN 4 POINTERS: * 04200000
* 1. POINTER TO 8 CHARACTER FIELD WHICH CONTAINS THE * 04250000
* OSREQ FUNCTION BEING PERFORMED * 04300000
* POSSIBLE FUNCTIONS ARE: STORE * 04350000
* RETRIEVE * 04400000
* CHANGE * 04450000
* QUERY * 04500000
* DELETE * 04550000
* 2. POINTER TO 44 CHARACTER FIELD WHICH CONTAINS THE * 04600000
* OBJECT NAME ASSOCIATED WITH THE REQUESTED FUNCTION * 04650000
* 3. POINTER TO 44 CHARACTER FIELD WHICH CONTAINS THE * 04700000
* COLLECTION NAME ASSOCIATED WITH THE REQUESTED FUNCTION * 04750000
* 4. POINTER TO 8 CHARACTER FIELD WHICH CONTAINS THE * 04800000
* USERID ASSOCIATED WITH THE REQUESTED FUNCTION * 04850000
* REGISTER 9 WILL CONTAIN THE ADDRESS OF A 1024 BYTE AREA OF * 04900000
* STORAGE WHICH CAN BE USED AS THIS PROGRAM'S AUTOMATIC STORAGE * 04950000
* * 05000000
* OUTPUT: * 05050000
* A RETURN CODE IS PLACED IN REGISTER 15: * 05100000
* * 05107500
* Return * 05115000
* Code Description * 05122500
* -----
* 0 AUTHORIZED * 05130000
* User is authorized to perform this function. The exit will * 05137500
* continue to be called for all normally called OSREQ * 05145000
* functions: * 05152500
* STORE, RETRIEVE, QUERY, CHANGE, DELETE, and STORE BEGIN. * 05160000
* * 05167500
* 16 BYPASSED * 05175000
* The current user and all future users are authorized. Exit * 05182500
* will now be BYPASSED (not called again for any function). * 05190000
* * 05197500
* 224-252 RESERVED (Not Authorized) @L1A * 05205000
* Reserved for IBM. It is recommended that installations do * 05212500
* not use return code values in this range because their * 05220000
* meaning could change in the future. However, they are * 05227500
* currently interpreted as: * 05235000
* User is not authorized to perform this function. No change * 05242500
* is made to the BYPASS status of any OSREQ function. * 05250000
* * 05257500
* 253 STORE RESTRICTED (No Bypass) @L1A * 05265000
* Store to existing collection only. * 05272500
* - For STORE (and STORE BEGIN) function: User is authorized * 05280000
* to store into an existing collection only. Attempts to * 05287500
* store into a collection that does not exist will fail. * 05295000
* - All other OSREQ functions: NOT Authorized. * 05302500
* * 05310000
* This is valid for the current invocation only. No change * 05317500
* is made to the BYPASS status of any OSREQ function. * 05325000
* * 05332500
* 254 BYPASS CURRENT FUNCTION (IF STORE, RESTRICTED) @L1A * 05340000
* Current and future users are authorized to perform the * 05347500
* current function. The exit will be BYPASSED (not called * 05355000
* again) for the current function. If the current function * 05362500
* is a STORE (or STORE BEGIN) then this exit will be bypassed * 05370000
* for subsequent STORE requests. This STORE request and * 05377500
* subsequent STORE requests will be allowed into existing * 05385000
* collections only. Attempts to store into a collection that * 05392500
* does not exist will fail. * 05400000
* Note: If an administrator needs to create a new collection * 05407500
* after this has been set, he'll have to first reset the exit * 05415000
* via the LIBRARY RESET,CBRUXSAE operator command. * 05422500

```

```

*
* 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) @L1*
* 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. @L1*
*
* 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=0W20657 1B0 950501 TUCUJT: Initial release
*
* $01=0W36250 1E0 990104 TUCUJT: Change default to return a @01*
* RC=16 to indicate that the @01*
* exit is not used, therefore @01*
* should not be invoked again @01*
* (Roll up of 0W35784 1C0, 1D0) @01*
* $L1=0AMR1B R11 080523 TUCTMD: OAMR1B CBRUXSAE Enhancement @L1*
* Add new return codes for @L1*
* STORE to existing Collection @L1*
* only, and BYPASS individual @L1*
* OSREQ Functions @L1*
* $03=0A45166 C10 140519 TUCSMN: EXIT SUPPOSED TO DEFAULT WITH @03*
* RETURN CODE OF 16. @03*
*
* **** 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

```

```

* 05430000
* 05437500
* 05445000
* 05452500
* 05460000
* 05467500
* 05475000
* 05482500
* 05490000
* 05497500
* 05505000
* 05512500
* 05520000
* 05527500
* 05535000
* 05542500
* 05550000
* 05557500
* 05565000
* 05572500
* 05580000
* 05587500
* 05595000
* 05602500
* 05610000
* 05617500
* 05625000
* 05632500
* 05640000
* 05650000
* 05700000
* 05750000
* 05800000
* 05850000
* 05900000
* 05950000
* 06000000
* 06050000
* 06100000
* 06150000
* 06200000
* 06250000
* 06300000
* 06350000
* 06400000
* 06450000
* 06500000
* 06550000
* 06600000
* 06650000
* 06700000
* 06725000
* 06750000
* 06756200
* 06762400
* 06768600
* 06774800
* 06781000
* 06787200
* 06788000
* 06788800
* 06789600
* 06790800
* 06792000
* 06792001
* 06792002
* 06793400
* 06800000
* 06850000
* 06900000
* 06950000
* 07000000
* 07050000
* 07100000
* 07150000
* 07200000
* 07250000
* 07300000
* 07350000
* 07400000
* 07450000
* 07500000

```

```

TITLE 'CBRUXSAE WORKING STORAGE'                                07550000
*-----*
*                                                                * 07600000
*                                                                * 07650000
*      MODULE AUTO DATA AREA DEFINITIONS                      * 07700000
*                                                                * 07750000
*-----*
WORKAREA DSECT , CBRUXSAE AUTO DATA AREA                      07800000
SAVEAREA DS 18F SAVE AREA                                     07850000
          DS CL440 AVAILABLE STORAGE                          07900000
WORKLEN EQU *-WORKAREA                                         07950000
          SPACE 2                                              08000000
          TITLE 'STANDARD REGISTER DEFINITIONS'                08050000
*-----*
*                                                                * 08100000
*                                                                * 08150000
*      STANDARD REGISTER DEFINITIONS                          * 08200000
*                                                                * 08250000
*                                                                * 08300000
*-----*
R0 EQU 0 GENERAL REGISTER 0                                     08350000
R1 EQU 1 GENERAL REGISTER 1                                     08400000
R2 EQU 2 GENERAL REGISTER 2                                     08450000
R3 EQU 3 GENERAL REGISTER 3                                     08500000
R4 EQU 4 GENERAL REGISTER 4                                     08550000
R5 EQU 5 GENERAL REGISTER 5                                     08600000
R6 EQU 6 GENERAL REGISTER 6                                     08650000
R7 EQU 7 GENERAL REGISTER 7                                     08700000
R8 EQU 8 GENERAL REGISTER 8                                     08750000
R9 EQU 9 GENERAL REGISTER 9                                     08800000
R10 EQU 10 GENERAL REGISTER 10                                 08850000
R11 EQU 11 GENERAL REGISTER 11                                 08900000
R12 EQU 12 GENERAL REGISTER 12                                 08950000
R13 EQU 13 GENERAL REGISTER 13                                 09000000
R14 EQU 14 GENERAL REGISTER 14                                 09050000
R15 EQU 15 GENERAL REGISTER 15                                 09100000
*-----*
*                                                                * 09150000
*      MISCELLANEOUS CONSTANT VALUES                          * 09154500
*-----*
*                                                                * 09159000
*                                                                * 09163500
UXSAEDIS EQU 16 RC=16 TELLS OSR TO DISABLE @01A 09168000
* FURTHER CALLS TO THIS SECURITY @01A 09172500
* AUTHORIZATION EXIT AND HANDLE @01A 09177000
* SUBSEQUENT INVOCATIONS AS @01A 09181500
* AUTHORIZED USERS @01A 09186000
*-----*
*                                                                * 09190500
*                                                                * 09195000
*      TITLE 'CBRUXSAE - SAMPLE OSREQ TX AUTH INSTALLATION EXIT'
*-----*
*                                                                * 09200000
*                                                                * 09250000
*                                                                * 09300000
*      CBRUXSAE ENTRY POINT                                    * 09350000
*                                                                * 09400000
*-----*
*                                                                * 09450000
CBRUXSAE CSECT , SAMPLE OSREQ TX AUTH INST EXIT                09500000
CBRUXSAE AMODE 31                                              09550000
CBRUXSAE RMODE ANY                                           09600000
          SAVE (14,12),, 'CBRUXSAE&SYSDATE' SAVE CALLER'S REGISTERS AND +09650000
          LR R12,R15 MARK ENTRY POINT                          09700000
          USING CBRUXSAE,R12 COPY ENTRY POINT ADDRESS          09750000
          USING WORKAREA,R9 CBRUXSAE BASE REGISTER            09800000
          ST R13,SAVEAREA+4 ADDRESSABILITY TO DATA AREA      09850000
          LA R0,SAVEAREA BACKWARD CHAIN SAVE AREAS           09900000
          ST R0,8(R13) CBRUXSAE SAVE AREA ADDRESS             09950000
          LR R13,R0 FORWARD CHAIN SAVE AREAS                  10000000
          LR R11,R1 SET CBRUXSAE SAVE AREA ADDRESS            10050000
          USING UXSAEINP,R11 STORE PARAMETERS IN DATA AREA   10100000
          SPACE 2 ADDRESSABILITY TO PARAMETERS                10150000
*-----*
*                                                                * 10200000
*                                                                * 10250000
*                                                                * 10300000
*      RETURN TO THE CALLER                                    * 10350000
*                                                                * 10400000
*-----*
*                                                                * 10450000
EXIT DS 0H 10500000
      L R13,SAVEAREA+4 RESTORE CALLER'S SAVE AREA             10550000
      LA R10,UXSAEDIS SET DISABLE RETURN CODE @01A 10583300
      LR R15,R10 SAVE RETURN CODE @01C 10616600
      RETURN (14,12), RESTORE CALLER'S REGISTERS, THEN +10650000
      RC=(15) RETURN TO CALLER                                10700000
      SPACE 2 10750000
      END CBRUXSAE 10800000

```

Appendix D. Accessibility

Accessible publications for this product are offered through [IBM Documentation for z/OS \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS documentation see [How to Send Feedback to IBM](#) to leave documentation feedback.

Notices

This information was developed for products and services that are offered in the USA 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 grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America*

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.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

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

*IBM Corporation
Site Counsel
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 document 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains 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.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, 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 \(www.ibm.com/software/support/systemsz/lifecycle\)](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](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [Copyright and Trademark information \(www.ibm.com/legal/copytrade.shtml\)](http://www.ibm.com/legal/copytrade.shtml).

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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.

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 file system, optical, tape, and cloud, 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

Numerics

31-bit addressable virtual storage buffers [21](#)
64-bit addressable virtual storage buffers [21](#)

A

ACCESS function
 description [7](#)
 initializing the OSREQ interface [9](#)
 parameter keywords
 DB2ID [9](#)
 IADDRESS [9](#), [11](#), [36](#), [44](#)
 MF [9](#), [37](#)
 MSGAREA [9](#), [38](#)
 REACODE [9](#), [39](#)
 RETCODE [9](#), [40](#)
 TOKEN [9](#), [43](#)
 TTOKEN [9](#), [43](#)
 syntax [9](#)
accessibility
 contact IBM [101](#)
ACS (Automatic Class Selection)
 data class [3](#)
 description [3](#)
 management class names [26](#)
 SMS construct definitions [3](#)
 storage class assignment [13](#)
 storage class name [26](#)
 storage group [3](#)
application
 coordinating OAM's object identification [5](#)
 coordinating with Db2 and OAM [5](#)
assistive technologies [101](#)

B

buffer
 CBRIBUFL macro [49](#)
 data buffer list structure diagram [51](#)
 descriptor [49](#), [51](#)
 keyword parameter [34](#)
 object data [49](#)
 object data reblocking [91](#)
 page release segments [40](#)
 performance considerations [91](#)
 query buffer list structure diagram [54](#)
 RETRIEVE function [51](#)
 temporary storage [91](#)
BUFFER64 keyword parameter
 format [35](#)
BUFLIST [18](#), [23](#)
BUFLIST keyword parameter
 as pointer to CBRIBUFL macro area [49](#)
 format [35](#)
 specifying virtual storage buffers [18](#), [23](#)

C

CBRIBIND SAMPLIB job [46](#)
CBRIBUFL macro
 data buffer list structure diagram [51](#)
 description [49](#)
 DSECTs
 OBL [50](#)
 OBLB [50](#)
 OBLBDESC [50](#)
 used with a RETRIEVE request [51](#)
 used with a STORE request [51](#)
CBRIQEL macro
 description [51](#)
 DSECTs
 QEL [52](#)
 QELB [52](#)
 QELBDESC [52](#)
 QELQ [52](#)
 order retrieval keys [52](#)
 query buffer list structure diagram [54](#)
CBROSREQ [59](#)
CBROSREQ SAMPLIB job [59](#)
CBRUXSAE installation exit
 abend handling [95](#)
 description [93](#)
 programming notes [95](#)
 register contents on entry [94](#)
 sample installation exit [95](#)
CHANGE [11](#)
CHANGE function
 changing an object's management characteristics [11](#)
 date
 processing expiration [47](#)
 updating last referenced [13](#)
 updating pending action [13](#)
 description [7](#)
 parameter keywords
 COLLECTN [35](#)
 MF [37](#)
 MGMTCLAS [38](#)
 MSGAREA [38](#)
 NAME [38](#)
 REACODE [39](#)
 RETCODE [40](#)
 RETPD [41](#)
 STORCLAS [43](#)
 TOKEN [43](#)
 TTOKEN [43](#)
 choosing data types [4](#)
CICS (Customer Information Control System)
 object storage [2](#)
 synchronization with SYNCPOINT [45](#)
 usage requirements [45](#)
 using the OSREQ macro [44](#)
class
 assignments [27](#)

- class (*continued*)
 - data [3](#)
 - defaults [3](#)
 - explicit names [5](#)
 - management [3](#)
 - storage [3](#)
- collection
 - description [1](#), [3](#)
 - error conditions [46](#)
 - naming conventions [5](#)
 - object defaults [3](#), [23](#)
 - processing an object in a collection [27](#)
- COLLECTN [11](#), [14](#), [15](#), [18](#), [23](#)
- COLLECTN keyword parameter
 - collection name length field [18](#)
 - description [35](#)
 - format [35](#)
 - identifying an object for deletion [14](#)
 - querying on an object in a collection [15](#)
 - retrieving an object in a collection [18](#)
- contact
 - z/OS [101](#)

D

- DASD (Direct Access Storage Device)
 - in OAM storage hierarchy [2](#)
 - in object data storage, using [91](#)
- data class
 - ACS routine, updating [3](#)
 - description [3](#)
- data types
 - choosing [4](#)
 - that work well with OAM [4](#)
- databases
 - synchronizing activities [5](#), [11](#), [91](#)
- Db2
 - call attachment facility (CAF) [11](#), [45](#)
 - coordinating with OAM and your application [5](#)
 - deadlocks [47](#)
 - load modules, using JOBLIB and STEPLIB statements in [45](#)
 - message data area [38](#)
 - OSR functions [2](#)
 - timeouts [47](#)
- Db2 SQL
 - error reason codes [49](#)
- DELETE [11](#), [14](#)
- DELETE function
 - deleting an existing object [6](#), [14](#)
 - description [7](#)
 - parameter keywords
 - COLLECTN [35](#)
 - MF [37](#)
 - MSGAREA [38](#)
 - NAME [38](#)
 - REACODE [39](#)
 - RETCODE [40](#)
 - TOKEN [43](#)
 - TOKEN [43](#)
- deleting objects [6](#)
- DSECT
 - CBRIBUFL macro [49](#)
 - CBRIQEL macro [52](#)

E

- error reason codes
 - Db2 SQL [49](#)
 - OAM issuing messages [49](#)
- exit, installation
 - abend handling [95](#)
 - description [93](#)
 - programming notes [95](#)
 - register contents on entry [94](#)
- expiration date processing
 - automatic deletion of objects [47](#)
 - management class retention limit [47](#)
 - object retention period [47](#)
 - reserved value [47](#)
 - valid retention periods [47](#)

F

- file system
 - NFS [54](#)
 - retrieval response time [54](#)
 - zFS [54](#)
- functions
 - OSREQ macro
 - ACCESS [9](#)
- functions used in [11](#), [14](#), [15](#), [18](#), [23](#), [34](#)

I

- IADDRESS keyword parameter
 - application connection to Db2 [46](#)
 - as direct identifier for entry point address [36](#)
 - as optional parameter [44](#)
 - description [36](#)
 - effects in processing environments [11](#)
 - format [36](#)
 - in the ACCESS function [9](#), [44](#)
 - parameter list [36](#)
 - using with structured query language (SQL) [11](#)
- implementing functions
 - with OSREQ macro [9](#)
- in the CHANGE function [11](#)
- in the DELETE function [14](#)
- in the QUERY function [15](#)
- in the RETRIEVE function [18](#)
- in the STORE function [23](#)
- in the UNACCESS function [34](#)

J

- JOBLIB statements
 - assigning concatenation to authorized libraries [45](#)
 - using with Db2 load modules [45](#)

K

- keyboard
 - navigation [101](#)
 - PF keys [101](#)
 - shortcut keys [101](#)
- keyword parameter descriptions [34](#)

L

LENGTH [18](#)
LENGTH keyword parameter
 as optional parameter [37](#)
 description [37](#)
 format [37](#)
 in the RETRIEVE function [18](#)
 omitting the [37](#)
 specifying a portion of an object for retrieval [18](#)
 value of zero [37](#)
LENGTH64 keyword parameter
 format [37](#)
list [23](#), [30](#)

M

macro
 CBRIBUFL [49](#)
 CBRIQEL [51](#)
 OSREQ [7](#)
management class
 assigning to objects [26](#)
 changing [11](#), [13](#)
 defaults [5](#), [23](#)
 description [3](#)
 expiration date processing [47](#)
 format [37](#)
management policy
 overriding defaults [5](#)
messages
 Db2 data area [38](#)
 OSREQ return and reason codes [48](#)
MF [11](#), [14](#), [15](#), [18](#), [23](#)
MF keyword parameter
 as optional input parameter [46](#)
 description [37](#)
 format [8](#), [37](#)
 functions used in
 ACCESS [9](#)
 OSREQ macro forms
 specifying the TOKEN keyword parameter [44](#)
 using the COMPLETE operand [44](#)
 specifying parameters [44](#), [46](#)
MGMTCLAS [11](#), [23](#)
MGMTCLAS keyword parameter
 description [38](#)
 format [11](#), [38](#)
 omitting the [46](#)
MSGAREA [11](#), [14](#), [15](#), [18](#), [23](#)
MSGAREA keyword parameter
 as an optional parameter [9](#)
 description [38](#)
 format [38](#)
 functions used in
 ACCESS [9](#)
 STORE [23](#)

N

NAME [11](#), [14](#), [15](#), [18](#), [23](#)
NAME keyword parameter
 description [14](#), [15](#), [18](#), [23](#)

NAME keyword parameter (*continued*)
 format [38](#)
 object name length field as input for the [18](#)
navigation
 keyboard [101](#)

O

OAM
 coordinating with application and Db2 [5](#)
object
 access time [5](#)
 administration [3](#)
 changing an object's management characteristics [11](#)
 data reblocking [91](#)
 deleting an existing object [14](#)
 deleting directory information [7](#)
 descriptive information [5](#)
 establishing the storage management policy [2](#)
 expiration date processing [47](#)
 name field [56](#)
 name, qualifying the [9](#)
 processing large objects [91](#)
 querying the directory [7](#)
 retrieval response time [56](#)
 retrieving objects [18](#)
 separating [6](#)
 size restrictions and limitations [45](#)
 storage device basis [27](#)
 storing directory information [7](#)
 temporary storage [91](#)
Object Access Method (OAM)
 application program interface [7](#)
 choosing data types [3](#)
 description [1](#)
 establishing the storage management policy [2](#)
 naming conventions [3](#)
 reason codes [48](#)
 return codes [48](#)
 SMS construct definitions [3](#)
 understanding [1](#)
 understanding the components
 Library Control System (LCS) [2](#)
 OAM Storage Management Component (OSMC) [2](#)
 Object Storage and Retrieval (OSR) [2](#)
object retrieval [92](#)
object storage hierarchy
 adding objects to [21](#)
objects
 adding to object storage hierarchy [21](#)
 deleting [6](#)
 retrieval of [92](#)
 retrieving a partial object [4](#)
 storage of [91](#)
OFFSET [18](#)
OFFSET keyword parameter
 description [39](#)
 format [39](#)
 in the RETRIEVE function [39](#)
 omitting the [39](#)
 retrieving an object [18](#), [37](#)
 retrieving part of an object [37](#)
OFFSET64 keyword parameter
 format [39](#)

optical

object retrieval [91](#)

volumes

reading and writing [2](#)

OSREQ macro

CBRIBUFL macro [34](#), [49](#)

CBRIQEL macro [51](#)

CBROSREQ SAMPLIB job [59](#)

choosing form [8](#)

coding guidelines [8](#)

criteria for OSREQ macro use [3](#)

description [1](#), [7](#)

ending the OSREQ interface [34](#)

functions of the macro [7](#)

how to read syntax diagrams [x](#)

implementing functions with [9](#)

initializing the macro [9](#)

optional input parameter [46](#)

OSREQ keyword parameter descriptions [34](#)

OSREQ return and reason codes [48](#)

register values at invocation [47](#)

sample program using [59](#)

supported functions [9](#)

under CICS [44](#)

usage recommendations [44](#)

usage requirements [45](#)

using the OSREQ macro [7](#)

P

parameter

input/output requirements
[45](#)

keywords

BUFFER64 [35](#)

BUFLIST [9](#), [34](#), [35](#)

COLLECTN [9](#), [34](#), [35](#)

IADDRESS [9](#), [36](#)

LENGTH [9](#), [37](#)

LENGTH64 [37](#)

MF [9](#), [37](#)

MGMTCLAS [9](#), [37](#), [38](#)

MSGAREA [9](#), [38](#)

NAME [9](#), [38](#)

OFFSET [9](#), [39](#)

OFFSET64 [39](#)

QEL [9](#), [39](#)

REACODE [9](#), [39](#)

RELBUF [9](#), [40](#)

RETCODE [9](#), [40](#)

RETPD [9](#), [40](#), [41](#), [47](#)

SIZE [9](#), [42](#)

SIZE64 [42](#)

STORCLAS [9](#), [43](#)

TOKEN [9](#), [43](#)

TOKEN [9](#), [43](#)

VIEW [9](#), [43](#)

OSREQ conventions [44](#)

parameter keywords [11](#), [14](#), [15](#), [18](#), [23](#)

product knowledge

required [ix](#)

programming interface information [106](#)

Q

QEL [15](#)

QEL (query element list) keyword parameter

as pointer to CBRIQEL macro [51](#)

as query buffer length field (QELBBLTH) [55](#)

as retrieval order key fields

backup retrieval order key (QELQBROK) [52](#), [53](#)

primary retrieval order key (QELQPROK) [52](#), [53](#)

secondary backup retrieval order key (QELQB2OK)
[52](#), [53](#)

as retrieval response time field (QELQERRT) [54](#)

buffer space [51](#), [52](#), [54](#), [55](#)

description [15](#), [39](#), [51](#)

DSECT description [52](#)

format [39](#)

in the CBRIQEL macro [51](#), [53](#), [54](#)

in the QUERY function [15](#)

QUERY [15](#)

QUERY function

CBRIQEL macro [51](#)

description [7](#)

generic search [15](#)

getting object characteristics [15](#)

name conventions [38](#)

parameter keywords

COLLECTN [35](#)

MF [37](#)

MSGAREA [38](#)

NAME [38](#)

QEL [39](#)

REACODE [39](#)

RETCODE [40](#)

TOKEN [43](#)

TOKEN [43](#)

QEL keyword parameter [51](#)

query buffer

mapping [51](#)

QELUSED field parameter [55](#)

retrieving an existing object [18](#)

R

REACODE [11](#), [14](#), [15](#), [18](#), [23](#)

REACODE keyword parameter

as an optional parameter [9](#), [39](#)

description [39](#)

format [39](#)

functions used in

ACCESS [9](#)

STORE [23](#)

reason codes

OSREQ macro [48](#)

REACODE keyword parameter

in the ACCESS function [9](#)

in the STORE function [23](#)

RECALL keyword parameter

functions used in [39](#)

recovery, object

successful [18](#)

use of the RETRIEVE function in [18](#)

RELBUF [23](#)

RELBUF keyword parameter

default value [40](#)

RELBUF keyword parameter (*continued*)

description [40, 91](#)

format [40](#)

required product knowledge [ix](#)

RETCODE [11, 14, 15, 18, 23](#)

RETCODE keyword parameter
as an optional parameter [9](#)

description [40](#)

format [40](#)

functions used in

ACCESS [9](#)

STORE [23](#)

RETCODE2 keyword parameter

functions used in [40](#)

retention period

changing for previously stored objects [11](#)

expiration attributes [23, 47](#)

expiration date processing [13, 47](#)

management class assignment [13](#)

null parameter value [13, 46](#)

overriding [40](#)

specifying on a STORE function [23](#)

specifying override retention period [41, 47](#)

RETPD [11, 23](#)

RETPD keyword parameter

description [41](#)

format [41](#)

range for parameter values [13, 47](#)

RETRIEVE [18](#)

RETRIEVE function

backup retrieval [18](#)

buffer use [51](#)

date

updating last referenced [18](#)

updating pending action [18](#)

description [7, 18](#)

parameter keywords

BUFFER64 [35](#)

BUFLIST [35](#)

COLLECTN [18, 35](#)

LENGTH [37](#)

LENGTH64 [37](#)

MF [37](#)

MSGAREA [38](#)

NAME [38](#)

OFFSET [39](#)

OFFSET64 [39](#)

REACODE [39](#)

RETCODE [40](#)

SIZE64 [42](#)

TOKEN [43](#)

TOKEN [43](#)

VIEW [18, 43](#)

QUERY output using the [18](#)

QUERY request as input [17, 18](#)

retrieval response time [56](#)

single object recovery and the [18](#)

return codes [48](#)

S

sample installation exits

CBRUXSAE [95](#)

sample program

sample program (*continued*)

CBROSREQ [59](#)

for object storage request [59](#)

using OSREQ macro [59](#)

SAMPLIB job

CBRIBIND [46](#)

CBROSR2 [67](#)

CBROSR3 [75](#)

CBROSREQ

generating the IADDRESS keyword parameter [57](#)

ways to use [57](#)

CBROSRSP [83](#)

CBRUXSAE [93](#)

security authorization checking [93](#)

shortcut keys [101](#)

size

keyword [42](#)

processing large objects, limitations on [91](#)

restrictions and limitations, object [45](#)

SIZE [23](#)

SIZE keyword parameter

description [42](#)

format [42](#)

specifying number of bytes [23, 43](#)

SIZE64 keyword parameter

format [42](#)

SSTOREEND [32](#)

STEPLIB statements

assigning concatenation to authorized libraries [45](#)

using with Db2 load modules [45](#)

storage

of objects [91](#)

storage class

assigning to objects [26](#)

changing for an object [11](#)

defaults [5, 23](#)

description [3](#)

storage group

affiliating libraries with a [2](#)

assigning backup storage groups using SETOSMC

statements [3](#)

assigning collections to a [6](#)

description [3](#)

OAM storage hierarchy [2, 3](#)

storage management

class, changing [13](#)

constructs [3](#)

establishing the storage management policy [2](#)

storage management policy

establishing [2](#)

STORCLAS [11, 23](#)

STORCLAS keyword parameter

description [43](#)

format [43](#)

null parameter value [46](#)

omitting the [43](#)

STORE [23](#)

STORE function

collection name [23](#)

description [7, 23](#)

expiration date processing [47](#)

parameter keywords

BUFFER64 [35](#)

BUFLIST [35](#)

STORE function (*continued*)

parameter keywords (*continued*)

- COLLECTN [35](#)
- LENGTH64 [37](#)
- MF [37](#)
- MGMTCLAS [38](#)
- NAME [38](#)
- OFFSET64 [39](#)
- REACODE [39](#)
- RELBUF [40](#)
- RETCODE [40](#)
- RETPD [41](#)
- SIZE [42](#)
- SIZE64 [42](#)
- STORCLAS [43](#)
- TOKEN [43](#)
- TOKEN [43](#)

performance considerations [91](#)

syntax [23](#)

Store Sequence functions

- canceling [35](#)
- overview [21](#)
- RETCODE2 value [41](#)
- SIZE value [42](#)
- SSTOREEND [32](#)
- STOREBEG [27](#)
- STOREPRT [30](#)
- timeout interval [43](#)

STOREBEG [27](#)

STOREPRT [30](#)

structured query language (SQL)

COMMIT and [11](#)

CONNECT and [11](#)

interface module entry point address [11](#)

using with the IADDRESS function [11](#)

summary of changes [xv](#)

syntax [11](#), [14](#), [15](#), [18](#), [34](#)

syntax diagrams

- ACCESS [9](#)
- how to read [x](#)
- STORE [23](#)

T

TOKEN [11](#), [14](#), [15](#), [18](#), [23](#)

TOKEN keyword parameter

- causes abend, invalid [44](#)
- clearing TOKEN contents [11](#), [43](#)
- description [43](#)
- format [43](#)
- functions used in
 - ACCESS [9](#)
- passing to subroutines [44](#)
- setting the [9](#)

trademarks [106](#)

TOKEN [11](#), [14](#), [15](#), [18](#), [23](#)

TOKEN keyword parameter

- description [43](#)
- format [43](#)
- functions used in
 - ACCESS [9](#)
 - STORE [23](#)

U

UNACCESS [34](#)

UNACCESS function

- clearing TOKEN contents [11](#), [43](#)
- description [7](#)
- ending the OSREQ interface [34](#)
- parameter keywords
 - MF [37](#)
 - MSGAREA [38](#)
 - REACODE [39](#)
 - RETCODE [40](#)
 - TOKEN [43](#)

user interface

- ISPF [101](#)
- TSO/E [101](#)

V

VIEW [18](#)

VIEW keyword parameter

- default value [43](#)
- description [43](#)
- format [43](#)
- no second backup object exists when issuing the [18](#)
- valid values [18](#)



Product Number: 5655-ZOS

SC23-6865-70

