z/VM
7.4

*Reusable Server Kernel
Programmer's Guide and Reference*

IBM

Note:

Before you use this information and the product it supports, read the information in "Notices" on page 463.

Contents

Figures

Tables

About This Document

This document describes how you can use the IBM z/VM reusable server kernel to develop and execute server programs on the z/VM Conversational Monitor System (CMS).

Intended Audience

This document is for programmers who want to develop server programs and run them in the CMS environment.

This document covers advanced material in server construction and is not for beginning programmers. To use the material in this document, readers should:

- Know one of the supported programming languages, and
- Understand concurrent programming concepts, including both general techniques and specific concepts relevant to CMS Application Multitasking, and
- Have experience with CMS application development and the tools and facilities used by CMS application developers (for example, the GENMOD command and the Callable Services Library), and
- Have a working knowledge of CMS and z/VM as they appear to the CMS application developer, and
- Have application development experience with at least one z/VM connectivity technology, such as TCP/IP.

Where to Find More Information

For more information, see at the back of this document.

How to provide feedback to IBM

We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information. See How to send feedback to IBM for additional information.

Summary of Changes for z/VM: Reusable Server Kernel Programmer's Guide and Reference

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.

SC24-6313-74, z/VM 7.4 (September 2024)

This edition supports the general availability of z/VM 7.4. Note that the publication number suffix (-74) indicates the z/VM release to which this edition applies.

SC24-6313-73, z/VM 7.3 (September 2023)

This edition includes terminology, maintenance, and editorial changes.

SC24-6313-73, z/VM 7.3 (September 2022)

This edition supports the general availability of z/VM 7.3. Note that the publication number suffix (-73) indicates the z/VM release to which this edition applies.

SC24-6313-01, z/VM 7.2 (September 2020)

This edition supports the general availability of z/VM 7.2.

Chapter 1. Basic Concepts

Motivation

Most operating systems suitable as server platforms offer a variety of technologies to the server author. For example, such operating systems might offer one or more sets of communication interfaces, a threading interface, a file system interface, an enrollment and authorization interface, storage management primitives, and so on. In some cases, the technologies offered the server author are complex, advanced technologies for which the deployment strategies, programming interfaces, and even the problems solved are apparent only after much study.

The problem created by such systems is that they foist the technology assimilation, assessment, deployment, and integration responsibilities onto the server author. To use the system's technologies in a smart way, the server developer must learn all the system's technology elements, understand their APIs, understand the problems each element is designed to solve, and understand how these apparently-discrete technology elements relate to one another. This creates a large burden for the server developer, and it creates a situation in which each server author (at different companies, for example) must endure the same learning curve in order to construct a server that exploits the technology of the operating system underneath it. Alternatively, such systems create the problem that server authors do not exploit the systems' technologies because they do not understand the technologies or how to apply them; this creates a problem for the server applications being developed -- they do not use the system optimally.

To overcome these problems on z/VM, IBM studied the problem of z/VM server construction and identified problems common to many servers. Further, it identified the technologies relevant to solving those problems in an optimal way and is delivering server enablers employing these technologies. IBM's first efforts in this area produced *Server Tasking Environment/VM* and its follow-on, *CMS Application Multitasking*; these very significant CMS enhancements moved CMS from a single-processor, single-threaded programming environment to a parallel, multithreaded system. Continued work in this area has produced not more operating system code but rather has produced an "empty" server program that server writers can use as a starting point for server construction. This "empty" server, called the *reusable server kernel*, consists of a text library of routines and a macro library of function prototypes and constant definitions. To construct an actual server program, the server author attaches application-specific code to a set of interfaces in the reusable server kernel. The result of such attachment is a server program heavily exploitive of the z/VM system's best technologies.

A specific example of the reusable server kernel's ability to relieve the server author of technology exploitation will be helpful. It is well known that building a z/VM server in a multithreaded fashion helps boost the server's performance and makes the server easier to design and understand. A server author desiring to write such a program on his own would need to understand how to use CMS Application Multitasking to construct a multithreaded program, and he would also need to decide upon a strategy for dividing the server into multiple threads of execution. The reusable server kernel, though, lets the server author *ignore* how to use CMS's tasking primitives to implement such a structure; instead, the reusable server kernel itself organizes the server into this form, maintaining its own structures and strategies for doing so. The only work left for the server author is to identify (through a server kernel-provided programming interface) one or more "get request, do request, answer client" loops, or "services". The server kernel replicates these services on multiple threads, doing so in response to the workload moving through the server. In other words, it is the server kernel that makes the author's code multithreaded, not the author.

The reusable server kernel provides help in more than just multithreading. Additional help is provided in these areas:

Topic	Description	Page
Table 1. Additional Help Areas		
Topic	**Description**	**Page**
Connectivity	A big part of server design and development is the selection and deployment of connectivity strategies for the server program. The reusable server kernel includes line drivers for both bulk-data and operator-oriented protocols and unifies all of these line drivers under a single interface. The server writer develops no communication code when he uses the reusable server kernel.	Chapter 2, "Connectivity and Line Drivers," on page 11
DASD I/O	The reusable server kernel organizes the server's DASD volumes into one or more _storage groups_. This set of storage groups can be brought online, brought offline, changed in size, and so on through a set of APIs or a set of commands. I/O to these storage groups is thread-synchronous, thread-blocking, and does not serialize on the base virtual processor. When the server runs in an XC-mode virtual machine, the reusable server kernel can be configured to use CP's MAPMDISK facility to perform I/O to its storage groups. Using MAPMDISK lets the server program feel the benefits of caching and the I/O efficiencies of the paging subsystem. In other virtual machine types, or if using MAPMDISK is inappropriate for some other reason, the reusable server kernel can use DIAGNOSE X'0250' or DIAGNOSE X'00A4' for storage group I/O.	Chapter 3, "DASD Management," on page 25
File Caching	Many servers, such as HTTP daemons, are read-intensive with respect to CMS's file systems (minidisk, Shared File System, and Byte File System). The reusable server kernel offers a file caching API that lets the server cache such files in a VM Data Space. The caching support offers an open-read-close model for file reading; when the server opens a file through this API, the reusable server kernel loads the file into a VM Data Space and keeps it there for reuse until it becomes stale or is forced out because of storage contention. The server can instruct the server kernel to perform code page translation or record delineation scheme transformations on the file as part of loading it into the cache. This lets the cached file be kept in the data space in the form most useful to clients.	Chapter 4, "File Caching," on page 31
Authorization	The reusable server kernel provides callable entry points for managing the authorization of users to objects. These entry points implement a class-oriented paradigm wherein the objects, classes, and access types for each class are completely defined by the server writer. The authorization data can reside on CMS minidisks or in either accessed or unaccessed Shared File System directories.	Chapter 5, "Authorization," on page 35

Table 1. Additional Help Areas (continued)		
Topic	**Description**	**Page**
Enrollment	Most servers maintain some kind of user database. In the abstract, these databases are usually nothing more than indexed access methods. The reusable server kernel offers an API containing insert, delete, and lookup operations for records having fixed-length, 64-byte keys and up to 65,450 bytes of data. The reusable server kernel holds the records in a VM Data Space, hashing them for quick lookup, and backs the VM Data Space with a file in the Shared File System. The hashing scheme makes it possible to hold many hundreds of thousands of records with very good performance.	Chapter 6, "Enrollment," on page 43
Indexing by Prefixes	The reusable server kernel provides APIs that allow the server application to build and interrogate indices by prefix. The reusable server kernel keeps each index in its own VM Data Space while allowing multiple RSK-based service machines concurrent access.	Chapter 7, "Indexing by Prefixes," on page 47
Anchors	Callable entry points let the server program set and query the value of a server-wide anchor word.	Chapter 8, "Anchors," on page 49
Memory Management	The reusable server kernel provides callable storage allocation and release primitives designed for multithreaded servers and suitable for most situations. In addition, these APIs can allocate and release storage in a VM Data Space.	Chapter 9, "Memory Management," on page 51
Run-time Environment	The reusable server kernel provides an automatic storage management convention that improves the performance of the server by minimizing the number of storage management calls needed to manage automatic storage (that is, execution stack storage). This convention prevents storage management calls in most cases.	Chapter 11, "Run-Time Environment," on page 59
Worker Machines	The reusable server kernel provides a facility that lets the server author run server work in a pool of virtual machines, instead of all in a single machine. The server kernel takes care of autologging these worker machines and moving data between the central server and the workers. This is useful for offloading complex functions or for isolating risky or time-consuming operations.	Chapter 10, "Worker Machines," on page 53
Configuration and Operation	The reusable server kernel's operation is configurable and controllable through a set of commands. These commands let the operator start and stop services, manipulate storage groups, and perform other tasks related to server management. This set of commands can be used by an exec through ADDRESS RSK as part of an initialization strategy or can be submitted through several of the reusable server kernel's line drivers.	Chapter 12, "Initialization and Profiles," on page 63

Table 1. Additional Help Areas (continued)		
Topic	**Description**	**Page**
Socket Library	The RSK socket library is a PL/X application programming interface for socket programming. Although the library does not provide a one-for-one correspondent for every IUCV socket function, it does provide many of the basic operations needed to communicate with other socket programs.	Chapter 16, "RSK Sockets," on page 327

Overall Server Organization

Fundamentally, a *server program* is a program that accepts requests from *clients* and generates responses for those clients. Some servers are very transaction-oriented; they accept a single, entire request from a client, produce an entire response for the client, and then wait for another request from the client. Other servers are much more stream-oriented; in these situations, the server and client carry on a running dialogue over which they exchange information freely with one another, perhaps not according to any strict request/response paradigm. The server author's choice of interaction paradigm is based usually on the kind of work being performed and the kind of communication technology being used. Personal preference no doubt also plays a role in this choice.

Whether the relationship is transaction-oriented or stream-oriented, the primary job of the server is to handle requests from clients. Though handling of such system facilities as communications, virtual storage, disk, and I/O devices is part of the overall picture in the server, the essential job of the server is to *interact with the client*. All of the logic in the server supports this fundamental operation. Even interaction with the server operator is a form of interacting with a client, though at first glance it might seem that interacting with the operator is fundamentally different from interacting with "regular" clients.

The reusable server kernel strongly emphasizes this fundamental property by organizing the server writer's work precisely along these lines. The server writer's primary responsibility is to provide one or more routines, called *services*, whose job is to interact with a client over an abstract channel. The server writer also provides a server mainline, the responsibility of which is to bring up the server, wait for it to finish, and then take it down. Figure 1 on page 5 illustrates this organization.
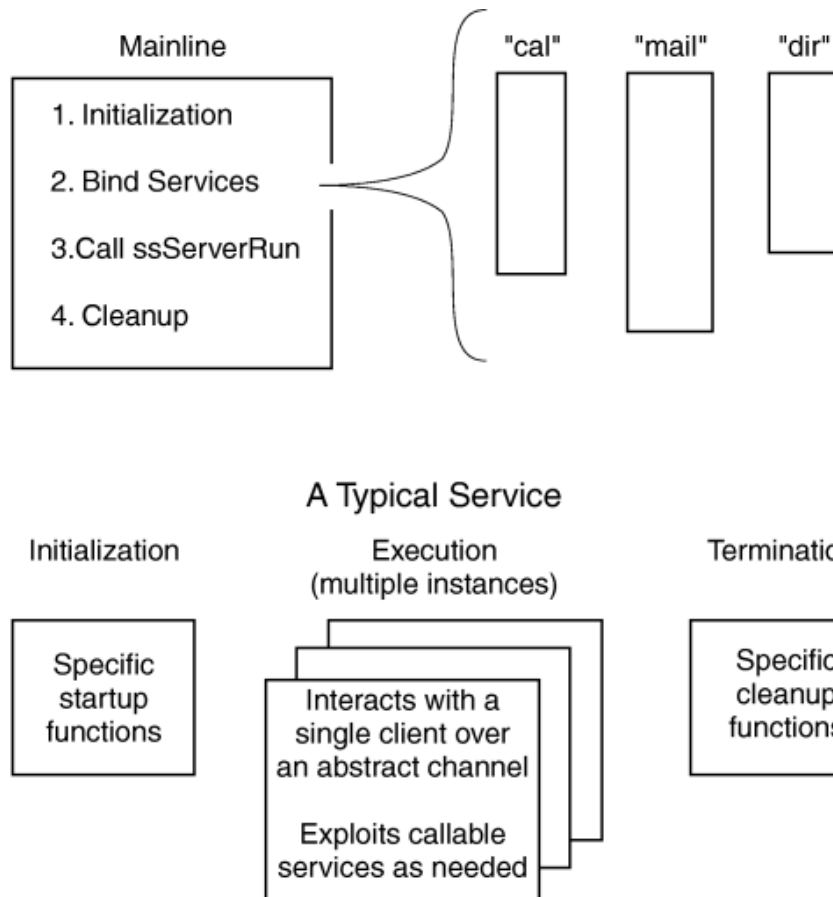
Figure 1. Reusable Server Kernel Overview

Jobs of the Mainline

The server mainline gets control shortly after the server module is invoked. It has a few essential responsibilities:

1. It may perform server-wide initialization, such as reading and processing a configuration file, checking and adjusting the virtual machine configuration, or starting a console log.

2. It must identify, or *bind*, one or more services. Binding a service makes it known to the reusable server kernel and thereby makes it eligible to be "started" through operator command.

3. It must call entry point `ssServerRun` to run the server program. Control returns to the mainline when the server has ended.

4. It may perform server-wide termination processing, such as closing a console log.

5. It must return to its caller.

More About Services

Service identification takes place during server initialization, in the mainline provided by the server author. The reusable server kernel provides a callable interface, `ssServiceBind`, which lets the server writer identify the set of services available. The server writer should arrange the mainline so that it calls `ssServiceBind` once for each service being offered. Once a service is bound, it is available for use for the life of the server.

`ssServiceBind` accepts as parameters a case-insensitive, eight-byte name for the service and certain descriptive information about the service. In response to the call, it builds a data structure called the *service block* or *S-block*, which is illustrated in Table 2 on page 6.

Table 2. Service Block, or S-Block		
Offset	**Length**	**Usage**
0	8	Used by IBM
8	8	Service name
16	4	Service name length
20	4	Address of initialization routine
24	4	Address of service routine
28	4	Address of termination routine
32	4	Service type
36	4	Service lockword
40	4	Current start count
44	4	Monitor data row pointer

Perhaps the most important parameters to `ssServiceBind` are the addresses of these key entry points:

- **Initialization entry point:** a reusable server kernel line driver calls a service's *initialization* entry point when it starts the service but before it lets the service do any work for clients, but only if the service is completely idle -- that is, only if the service is not currently handling clients through any other line driver.

  The initialization entry point should be prepared to accept a parameter list organized according to Table 3 on page 6. The return code and reason code in this parameter list are output parameters to be filled in by the initialization entry point. If the initialization entry point produces a nonzero return code, the start attempt will fail.

Table 3. Initialization Entry Point Parameter List. R1 points to this data structure on entry.		
Offset	**Length**	**Usage**
0	4	A(return code)
4	4	A(reason code)
8	4	A(S-block)

- **Service entry point:** a reusable server kernel line driver activates a service's *service* entry point in response to work accruing from clients. When a new client arrives, the line driver dedicates a thread -- an *instance* of the service -- to the new client and causes that thread to call the service entry point. [1] A given client is always served by the same instance, and a given instance serves exactly one client.

  The line drivers provided by the reusable server kernel are *parallelizing,* that is, they attempt to run a service's service entry point on more than one thread concurrently if necessary. Configuration parameter SRV_THREADS controls the maximum number of threads on which a given line driver will attempt to run a given service's service entry point. For more information, see Table 31 on page 66.

  The service entry point should be prepared to accept a parameter list organized according to Table 4 on page 7. By way of this parameter list, the reusable server kernel passes the service entry point the address of a crucial data structure called the *client block* or *C-block*. The C-block, which represents the partnership among the client, the line driver, and the instance, contains information the instance uses to

---

[1] Do not confuse *starting an instance* with a call to CMS's `ThreadCreate` function. The reusable server kernel keeps a pool of threads on which it runs service instances. Each such thread resides in its own dispatch class. Depending upon workload, there may be more than once instance of a given service executing at any given moment. In other words, the reusable server kernel parallelizes the server according to the workload moving through the server.

interact with the reusable server kernel and also contains fields identifying and characterizing the client. For more information on the C-block, see "From Line Driver to Instance" on page 15.

Table 4. Service Entry Point Parameter List. R1 points to this data structure on entry.		
Offset	**Length**	**Usage**
0	4	A(S-block)
4	4	A(C-block)

The relationship between the line driver and the instance is carried out through the CSL's queuing primitives, using a queue owned by the line driver, called the *line driver queue*. Information necessary to use this queue is contained in the C-block. To send messages to one another, the line driver and the instance use QueueSend to place messages on the queue. To receive messages from each other, the line driver and the instance use one of the "receive" primitives, such as QueueReceiveBlock, once again operating on the line driver queue. The selective-receipt facility of the CSL's queue routines is used so that the line driver and the set of instances using the line driver queue can all use the queue without interfering with one another. [2] Specific information about the exchange of messages between line drivers and services is available in Chapter 2, "Connectivity and Line Drivers," on page 11.

When handling of the client is complete, the service entry point should return to its caller.

- **Termination entry point:** a line driver drives a service's *termination* entry point as part of "stop" processing, if the service is not currently started through any other line drivers.

  The parameter list for the termination entry point is described in Table 5 on page 7.

Table 5. Termination Entry Point Parameter List. R1 points to this data structure on entry.		
Offset	**Length**	**Usage**
0	4	A(S-block)

Note: For information on the rest of the S-block fields, see "Writing Your Own Line Driver" on page 23.

Anything Else?

Beyond this, the organization of the server program is up to the server author. The usual approach will be to implement a mainline and one or more services, along perhaps with some service threads that perform encapsulated operations on shared data or some other repetitive work. The server author is strongly encouraged to use CMS Application Multitasking functions for communication among threads, implementation of critical sections, and performing other server-related operations.

Calling The Entry Points

Calls to the reusable server kernel's entry points are coded as ordinary assembler or PL/X function calls. Language bindings for each of these languages are provided in macro libraries — DMSGPI for assembler and DMSRP for PL/X.

DMSGPI Macros

The names of the macros are:

---

[2] Each IPC key generated by the reusable server kernel, whether for external or internal use, has BKW (X'C2D2E6') as its first three characters. This permits author-supplied code to exploit line driver queues for other purposes when it seems helpful.

Macro	Description	Page
SSASMANC	Anchor bindings	"Anchor Bindings (SSASMANC MACRO)" on page 415
SSASMAUT	Authorization bindings	"Authorization Bindings (SSASMAUT MACRO)" on page 416
SSASMCAC	File cache bindings	"Cache Bindings (SSASMCAC MACRO)" on page 421
SSASMCLI	Client bindings	"Client Bindings (SSASMCLI MACRO)" on page 423
SSASMENR	Enrollment bindings	"Enrollment Bindings (SSASMENR MACRO)" on page 425
SSASMMEM	Memory bindings	"Memory Bindings (SSASMMEM MACRO)" on page 428
SSASMSGP	Storage group bindings	"Storage Group Bindings (SSASMSGP MACRO)" on page 430
SSASMSRV	Service and server bindings	"Services Bindings (SSASMSRV MACRO)" on page 433
SSASMTRI	Trie API bindings	"Trie Bindings (SSASMTRI MACRO)" on page 436
SSASMUID	User ID bindings	"User ID Bindings (SSASMUID MACRO)" on page 438
SSASMWRK	Worker machine bindings	"Worker Bindings (SSASMWRK MACRO)" on page 439

DMSRP Macros

The names of the macros are:

Macro	Description	Page
SSPLXANC	Anchor bindings	"Anchor Bindings (SSPLXANC COPY)" on page 440
SSPLXAUT	Authorization bindings	"Authorization Bindings (SSPLXAUT COPY)" on page 441
SSPLXCAC	File cache bindings	"Cache Bindings (SSPLXCAC COPY)" on page 444
SSPLXCLI	Client bindings	"Client Bindings (SSPLXCLI COPY)" on page 447
SSPLXENR	Enrollment bindings	"Enrollment Bindings (SSPLXENR COPY)" on page 448
SSPLXMEM	Memory bindings	"Memory Bindings (SSPLXMEM COPY)" on page 450
SSPLXSGP	Storage group bindings	"Storage Group Bindings (SSPLXSGP COPY)" on page 451

Macro	Description	Page
SSPLXSRV	Service and server bindings	"Services Bindings (SSPLXSRV COPY)" on page 454
SSPLXTRI	Trie API bindings	"Trie Bindings (SSPLXTRI COPY)" on page 456
SSPLXUID	User ID bindings	"User ID Bindings (SSPLXUID COPY)" on page 457
SSPLXWRK	Worker machine bindings	"Worker Bindings (SSPLXWRK COPY)" on page 458

These macros are invoked with the same conventions as the CMS Application Multitasking macros, namely:

- for Assembler, just invoke the macro through its name.
- for PL/X, use `%include syslib(`*`macro`*`);`.

Of course, you must make these macro libraries available to your compiler or assembler by using the `GLOBAL MACLIB` command.

A single standard for procedure linkage is used throughout the server. This standard affords each procedure, whether customer-written or IBM-supplied, an extremely fast method for obtaining and releasing automatic storage (that is, storage for local variables and save areas). All of the reusable server kernel entry points expect the server author to use this linkage to call them, and the reusable server kernel drives all customer-written routines (thread entry points, server entry point, and so on) using this linkage. Macros are provided to implement the procedure linkage. For more information, see Chapter 11, "Run-Time Environment," on page 59.

Building a Server Module

To create a server using the reusable server kernel, the server author writes a set of application-specific code, calling the reusable server kernel entry points as desired. Using an appropriate language processor, the server author prepares one or more object modules (files of file type TEXT) containing his application. Exactly one of these object modules defines entry point RSKMAIN, which is the server's entry point. [3]

To build his module, the server writer link-edits his object code with the reusable server kernel object library and any other object libraries needed. The result of the link-edit is a module containing both the author's application and the appropriate reusable server kernel code. For example, if the server were implemented in a single object deck called SAMPLE, this sequence of CMS commands would accomplish the link-edit:

```
GLOBAL TXTLIB BKWLIB DMSPSLK DMSAMT VMMTLIB VMLIB CMSSAA
LOAD SAMPLE ( CLEAR DUP   AUTO   LIBE NOINV FULLMAP RLDSAVE
INCLUDE VMSTART ( NOCLEAR DUP   AUTO   LIBE NOINV FULLMAP RLDSAVE RESET VMSTART
GENMOD SAMPLE ( MAP STR
```

The effect of these commands is to produce SAMPLE MODULE, the resultant server, and SAMPLE LOADMAP, the load map associated with the module.

Note:

1. If there were multiple customer-supplied object modules, they would be accounted for in this procedure by inserting the appropriate INCLUDE commands after the LOAD of the server mainline.

---

[3] This is very much like the APPLMAIN required by a CMS Application Multitasking program. In fact, the reusable server kernel *is* a CMS Application Multitasking program and provides its own APPLMAIN. RSKMAIN is the label of the first instruction of the actual server code written by the server author.

2. It is important to note that BKWLIB appears ahead of DMSAMT in the text library search order. BKWLIB contains a DMSLESB (language environment selector text deck) that overrides the one found in DMSAMT.

Setup At A Glance

In addition to the module you build, you will need these additional files to run your server:

Table 6. Files Needed to Run Your Server

File	Description
BKWRTE MODULE	This is the run-time environment manager program for the server. Place this file somewhere in the server's file mode search order.
BKWUME TEXT	This is the reusable server kernel's message repository. Make sure your server's virtual machine issues SET LANGUAGE (ADD BKW USER as part of its PROFILE EXEC.
PROFILE RSK	The reusable server kernel runs this exec just after your server module begins execution; the PROFILE RSK you write contains the configuration and startup commands you need for your specific environment.
User ID Mapping File	Controls the reusable server kernel's translation from connectivity-specific client identifiers to a normalized, flat client name space.

If you plan to use certain other features of the reusable server kernel, you will need to perform additional setup operations, according to:

Table 7. Additional Setup Tasks

Feature	Task	Page
Storage groups	You will need to provide a *storage group configuration file*.	Chapter 3, "DASD Management," on page 25
Authorization API	You will need to set up *authorization data*.	Chapter 5, "Authorization," on page 35
Enrollment API	You will need to set up *enrollment files*.	Chapter 6, "Enrollment," on page 43
Worker API	You will need to set up *worker machines*.	Chapter 10, "Worker Machines," on page 53

Other Considerations

The reusable server kernel manages the server as a CMS Application Multitasking program. All the information contained in the publication *z/VM: CMS Application Multitasking* applies to programs written using the reusable server kernel. For more information, see *z/VM: CMS Application Multitasking*.

Chapter 2. Connectivity and Line Drivers

Server authors usually desire that their servers support many connectivity methods, for this increases the variety and number of clients that can be served. For example, a database server might desire to use TCP/IP and spool files as connectivity methods for clients; this would let clients reside on a variety of networks and platforms. Similarly, a server author might desire that the server program accept operator commands and deliver operator responses over a number of channels (CP MSG, CP SMSG, virtual console I/O); this would let the server program be operable remotely or locally, with no extra work being done by the server author.

A major problem in supporting heterogeneous connectivity is that the server author must learn a set of communication interfaces for each connectivity technology to be supported, and he must write exploiting code for each connectivity API. Further, the higher levels of such exploiting code are usually similar, regardless of the transport technology being exploited; for example, most connection-oriented transports support *initialize*, *send*, *receive*, and *terminate* primitives, and the server's treatment of those primitives is remarkably similar from one transport to the next. Thus an additional problem, duplication of effort, is also apparent.

The reusable server kernel relieves the server author of the burden of supporting multiple connectivity technologies. It furnishes the server writer with a set of line drivers and does so in a way that hides most communication differences from the server writer. Each line driver performs these basic functions for the server core:

- It creates and deletes service instances in response to the arrival and departure of clients.
- It collects bytes from clients and delivers them to service instances according to the mapping between service instances and clients and in the order in which said bytes arrive.
- It acts as the transmission agent for the set of service instances, sending bytes to clients in the order in which the respective clients' service instances emit them.
- It ascertains the identities of clients, mapping them into a single user id space, and informs service instances of said identities.

Each of these functions is performed in a way consistent with the APIs and capabilities of the respective connectivity technologies.

The reusable server kernel provides a set of line drivers, one driver for each transport protocol it supports:

- APPC/VM (global, local, and private resource managers)
- IUCV
- TCP/IP
- UDP/IP
- Spool files
- MSG/SMSG
- Virtual console
- Subcom

Each driver is organized according to .
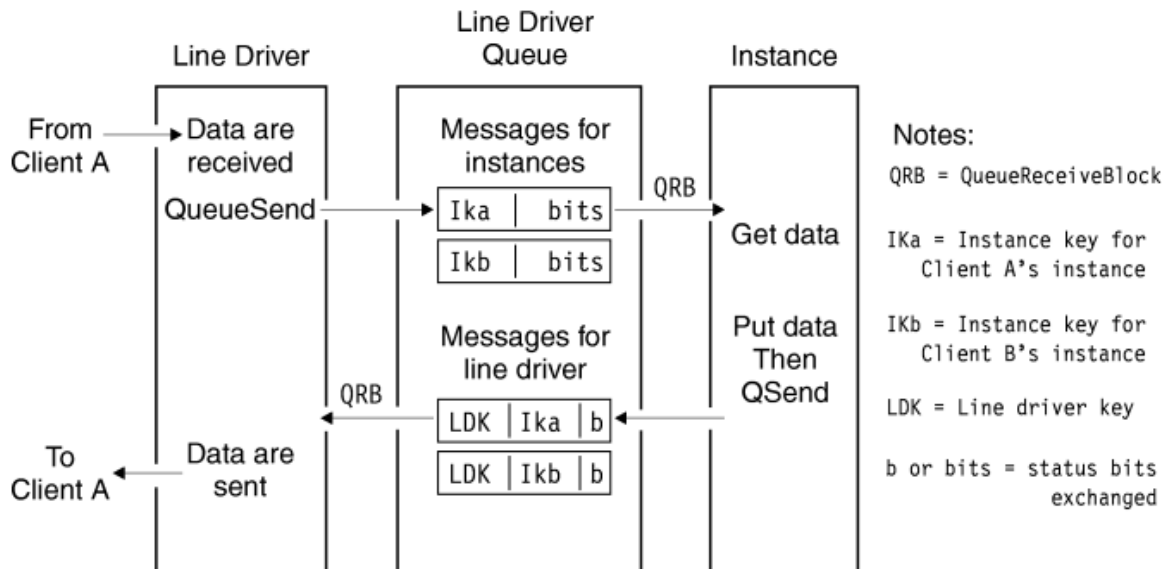
Figure 2. Line Driver Organization

The Service Instance's View

As introduced earlier, a service instance interacts with a line driver through two mechanisms:

- When a line driver starts an instance, it passes the instance a control block that describes the partnership among the client, the line driver, and the instance. This control block is called the *client block* or *C-block*.

- To interact with one another, the line driver and the instance exchange messages using a CMS queue maintained by the line driver. This queue is called the *line driver queue*. They also enqueue and dequeue data on a set of reusable server kernel-maintained client buffers. These buffers are accessed with the `ssClientDataGet` and `ssClientDataPut` primitives.

This section describes the C-block and the messages exchanged through the line driver queue. [4]

The Client Block, or C-Block

As mentioned in "More About Services" on page 5, the relationship between a line driver and an instance of a service is carried out through a control block -- the *C-block* -- and a CMS queue. Some of the most important information in the C-block, then, is information describing the queue to be used and how it is to be used. This information appears in the C-block in the form of queue handles and message keys. Table 8 on page 12 summarizes the fields of the C-block.

Table 8. Client Block, or C-Block			
Offset	**Length**	**Usage**	**Description**
0	4	S-block pointer	The address of the S-block for the service with which this instance is affiliated.
4	8	Line driver name	The name of the line driver with which the service is interacting. The names are given in Table 9 on page 13.

---

[4] For the server writer's convenience, macros SSPLXSRV COPY and SSASMSRV MACRO contain mappings of the C-block and the messages exchanged by way of the line driver queue.

Table 8. Client Block, or C-Block (continued)			
Offset	**Length**	**Usage**	**Description**

12	4	Line driver status word	Specific information about the line driver. The bits of the status word have these meanings:
			
			Bit
			    Meaning
			X'80000000'
			    The line driver is *record-oriented*:
			
			    • When supplying the instance with client input, the line driver organizes the client's input as a sequence of records. Each record is prefixed with a four-byte length field. The value stored in the four-byte length field does *not* include the length of the length field itself.
			    • When producing output for the client, the instance must organize the output as a sequence of records, as described previously.
			
			    The MSG/SMSG, CONSOLE, SUBCOM, and SPOOL drivers are record-oriented.

16	4	Line driver queue handle	The queue handle the instance should use to receive messages from and send messages to its associated line driver.
20	4	Line driver service ID	The service ID of the line driver queue. This might be useful to the instance in some situations.
24	4	Instance identifier	An integer identifier assigned to this instance by the line driver. This numeric identifier will never be reused by this line driver.
28	4	Instance thread ID	The CMS thread ID of the thread on which the instance is running.
32	32	Instance key	The key the line driver will use when it transmits messages needing the instance's attention. Such messages will be placed on the line driver queue, are indicative of client activity, and are organized according to Table 12 on page 15. The instance key is the key the instance should use in its *receive* (for example, QueueReceiveBlock) call.
64	32	Line driver key	The key the instance should use when it transmits messages needing the line driver's attention. Such messages should be placed on the line driver queue, are usually indicative of the instance's having queued data for transmission to the client, and are organized according to Table 13 on page 16.
96	64	Mapped user ID of client	The reusable server kernel's best attempt at assessing the user ID of the client. Depending on the communication transport being used, this assessment is made in several different ways, as shown in Table 10 on page 14.
160	4	Total bytes into instance	The total number of bytes the instance's client has sent the instance so far.
164	4	Total bytes from instance	The total number of bytes the instance has sent to the client so far.
168	4	Bytes waiting for instance	The number of bytes waiting to be consumed by the instance.
172	4	Bytes waiting for line driver	The number of bytes waiting to be consumed by the line driver.
176	8	Start STCK	The time at which the client began communicating with the server, stored according to the format of the Store Clock (STCK) instruction.
184	8	Reserved for IBM	
192	128	Reserved for IBM	
320	Varies	Line-driver-specific data	The data is organized differently for each line driver, as shown in Table 11 on page 14.

Table 9. Line Driver Names. All names are padded on the right with spaces (X'40').	
Line Driver	**Name in C-Block**
APPC/VM	APPC

Line Driver	Name in C-Block
IUCV	`IUCV`
TCP/IP	`TCP`
UDP/IP	`UDP`
SPOOL	`SPOOL`
MSG/SMSG	`MSG`
Console	`CONSOLE`
Subcom	`SUBCOM`

Table 10. User ID Mapping Schemes

Transport	Method
APPC/VM	Security user ID of conversation, mapped through user ID mapping file
IUCV	Field `IPVMID` of connection pending EIB, mapped through user ID mapping file
MSG	User ID and node of origin of message, mapped through user ID mapping file
TCP/IP	IP address of client, mapped through user ID mapping file
UDP/IP	IP address of client, mapped through user ID mapping file
Spool	User ID and node of origin of spool file, mapped through user ID mapping file
Console	Literal ⋆
Subcom	Literal ⋆

Table 11. Line-Driver-Specific Portion of C-Block

Line Driver	Data

TCP/IP	**0.4**
	      IP address of client
	4.4
	      Port number of client
	8.4
	      Port where TCP line driver is listening

UDP/IP	**0.4**
	      IP address of client
	4.4
	      Port number of client

APPC/VM	**0.8**
	      Security user ID of client
	8.17
	      Locally known LU of client

IUCV	**0.8** Field `IPVMID` from connection pending EIB

Table 11. Line-Driver-Specific Portion of C-Block (continued)	
Line Driver	**Data**

Spool	**0.8**
	    Reserved for IBM
	
	8.8
	    User ID of client
	
	16.8
	    Node of client
	
	24.4
	    Spool ID of reader file (character form)

MSG/SMSG	**0.4**
	    Reserved for IBM
	
	4.8
	    User ID of client
	
	12.8
	    Node of client

Console	None present
Subcom	None present

From Line Driver to Instance

A reusable server kernel line driver transmits a message to the instance each time something "interesting" happens with respect to the client. This message serves to notify the instance that something has happened and to advise the instance that it might wish to take a corresponding action. The message contains status bits that indicate exactly how the relationship with the client has changed. This message is organized according to Table 12 on page 15. The instance can pick up these notifications using `QueueReceiveBlock`, [5] using the line driver queue handle and instance key from the C-block.

Each message to an instance will have its *message type* field set to `ss_srv_msgtype_instance`. [6] Usually the instance's reaction to such a notification will be to attempt to retrieve data from the client and process it. To do so, the instance should use `ssClientDataGet`.

When the instance sees a message in which the *line driver STOP* bit is set, it should:

- Emit any remaining transmissions intended for its current client
- Transmit a *STOP acknowledgement* message to the line driver
- Return to its caller.

For more information, see "From Instance to Line Driver" on page 16.

Table 12. Message from Line Driver to Instance. The reusable server kernel always transmits this message using key offset 0 and key length 32.		
Offset	**Length**	**Usage**
0	32	Instance's key
32	4	Message type

---

[5] `QueueReceiveImmed` is also acceptable.
[6] Defined in SSPLXSRV COPY and SSASMSRV MACRO.

Offset	Length	Usage

36	2	Client status bits
		
		X'8000'
		    Client has closed connection
		
		X'4000'
		    Connection closed abnormally
		
		X'2000'
		    Client has finished sending
		
		X'1000'
		    Line driver requests STOP
		
		X'0800'
		    New data from client

Table 12. Message from Line Driver to Instance. The reusable server kernel always transmits this message using key offset 0 and key length 32. *(continued)*

From Instance to Line Driver

To send data to the client, the instance should use routine `ssClientDataPut` and then notify its line driver of the new data by using QueueSend. The precise form of the message the instance should transmit is given in .

The instance should set the *message type* field to `ss_srv_msgtype_linedriver` in each message it transmits to the line driver.

To inform the line driver that it has queued additional information for the client, the instance should set the *instance has queued output* bit in the message it transmits to the line driver.

To acknowledge a stop request from the line driver, or to indicate that it is spontaneously stopping for its own reasons, the instance should set the *stop acknowledgement* bit in the message it transmits to the line driver.

Table 13. Message from Instance to Line Driver. The instance always transmits this message using key offset 0 and key length 32.

Offset	Length	Usage
0	32	Line driver's key
32	4	Message type
36	32	Instance's key

68	2	Instance status bits
		
		X'8000'
		    Stop acknowledgement
		
		X'4000'
		    Instance has queued output

TCP/IP Considerations

To use TCP/IP, the server machine must be configured for TCP/IP operation. Typically this means that the server must be enabled to use IUCV to communicate with the TCP/IP service machine. These CP directory considerations apply:

- The server machine must be permitted to connect to the TCP/IP service machine. Typically the TCP/IP service machine has `IUCV  ALLOW` in its own CP directory entry; when this is the case, no special work is required in the server machine's directory entry.

- The server machine's MAXCONN must be set high enough to let TCP/IP activity proceed. The reusable server kernel's TCP/IP line driver consumes one IUCV path ID per started service.

These other considerations apply:

- When the TCP/IP line driver starts a service, it binds the service's port number onto the adapter address specified in the START command and issues `listen()` with a backlog queue size of 10.
- Clients should connect to the reusable server kernel using stream sockets.
- The reusable server kernel creates all its sockets in addressing family AF_INET.
- The TCP/IP line driver uses the reusable server kernel's user ID mapping facility with connectivity identifier TCP to map the client's IP address into a single-token user ID. [7] Because IP addresses can be spoofed, this feature should be exploited only if the IP network is trusted.
- If the reusable server kernel is not able to map the user ID, then it behaves according to the setting of configuration parameter NOMAP_TCP:

    OFF
    Connection is closed

    ON
    User ID $UNKNOWN is passed to instance

UDP/IP Considerations

Like using TCP/IP, using UDP/IP requires that the server machine be configured for TCP/IP operation. Again, this means that the server must be enabled to use IUCV to communicate with the TCP/IP service machine. To achieve this, follow the same procedures as you would use to set up for TCP/IP operation. Be aware that the UDP/IP line driver consumes one IUCV path per started service, just as the TCP/IP line driver does; plan your MAXCONN accordingly.

The following other considerations apply:

- When the UDP/IP line driver starts a service, it binds the service's port number onto the adapter address specified in the START command.
- Clients should send to the server using datagram sockets and should expect the server's response to come as one or more datagrams.
- The reusable server kernel considers each received datagram to be representative of a distinct transaction. When a datagram arrives, the reusable server kernel creates a service instance and passes the datagram's contents to the service instance through `ssClientDataPut`. In other words, a service instance will only ever "see" **one** inbound datagram from a client. Each inbound datagram is considered to be its own transaction and accordingly is delivered to a separate instance.
- For a given service instance, the reusable server kernel will emit as many response datagrams to the client as are necessary, until the service indicates completion of the transaction through usual means (*stop acknowledgement* bit set in IPC message to line driver).
- The UDP/IP line driver uses the reusable server kernel's user ID mapping facility with connectivity identifier UDP to map the client's IP address into a single-token user ID. [8] Because IP addresses can be spoofed, this feature should be exploited only if the IP network is trusted.
- If the reusable server kernel is not able to map the user ID, then it behaves according to the setting of configuration parameter NOMAP_UDP:

    OFF
    Datagram is ignored

    ON
    User ID $UNKNOWN is passed to instance

---

[7] In the call to `ssUseridMap`, parameter *nodename* is filled with the IP address and parameter *userid* is filled with *.
[8] In the call to `ssUseridMap`, parameter *nodename* is filled with the IP address and parameter *userid* is filled with *.

IUCV Considerations

To use IUCV, the server virtual machine must be configured for IUCV operation. Typically this means the following for the server's CP directory entry:

- `IUCV ALLOW` should be specified so that clients can connect to the server virtual machine.
- `OPTION MAXCONN` must be set large enough to handle the number of clients you anticipate will be connected to the server concurrently. Allow one connection for each client.

For more information, see *z/VM: Connectivity*.

The following specific considerations apply to the use of IUCV. These considerations will be particularly helpful in writing clients.

- The server kernel uses CMS's `CMSIUCV` and `HNDIUCV` macros for IUCV path management, so as not to interfere with other IUCV or APPC/VM usage in the server virtual machine.
- The reusable server kernel opens an `HNDIUCV` exit for each service it starts. Usually, the name of the exit matches the name of the service. The server operator can override this with the `IUCV START` command if some other exit name must be used.
- A client wishing to connect to an reusable server kernel-managed service must specify the name of the service's exit routine in the `IPUSER` field of its `IUCV CONNECT` parameter list.
- The server kernel issues `IUCV ACCEPT` with `MSGLIM` set to 65535. The server administrator can force a lower value by installing an appropriate IUCV control statement in the server's CP directory entry.
- The reusable server kernel produces the client's mapped user ID by calling `ssUseridMap` with connectivity identifier IUCV, specifying the **local nodename** and the VM user ID of the client (field `IPVMID` of the connection pending EIB) as the remaining inputs.
- If the reusable server kernel is not able to map the user ID, then it behaves according to the setting of configuration parameter `NOMAP_IUCV`:

   OFF
   > Path is severed

   ON
   > The `IPVMID` field of the connection pending EIB is passed to the instance
- The reusable server kernel lets the client use `IUCV SEND` with either DATA=PRMMSG or DATA=BUFFER. However, the reusable server kernel always transmits using DATA=BUFFER.
- The reusable server kernel does not permit the client to use `IUCV SEND,TYPE=2WAY`. All sends to the server must be one-way sends. If the client attempts a two-way send, the reusable server kernel will sever the path.
- The server kernel will tolerate IUCV priority messages but never sends them.
- Data arriving from the client is queued to the affiliated service instance in the order that the message pending interrupts arrive, without regard to any other factors.
- The server kernel is optimized for 64 KB transfers between the client and the server. In fact, the reusable server kernel never transmits more than 64 KB in a single IUCV message. Best results will be achieved when the client takes this optimization into account.
- The reusable server kernel does not permit the client to use `IUCV QUIESCE` or `IUCV RESUME`. It will sever the path if the client tries these. Similarly, the reusable server kernel never uses these macros itself.

APPC/VM Considerations

To use APPC/VM, the server virtual machine must be configured for APPC/VM operation. Typically this involves one or more of these:

- Adding proper IUCV-related statements to the virtual machine's directory entry. These statements control the names of the resources the machine is allowed to identify and the number of concurrent

conversations the machine is allowed to use. Sometimes permitting clients to connect is also accomplished here.

- If the virtual machine is managing an APPC/VM private resource,

    - The virtual machine must IPL CMS with parameter AUTOCR.
    - The virtual machine should run with Fullscreen CMS off.
    - File `PROFILE EXEC` should contain `SET SERVER ON`.
    - File `$SERVER$ NAMES` must be set up to map the resource name to the name of the server program and to identify the clients permitted to connect.

For more information, see *z/VM: Connectivity*.

The following specific considerations apply to the use of APPC/VM. These considerations will be particularly helpful in writing clients.

- To allocate a conversation to the server, the client should use the LU name appropriate for the server virtual machine's location and resource type and a TPN equal to the one used in the server's APPC START command. For more information, see Chapter 14, "Command Descriptions," on page 77.

- The APPC/VM line driver accepts either mapped or basic conversations. Be aware, though, that inbound APPC record boundaries are not visible to the instance and that the instance has no control over record boundaries in outbound APPC records.

- The APPC/VM line driver uses the connectivity identifier APPC, the LU of the client, and the user ID of the client as input to its user ID mapping function. For more information on user ID mapping, see Chapter 12, "Initialization and Profiles," on page 63. The client's node is taken to be his LU (field `CPEVPLKL` of the connection pending extended data) and his user ID is taken to be field `IPVMID` of the connection pending EIB. If the conversation was allocated with SECURITY(NONE), the server kernel substitutes `$UNKNOWN` for the X'0000000000000000' user ID CP supplies in the EIB.

- If the reusable server kernel is not able to map the user ID, then it behaves according to the setting of configuration parameter `NOMAP_APPC`, as follows:

    OFF
    Conversation is severed

    ON
    The `IPVMID` field of the connection pending EIB (or `$UNKNOWN`, if SECURITY(NONE)) is passed to the instance.

- The reusable server kernel does not support SYNCLVL(CONFIRM) or SYNCLVL(SYNCPT) conversations. Attempts to use these will result in a sever.

Spool Considerations

These considerations apply when using spool files as a connectivity mechanism:

- Requests from clients arrive at the server virtual machine's reader from either the same node as the server or from remote nodes through RSCS or functional equivalent.

- Spool files containing requests must be encoded using one of the following techniques:

    - NETDATA encoding (`NEW` option of SENDFILE)
    - DISK DUMP encoding (`OLD` option of SENDFILE)

If a file encoded with some other technique arrives, the reusable server kernel will CP TRANSFER it to the user ID specified by the `SPL_CATCHER` configuration parameter, or if no such user ID is specified, the file will remain in the server's reader in USER HOLD status.

No matter which encoding is used, each data record of the sent file is extracted and given to the service as a record of input. (The spool driver is record-oriented.)

- The reusable server kernel considers only those reader files having filetype matching the value of configuration parameter `SPL_INPUT_FT`. All other reader files are ignored.

- When a spool file arrives, the reusable server kernel scans the reader for new work. When it finds a file whose filetype matches configuration parameter SPL_INPUT_FT, and whose filename matches a started service, and which is not in a hold of some kind, the driver reads the file's data from the spool and attempts to deliver the data to the started service.

- When SPOOL START is issued, the reusable server kernel scans the reader for new work, just as it would scan as a result of spool file arrival, but with the following addition: if a file would have been delivered to the newly-started service *except for the fact that the file has been found to be in USER HOLD state*, the file is changed to NOHOLD and its data is delivered to the newly-started service.

- If the file name of the spool file does not match the name of any started service, and if implicit VM routing is enabled for the spool driver, then the reusable server kernel delivers the file's data records to the CMS service, provided the CMS service has been started. For more information about implicit routing, see Chapter 12, "Initialization and Profiles," on page 63.

- While processing of a file is underway, the file remains in the reader in USER HOLD state.

- If delivery of the file's data to its service fails, or if the service fails to consume all of the data of the spool file, the file is left in the reader in USER HOLD state. Otherwise the file is purged.

- The spool driver uses the reusable server kernel's user ID mapping facility with connectivity identifier SPOOL to map the origin user ID and origin node of the spool file into a single-token user ID. For more information on the user ID mapping facility, see Chapter 12, "Initialization and Profiles," on page 63. This user ID is passed to the service instance as the client's user ID. However, if the spool driver's call to the user ID mapping facility reveals that no mapping exists, action is taken, if NOMAP_SPOOL is:

  - OFF, the file is placed in USER HOLD status and a message is issued to the server console.
  - ON, the file is passed to the service instance, with the origin user ID passed directly as the "mapped" user ID.

- The SPOOL line driver parallelizes requests. If a client sends multiple requests to the same service, the two requests might finish in an order other than the one in which they were sent. This applies also to the situation where the multiple requests are sent to different services.

MSG/SMSG Considerations

The CP MSG and CP SMSG commands can be used to send work to service instances being managed by the reusable server kernel. The following considerations apply:

- Each MSG or SMSG should bear as its first token the prefix supplied on the MSG START command that started the service. For example, to send a request called SHUTDOWN to the service started with prefix CAL_OPER running in virtual machine SERVER, an operator might issue this command:

```
TELL SERVER CAL_OPER SHUTDOWN
```

- If the first token of the message (in the above example, CAL_OPER) does not match the name of any request processor registered in the server, and if implicit VM routing is enabled for the MSG/SMSG line driver, then the reusable server kernel delivers the command to the CMS service, provided the CMS service has been started.

  For more information about implicit routing, see Chapter 12, "Initialization and Profiles," on page 63.

- Each message the MSG/SMSG line driver places in a line driver queue contains a single MSG or SMSG sent to the server virtual machine.

- The MSG/SMSG line driver uses the user ID mapping facility with connectivity identifiers MSG and SMSG to map the user ID and node of the message sender to a single-token user ID. This user ID is the one passed to the request processor in the C-block header. However, if the driver's call to the user ID mapping facility reveals that no mapping exists, action is taken as follows:

  - If NOMAP_MSG is OFF, the message is ignored and an error message is written to the server console.
  - If NOMAP_MSG is ON, the message is sent to the service instance, with the origin user ID passed directly as the "mapped" user ID.

- The MSG/SMSG line driver is record-oriented.

- The MSG/SMSG line driver parallelizes requests. If a client sends multiple requests to the same service, the two requests might finish in an order other than the one in which they were sent. This applies also to the situation where the multiple requests are sent to different services.
- When the MSG/SMSG driver builds output, it prefixes each line of service output with the prefix assigned to the service, padded to 8 characters. For example, for service CAL_OPER above, each line of output produced by the CAL_OPER service would be prefixed with CAL_OPER.

Virtual Console Considerations

The reusable server kernel runs the server virtual machine's console in line mode. These considerations apply:

- When entering a command for a service, the operator should use the prefix supplied on the CONSOLE START command as the first token of the command line. For example, to send a request called SHUTDOWN to the service called CAL_OPER, the operator should enter the following on the virtual machine's console:

```
CAL_OPER SHUTDOWN
```

- If the first token of the command (in the above example, CAL_OPER) does not match the name of any request processor registered in the server, and if implicit VM routing is enabled for the console line driver, then the reusable server kernel delivers the command to the CMS service, provided the CMS service has been started. For more information about implicit routing, see Chapter 12, "Initialization and Profiles," on page 63. The console driver:
  - Always supplies * as the mapped client user ID.
  - Is record-oriented.
  - Parallelizes the services it starts. Requests sent to a given service are begun in the order in which they are typed, but they might complete in a different order.
- When the console driver routes output to the console, it prefixes each line of service output with the prefix assigned to the service, padded to 8 characters. For example, for service CAL_OPER above, each line of output produced by the CAL_OPER service would be prefixed with CAL_OPER. For this reason, if it is possible in your environment, the server virtual machine's console should be wider than 80 columns. IBM recommends that you use at least 90 columns for the console.

Subcom Considerations

The reusable server kernel supplies a subcom, RSK, to which execs may direct commands; the output of such commands is written to the virtual console. These considerations apply:

- When issuing a command to a service, the exec writer should use the prefix supplied on the SUBCOM START command as the first token of the command. For example, to issue a command called SHUTDOWN to the service called CAL_OPER, the exec writer might code:

```
address 'RSK' 'CAL_OPER SHUTDOWN'
```

- If the first token of the command (in the above example, CAL_OPER) does not match the name of any request processor registered in the server, and if implicit VM routing is enabled for the SUBCOM line driver, then the reusable server kernel delivers the command to the CMS service, provided the CMS service has been started. For more information about implicit routing, see Chapter 12, "Initialization and Profiles," on page 63.
- The SUBCOM driver always supplies * as the mapped client user ID.
- The SUBCOM line driver is record-oriented.
- The SUBCOM driver does not return to the calling EXEC until the command is complete.
- The SUBCOM driver routes service output to the virtual console, in the manner of the console line driver.
- Because services do not generate return codes, the server author should not use Rexx variable rc as an indication of the completion status of commands issued through the SUBCOM driver.

Line Driver Commands

As mentioned earlier, services are started and stopped by line drivers. This is done through line driver *commands*. Largely speaking, line driver commands are present to perform these important functions:

- *Starting* a service is nothing more than connecting it to a reusable server kernel line driver -- the *start* operation is an instruction to a line driver to prepare for communication and connect its communication device or channel to a named service. In other words, an operator starts a service by issuing a command that's interpreted by a specific line driver; in response to the command, the line driver begins driving work through the service.

- *Stopping* a service is nothing more than informing a line driver that its communication method should be shut down; as a consequence of this, no more client activity will be reflected to the corresponding service through that line driver. The stop can be graceful or immediate.

Though the reusable server kernel contains a number of line drivers, the command sets understood by all of the line drivers are roughly the same. Each line driver supports START and STOP commands and a few queries. The syntax of these commands differs slightly from line driver to line driver to accommodate differences in transport attributes; for example, the TCP/IP line driver expects a port number to appear in its START command, while the spool line driver expects a file name.

For more information on the line driver commands, see Chapter 14, "Command Descriptions," on page 77.

More Detail on Line Drivers

A line driver is nothing more than a service that supplies other services with a method to interact with clients. Here is an overview and some information about how you can write your own line drivers.

Line Drivers as Services

Recall that in the reusable server kernel, a *service* is just a routine that takes input from a line driver and which delivers output to a line driver. The line driver takes care of routing data between the client and the service.

Consider also that a reusable server kernel line driver is itself a program that takes input from a client; this input is just operator commands (START, for example). Similarly, a reusable server kernel line driver is itself a program that produces output for its "client" (the operator). This output is command response text, such as the result of a LIST command.

Because of this nature of a line driver, we can see that a line driver can be implemented *as a reusable server kernel service*. To send commands to and receive responses from this service, we just have to START it through some other line driver; we would then have a means to send it commands and gather its responses.

For example, consider the TCP/IP line driver. It accepts commands -- such as START -- from its operator and produces command responses for its operator. How does it do this? Well, it does so *by way of the line driver over which it is interacting with the operator*. In other words, the TCP/IP line driver **is a *service* sourced by some other line driver**, such as the console line driver.

Continuing this, we see that if we want to issue commands to the TCP/IP line driver by using the virtual console, we must start the TCP/IP line driver by using the command CONSOLE START TCP. [9] If we also wanted to control the TCP/IP line driver by way of MSG and SMSG, we could issue MSG START TCP. After having done both of these commands, we could control the TCP/IP line driver by all of these methods:

- Typing a command on the virtual console, the first token of said command being TCP.
- Sending a CP MSG to the server virtual machine, the first token of said message being TCP.
- Sending a CP SMSG to the server virtual machine, the first token of said message being TCP.

---

[9] Note that TCP is the service name of the TCP/IP line driver.

Self-Sourced Line Drivers

Now, consider the console line driver. Like the TCP/IP line driver, the console line driver is implemented as a service. This means that the commands supported by the console line driver, such as CONSOLE START, are issued to the console line driver by way of some other line driver, and the responses to said commands are delivered to the operator through said other line driver.

For example, if we were to issue MSG START CONSOLE, we would be able to use the CP MSG command to issue commands like CONSOLE START. When we did so, the response from the console line driver would appear at the virtual machine from which we issued CP MSG, because that's how the MSG/SMSG line driver disposes of responses from the services it controls.

But look again at that console line driver. When the reusable server kernel starts, the console line driver's command set (CONSOLE START and so on) is already usable by typing those commands on the virtual console. This is possible because the console line driver is built to be *self-sourcing*. In other words, it is capable of *starting itself*, and it does so when the reusable server kernel initializes.

The CONSOLE, SUBCOM, MSG/SMSG, and SPOOL line drivers are all self-sourcing. This means that when the reusable server kernel initializes, all of the following methods are available for issuing commands to these drivers:

- You can type CONSOLE START (for example) on the virtual console and the console line driver will handle the command and write the response to the virtual console.

- You can issue a CP MSG or CP SMSG command to send a command to the MSG/SMSG line driver from elsewhere (making sure the first token of that message or special message is MSG), and the MSG/SMSG line driver will handle the command and respond to you through CP's MSG command.

- From a REXX EXEC, you can use ADDRESS RSK to issue a command to the SUBCOM line driver (making sure the first token of that command is SUBCOM), and the SUBCOM line driver will handle the command and respond by writing its output to the virtual console.

- You can send a file to the SPOOL driver; it will process the lines therein as commands and return a file to you containing the results.

Writing Your Own Line Driver

The notion that the reusable server kernel implements line drivers as services permits the server author to add his own line drivers. To add a line driver, the server author just uses ssServiceBind in his RSKMAIN to bind the service, just as he would do for any other service he writes, except:

- He must at least specify service type ss_srv_srvtype_ld in his call to ssServiceBind. This informs the reusable server kernel that the service being bound is in fact a line driver.

- If he is writing a self-sourced line driver, he must specify ss_srv_srvtype_ldss in his call to ssServiceBind. This informs the reusable server kernel that the service being bound is a self-sourced line driver.

After calling ssServiceBind, RSKMAIN should proceed as usual, eventually calling ssServerRun. These considerations apply:

- The reusable server kernel does not take any special action for regular line drivers; the server author must use PROFILE RSK to start his line driver (for example, CONSOLE START MYDRIVER to enable his line driver to interact with the server operator through the virtual console).

- For a self-sourced line driver, the reusable server kernel does the following shortly after ssServerRun begins:

  - It drives the line driver's initialization entry point (known because of the ssServiceBind call the author placed in RSKMAIN).

  - If initialization worked, the reusable server kernel *creates a thread* and runs the line driver's service routine (again, known because of the recently-performed ssServiceBind) on that thread, *passing the service routine a C-block address of X'00000000'*.

The C-block address being zero is the self-sourced line driver's cue that it should initialize its device and prepare to accept its command set over its device.

Finally, the reusable server kernel provides entry point `ssServiceFind` so that an author-supplied line driver can retrieve descriptive information saved by `ssServiceBind`. This permits author-supplied line drivers to respond to their equivalent of the IBM-supplied drivers' START command. `ssServiceFind` takes a service name as input and returns the address of the service's S-block. For more information, see Table 2 on page 6.

Some of the fields of the S-block are relevant to the server author only in the context of author-supplied line drivers. These are:

- The *current start count* is a counter used to indicate the number of START commands that are current against the service. The counter is used in this manner:

  - If the counter is zero when a line driver performs a START of this service, the line driver should drive the service's initialization routine prior to letting the service's service routine get control.

    In any case, the line driver should increment the counter just prior to driving the service's service routine.

  - When the line driver performs a STOP operation, it should first stop all its instances of the service's service routine and then decrement the counter. If the counter becomes zero as a result of this decrement, the line driver should drive the service's termination routine.

- The *lockword* is intended for use with the Compare and Swap instruction (CS). It is a line driver's means for ensuring mutual exclusion in examination and setting of the start count and in the driving of a service's initialization and termination routines. If the lock word is zero then it is considered not to be held. Any nonzero value marks the lock as held. If an attempt to get the lock through CS fails, call `ThreadYield` before trying again.

Authorization

Permission to start and stop services can be controlled through configuration parameter AUTHCHECK_LD and the AUTH command set. This capability lets the server administrator set up subordinate operators who can control some services but not others. For more information, see "Other Services' Use of Authorization" on page 40.

Chapter 3. DASD Management

Authors of certain kinds of servers will require a DASD subsystem capable of high volume, high speed, parallelized I/O with a block-oriented model. The reusable server kernel DASD subsystem meets these requirements, is integrated with CMS Application Multitasking, and recognizes the CMS thread, not the VCPU or the virtual machine, as the entity that performs DASD I/O. Specific programming information is found in the `ssSgp` API descriptions, and operator-oriented information is found in the descriptions of the SGP command set.

DASD Subsystem Overview

The reusable server kernel accomplishes its DASD objectives through the following scheme:

- Defined to the reusable server kernel are one or more sets of CMS minidisks, each such minidisk formatted at 4 KB (kilobyte) blocksize and reserved (CMS FORMAT and RESERVE commands). Such minidisks provide the raw storage for the DASD model implemented by the server kernel. Each set of such minidisks is called a *storage group*. [10]

- For each storage group, the server kernel creates one or more VM data spaces. The total number of pages in the data spaces is equal to the total number of data blocks on the constituent minidisks.

- Through MAPMDISK, each storage group's minidisk set is mapped into the pages of its data space set. [11]

- To read DASD blocks, the reusable server kernel performs MVCL from the appropriate pages in the appropriate data space. In response to this, CP pages in the mapped DASD blocks as required. Paging is a virtual machine's fastest route through CP to the DASD; further, significant amounts of real and expanded storage are used by CP on the virtual machine's behalf to "cache the DASD blocks" (that is, keep the data space pages resident).

- To write DASD blocks, the reusable server kernel performs MVCL to the appropriate pages in the appropriate data space and follows the MVCL with MAPMDISK SAVE. After MAPMDISK SAVE, the reusable server kernel waits in a thread-blocking fashion for the save-complete external interrupt to arrive. Control returns to the calling thread only when the write is entirely complete.

The techniques described above are used by the server kernel on the server application's behalf; see . [12] In addition, all code and data structures involved in this scheme exhibit the execution traits desired in a multithreaded CMS model: they are all thread-blocking, thread-synchronous, 31-bit-capable facilities.

---

[10] The reusable server kernel contains no support for linking storage group minidisks at server startup or performing the CMS FORMAT and RESERVE commands against minidisks prior to attempting to add them to a storage group for the first time. These initialization processes need to be taken care of by the server operator using traditional methods. Further, the reusable server kernel DASD engine *requires* that its storage group minidisks be formatted at 4 KB and reserved. It will not operate upon minidisks that do not meet these criteria.

[11] For FBA DASD, each minidisk must start at a multiple-of-eight block number on the real DASD volume for data space mapping to work correctly.

[12] When VM Data Spaces are not available, the reusable server kernel uses DIAGNOSE X'250' in asynchronous, MDC-enabled fashion instead; if for some reason DIAGNOSE X'250' doesn't work, then DIAGNOSE X'A4' is used.
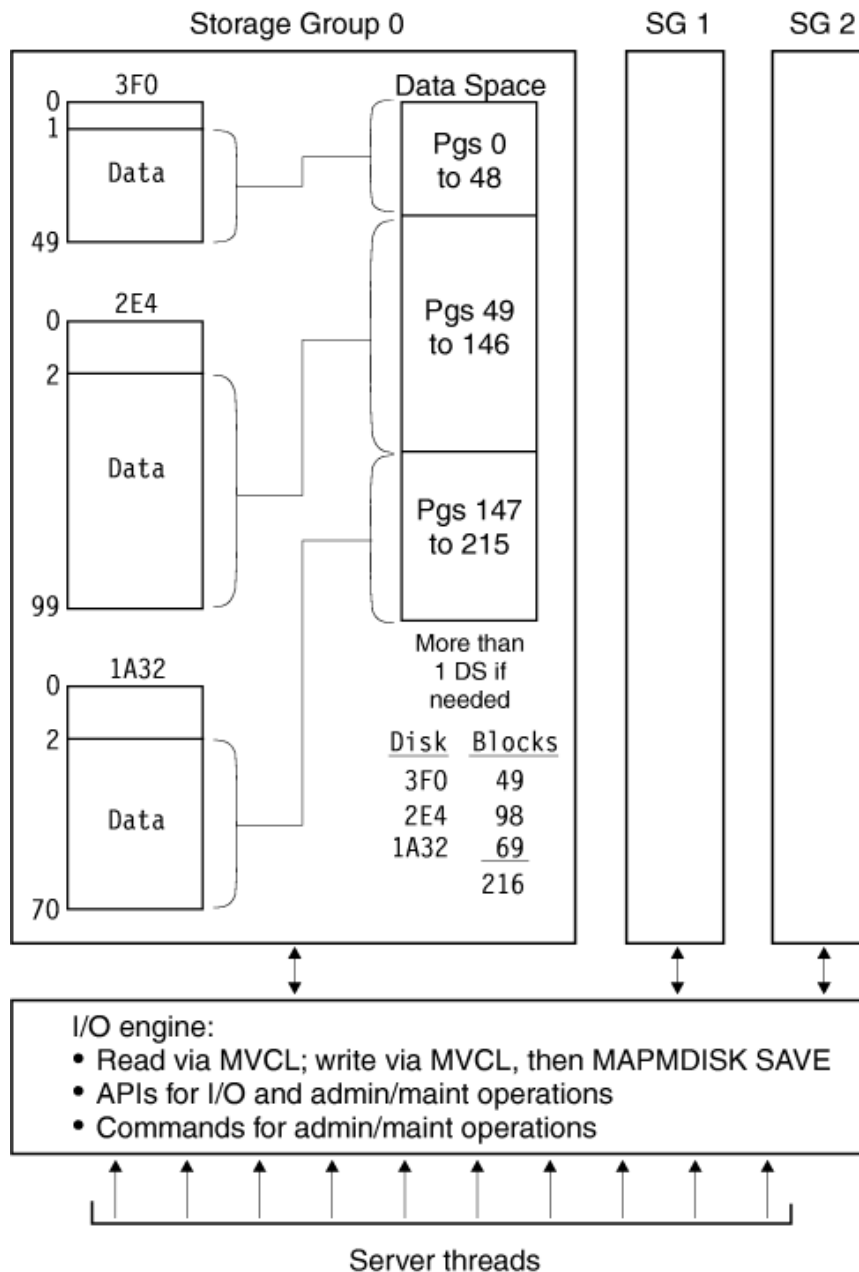
Figure 3. Reusable Server Kernel DASD

Limits

The reusable server kernel DASD subsystem exhibits these limits:

- The maximum number of storage groups is 1024.
- The maximum number of data blocks per storage group is X'FFFFFFFF' (16 TB).
- The maximum number of minidisks per storage group is 13,000.
- The total number of dataspace-mapped DASD blocks cannot exceed X'FFFFFFFF' (16 TB). [13]

Modes of Operation

A given storage group can be started in one of two I/O modes:

---

[13] The server kernel automatically switches to DIAGNOSE X'250' when this limit would be exceeded.

- *Block R/O*: the server program can read the DASD blocks but cannot write them. It is permissible for one or more of the minidisks in the storage group to be linked read-only.
- *Block R/W*: the server program can read or write the DASD blocks individually. All minidisks in the storage group must be linked read/write.

Each storage group's I/O mode is selected individually.

Programming Interfaces

Management and control of storage groups can be done through a set of storage group APIs. Callable APIs are provided to:

- Create and delete storage groups
- Vary storage groups online and offline
- List and query the defined storage groups
- Perform storage group I/O
- Find the number of a started storage group, given its name

These entry points all begin with name `ssSgp` and are described later in this book.

Administrator and Operator Considerations

A set of operator commands implements a subset of the storage group APIs. Commands are available to perform these functions:

- Create and delete storage groups
- Start and stop storage groups
- List and query the defined storage groups

For more information, see Chapter 14, "Command Descriptions," on page 77.

Creating a Storage Group

To build up a storage group from scratch, the server administrator performs these steps:

Table 14. Building a Storage Group Step

Step	Task	Command	Description	Page
1	Select some minidisks to make up the storage group.	`CP LINKCMS FORMAT`	Format each minidisk at 4 KB blocksize and reserve it. Make sure the server's virtual machine links the minidisks at startup time, for example, through `PROFILE EXEC` or `PROFILE RSK`. If FBA DASD is used, make sure each minidisk starts on a multiple-of-eight block boundary on the real FBA device.	N/A

Step	Task	Command	Description	Page
	Table 14. Building a Storage Group Step (continued)			
2	Create the storage group.	SGP CREATE	This informs the reusable server kernel of the minidisks' existence and instructs it to treat them together as a storage group. The server kernel records this information in the storage group configuration file.	"SGP CREATE" on page 174
3	Start the storage group.	SGP START	This makes the storage group available for I/O and the ssSgpRead and ssSgpWrite APIs can be used against it. You will probably want to put the SGP START command in PROFILE RSK so that the storage group starts each time the server starts.	"SGP START" on page 179

Changing the Minidisks in A Storage Group

To change the minidisk configuration of a storage group use these steps:

Step	Task	Command	Description	Page
	Table 15. Changing the Minidisk Configuration			
1	Stop the storage group	SGP STOP	This brings the storage group offline.	"SGP STOP" on page 180
2	Delete the storage group	SGP DELETE	This removes the storage group's definition from the storage group configuration file.	"SGP DELETE" on page 175
3	Create the storage group anew	SGP CREATE	This records the new storage group definition in the storage group configuration file.	"SGP CREATE" on page 174
4	Start the storage group	SGP START	This makes the storage group available for I/O.	"SGP START" on page 179

Deleting A Storage Group

To delete a storage group use these steps:

Step	Task	Command	Description	Page
Table 16. Deleting a Storage Group				
1	Stop the storage group	SGP STOP	This brings the storage group offline.	"SGP STOP" on page 180
2	Delete the storage group	SGP DELETE	This removes the storage group's definition from the storage group configuration file.	"SGP DELETE" on page 175

Chapter 4. File Caching

Servers having file-read-intensive workloads will find it beneficial to cache frequently-read files. Usually the application relies upon CMS's FSREAD cache and minidisk caching to achieve good performance, but these facilities have their limits.

To overcome these limits and extend the caching facilities available to the server writer, the reusable server kernel offers a file caching scheme based on VM Data Spaces. [14] A *file cache* is simply a data space whose contents -- files -- are controlled for the server by the server kernel. The server author decides the number and sizes of file caches he creates; he has both APIs and operator commands at his disposal for both creating and deleting file caches. Using APIs alone, the server program requests that files be cached in these data spaces; in response to the server's requests, the server kernel reads files using conventional CMS file APIs and holds them in data spaces, removing them either when they become stale or when data space storage becomes constrained. When storage constraints are an issue, the server kernel removes files in LRU (least recently used) fashion. Such removal is not visible to the server program.

Managing the Set of Caches

To create a file cache, the server operator can issue the CACHE CREATE command, or the server itself can call entry point `ssCacheCreate`. The cache is given an eight-byte name which the server kernel uses unchanged in a call to `ssMemoryCreateDS` to create the corresponding data space. Thus, cache names must be unique among all subpools the server kernel manages. The size of the cache is specified in pages.

To delete a cache, use either the CACHE DELETE command or entry point `ssCacheDelete`. The command or API call will not complete until all cached files are closed. Further, once the deletion has started, the caching of new files will not be permitted.

To obtain statistical information about a particular file cache, the server can call `ssCacheQuery`. Similarly, the server operator can issue the CACHE LIST command to see tabular output reflecting statistical information about all of the caches known to the server kernel.

For more information on how the server kernel maintains monitor data for each file cache, see Chapter 13, "Monitor Data," on page 71.

File Operations

To cache a file, the server calls entry point `ssCacheFileOpen`, supplying the name of the file to be cached. Any name acceptable to CSL routine DMSOPEN can be used. The server kernel keeps track of cached files using these DMSOPEN-acceptable names. In response to the call, the server kernel loads the file into the cache, making it ready for reading through another entry point, `ssCacheFileRead`; in addition, if the server kernel was able to load the file contiguously into data space storage, it informs the caller of this, returning to it the ALET and address the server can use to access the cached file directly. In any case, `ssCacheFileOpen` returns the size in bytes of the cached file. Finally, note that the file can be opened multiple times simultaneously; this permits open-read-close logic to be applied freely on a per-client basis.

Once the server has opened the file, it can read the file's data through one of two methods:

- If the file was loaded contiguously, the server can enter AR mode and read the data directly from the data space, using the ALET, address, and length returned by `ssCacheFileOpen`.
- If the file was not loaded contiguously, or if the server author chooses not to use AR mode, the server can call entry point `ssCacheFileRead` to read the data. This entry point's inputs are simply a file token, a zero-origin byte offset, and a length. It simply reads the cached data into the buffer passed by the caller. The server kernel permits multiple `ssCacheFileRead` calls to be in progress simultaneously against a given file.

---

[14] If VM Data Spaces are not available, the file caching facilities of the reusable server kernel do not work.

When the server is done reading the file, it issues call `ssCacheFileClose`. The file remains in the cache for subsequent use, unless it becomes stale or is pushed out because of storage contention.

Transformations

Recognizing that the server is likely to need to perform code page transformations on the files it manipulates, the server kernel includes a translation function with its caching support. When the server opens a file, it specifies a translation table to be applied to the file's data as it is loaded into the cache. The translation table can come from these places:

- The server kernel offers an entry point, `ssCacheXlTabSet`, which the server can call to identify a translation table that should be eligible for use as part of file loading. The table is known by an integer identifier and is nothing more than a 256-byte table to be applied to the file's data using the Translate (TR) instruction. The integer identifier supplied to `ssCacheXlTabSet` is also one of the inputs to `ssCacheFileOpen`.

- For the server author's convenience, the server kernel predefines certain tables:

Table ID	Table Function
0	No translation at all
1	1047 to 819 (EBCDIC to ASCII)
2	819 to 1047 (ASCII to EBCDIC)

The server kernel recognizes these tables' identifiers without the server having to invoke `ssCacheXlTabSet` first.

Just as it might have to perform code page translation on files it serves to clients, the server might also have to perform record boundary delimiter transformations. For example, a UNIX® client might want the records to be delimited by a line feed (X'0A'), while a DOS client might want a carriage return and line feed (X'0D0A') at the end of each record. Depending on the file's contents, it might even be appropriate not to insert any delimiters at all - a .JPG file, for example, falls into this category. Recognizing this, the server kernel lets the caller tell `ssCacheFileOpen` what should be done about record delimiting. Both line-end marker and prefixed record-length schemes are supported.

Example

Suppose that an HTTP server needs to serve file `INDEX HTML VMHOME:EWEBADM.VMPAGE` to a browser. As part of serving this file to the browser, the HTTP server will need to translate the file's data from EBCDIC to ASCII and will need to insert a CR-LF pair (X'0D0A') after each record. To serve the file, the server would call `ssCacheFileOpen`, requesting that appropriate data translation and record massaging be done as part of the load into the cache. The server kernel would return a *file token* as an output of `ssCacheFileOpen`, and if the file had been loaded contiguously into the data space, it would also return the ALET and address of the data space buffer in which the file resides. Finally, if the load is successful, `ssCacheFileOpen` also returns the size in bytes of the loaded, transformed file.

The server can read the file's contents using either `ssCacheFileRead` or AR mode. However, if all that is needed is to send the file's contents to the browser, the server can just call `ssClientDataPut`, passing it the ALET, address, and length returned by `ssCacheFileOpen`.

After the file has been sent, the server issues `ssCacheFileClose`. The file remains in the cache for the next client.

Stale Data

The server kernel's file caching scheme accommodates the notion that file contents change over time and that cached information can become stale as a result. When the server calls `ssCacheFileOpen`, the server kernel checks the file's update time and compares it against the update time of the cached copy. If there is a discrepancy, the file is reloaded. The currently cached copy -- now stale -- is disposed of

according to whether it is still in use (that is, is still open); if it is not in use it is dropped immediately, but if it is still open it is marked as stale and dropped when the server finally closes it. This scheme preserves consistency for open files while providing a means for new opens to see the latest version of a file.

Cache Utilization

It is important to recognize that the server kernel can have more than one copy of a file in a cache at one time. This can happen in these situations:

- If a file is loaded into a cache using several different code page translations or several different record delimiting schemes, a cached copy will be kept for each such representation requested. For example, if `INDEX  HTML` were opened using EBCDIC-to-ASCII and CR-LF delimiting, and then it were opened again using no code page translation and CMS two-byte-length record prefixing, the server kernel would keep both copies in memory.
- If a cached file is still open, it will not be dropped from the cache, even if the server kernel detects that it has become stale. The stale file will not be dropped until it is closed.

Constraints

The server author and administrator should keep these file caching constraints in mind:

- File cache names are used directly as input to `ssMemoryCreateDS`. The server writer and administrator must work to avoid name conflicts.
- The number of files that can be held in a cache is not strictly limited, but the overall size of a file cache is limited to 2 GB (the size of a data space). This means that a transformed file cannot exceed 2 GB. Note that multiple file caches are supported.
- The number and aggregate size of data spaces creatable by the server is controlled by `XCONFIG ADDRSPACE` in the server virtual machine's CP directory entry.
- Files whose transformed size would be greater than 16 MB (megabytes) are never cached contiguously.

Chapter 5. Authorization

Overview

Many servers appear to their clients as access methods for server-held objects. File system servers are a common example of these. For example, the CMS Shared File System implements an object class called *file* supporting a certain set of operations and an object class called *directory* supporting another set of operations. The users of the Shared File System transmit requests to an SFS server, asking the server to perform operations on these objects. The SFS server performs the operations and returns appropriate responses to the clients. No operations are possible against SFS-held objects other than those defined on objects of class *file* or *directory*.

Servers implementing such access methods usually require that the operations requested by clients be performed on the objects if and only if certain authorization guidelines are met. Consider again the Shared File System: to write to a file, a user must have write authority to it. [15] To support this checking of operations, the Shared File System contains its own authorization engine for managing the authorization rules. The authorization model used by the CMS Shared File System is built around objects, users, and actions; all of the interfaces to the authorization engine serve to manipulate and interrogate a rule base which records "who can do what to whom". Some of these interfaces, such as the GRANT AUTHORITY and REVOKE AUTHORITY commands, are externalized. Others are internal-only interfaces for the server's exclusive use.

The general model for authorization exemplified by the Shared File System applies to many different kinds of servers. To ease the development burden of the server writer, the reusable server kernel provides a set of APIs implementing a general-purpose authorization engine. The authorization model implemented by the reusable server kernel is an object-user-action model, just like the one implemented by the Shared File System. To use the reusable server kernel's authorization facility, the server author calls the API, performing such actions as defining an object class, defining a particular object, permitting a user to perform an operation, and testing whether an operation is permissible. A set of commands, intended for operator use, parallels the APIs provided.

The reusable server kernel authorization engine treats object classes, object names, user names, and permissions as abstract entities. It does not associate any particular meaning with these items. It merely facilitates the implementation of an authorization strategy by providing a rule engine capable of building, maintaining, and interrogating a rule base describing a relationship of objects, users, and actions. The object classes and operations defined, the objects defined, the users defined, and the permissions granted are left for the server writer to decide.

Entry Points

The reusable server kernel authorization API offers entry points that perform a number of different operations on the rule base. In particular, these are some of the programming interfaces available:

Table 17. Programming Interfaces

Programming Interfaces	Description	Page
ssAuthCreateClass	Creates an object class and associates a set of operations with it.	"ssAuthCreateClass — Create an Object Class" on page 217
ssAuthCreateObject	Creates a named object as an instance of a particular object class.	"ssAuthCreateObject — Create an Object" on page 219

---

[15] In truth, to *open* a file for write, the user must have write authority to it, even if he never actually writes to the file.

Table 17. Programming Interfaces (continued)		
Programming Interfaces	**Description**	**Page**
ssAuthDeleteClass	Removes all objects of a given class from the rule base and optionally removes the class from the rule base.	"ssAuthDeleteClass — Delete a Class" on page 221
ssAuthDeleteObject	Removes all rules for a given object from the rule base and optionally removes the object from the rule base.	"ssAuthDeleteObject — Delete an Object" on page 223
ssAuthDeleteUser	Removes all rules for a given user from the rule base.	"ssAuthDeleteUser — Delete a User" on page 225
ssAuthPermitUser	Adds, modifies, or deletes a specific rule in the rule base.	"ssAuthPermitUser — Permit a User" on page 233
ssAuthTestOperations	For a given user, object, and set of operations, determines which of the specified operations are permissible.	"ssAuthTestOperations — Test Operations" on page 242

A set of queries and some maintenance APIs are also provided.

Naming Conventions and Other Limits

To name objects, users, classes, and permissions, the authorization API uses character strings composed from an unrestricted alphabet. [16]

Table 18 on page 36 describes other conventions related to the naming of these items:

Table 18. Authorization API Naming Conventions		
Item	**Format**	**Length**
Object	V	1-256
User	V	1-64
Class	F	8
Action	F	4
Note:		
• The authorization API supports a maximum of 32 operations per object class.		

Group Authorization Considerations

The reusable server kernel's authorization model and API extend easily to group authorization situations. [17] To implement a group scheme, the programmer can perform the mapping of user ID to group name outside the scope of the reusable server kernel's authorization API and use the group names as "user IDs" in the reusable server kernel authorization API calls. In cases where group authorization provides acceptable security, using the authorization API in this way reduces the size of the authorization data and thereby decreases the time needed to search it.

---

[16] "Unrestricted alphabet" means that any of the 256 8-bit code points can appear in these names.

[17] In *group authorization*, access rights are extended to users not based on their individual identities but rather on their membership in a group of some kind. Unix and VMS are two systems where file authorization is based partially on users' organization into groups.

Persistent Storage of Authorization Data

The reusable server kernel keeps the authorization database in several disk files. These disk files let the authorization data persist from one invocation of the server program to the next.

The general idea is that the authorization database is divided into several files:

Table 19. Authorization Data File Format		
File Format	**Description**	**Page**
Data	Contains class, object, user and rule definitions. The records in this file are chained to one another to build up logical groupings, such as the set of rules associated with a given object or the set of objects belonging to a given class.	"The Data File" on page 367
Index	Contains hash tables that partition the data file records into equivalence classes (that is, hash buckets) to improve the performance of searches.	"The Index File" on page 369
Log	Contains all tracking of the writes to the index and data files for recovery purposes.	"The Log File" on page 370

The reusable server kernel is able to keep its authorization data in any of these disk repositories: [18]

- On CMS minidisks
- In the CMS Shared File System

All of the authorization files must be kept in the same kind of repository. Mixing repositories is not permitted.

Recognizing the critical nature of authorization data, the reusable server kernel manages its authorization files such that the authorization database can be recovered (that is, its internal consistency can be restored) if some kind of failure occurs. The management and recovery scheme used is a function of the repository in which the data files reside. When CMS minidisks are used, the reusable server kernel keeps twin copies of the authorization database and also keeps a log file to enable recovery after a failure. When the CMS Shared File System is used, just one copy of the authorization database is kept and the Shared File System's commit/backout facilities are exploited to maintain consistency.

When the reusable server kernel starts, it initializes the authorization data base (makes it completely empty) if it appears that the database has never been initialized. This assessment is made using the following criteria:

- *Shared File System:* if the index file appears not to be initialized, then an empty index is written.
- *Minidisks:* if the log file appears not to be valid, or if the log file appears valid but the index file appears not to be initialized, then an empty index is written.

You should back up your authorization index and data files frequently enough so that you can restore them without loss of data in case they are initialized accidentally.

The following sections give more specifics on the details of the various repositories.

Using CMS Minidisks

To keep the authorization files on minidisks, set configuration parameter AUT_LOCATION appropriately and supply names for:

---

[18] Configuration parameter AUT_LOCATION file tells the reusable server kernel where the data is being kept.

- Copy 1 of the data file (configuration parameter AUT_DATA_1),
- Copy 2 of the data file (configuration parameter AUT_DATA_2),
- Copy 1 of the index file (configuration parameter AUT_INDEX_1),
- Copy 2 of the index file (configuration parameter AUT_INDEX_2),
- The authorization log file (configuration parameter AUT_LOG).

These files do not all have to be on the same minidisk; you can spread them across minidisks if you want. [19] The only constraint is that *for each minidisk on which authorization files reside, there must be no open-for-output files on the minidisk other than the authorization files themselves*. In other words, do not put any of your server's other output files on the same minidisk with authorization data files. If this constraint is not observed then the reusable server kernel's commit and recovery logic **will not work** and if a failure occurs you might end up with unrecoverable authorization data.

When minidisks are used, the reusable server kernel guarantees consistency by using the log file to record changes that will be made and then applying the changes to the two copies sequentially. If an entire update does not complete successfully, the reusable server kernel uses the log file to decide how to recover the consistency of the authorization data and make the two copies identical again. If the update was completely applied to the first copy and then the update of the second copy failed, realigning the two copies does not lose the update. If the update was never completely applied to the first copy, the update will be backed out.

Using the CMS Shared File System

To use the CMS Shared File System, set configuration parameter AUT_LOCATION appropriately and supply names for:

- Copy 1 of the data file (configuration parameter AUT_DATA_1),
- Copy 1 of the index file (configuration parameter AUT_INDEX_1),

The data and index files need not reside in the same directory or even the same file pool server. [20] The directories in which the files reside can be accessed directories or unaccessed directories.

When the Shared File System is used, the reusable server kernel does not maintain a second copy of the data and index files and it does not keep a log file; it ignores the configuration parameters associated with these extra files (configuration parameters AUT_DATA_2, AUT_INDEX_2, and AUT_LOG). This is made possible because the Shared File System supports commit and backout semantics; the reusable server kernel does not have to manage recovery on its own.

When the Shared File System is used, the reusable server kernel uses this technique to maintain consistency of the authorization data:

1. At startup, the reusable server kernel gets a work unit ID and opens the two files on that work unit.
2. Each time an API call changes the database, the reusable server kernel writes the changes to the index and data files and then commits the work unit.
3. If one of the writes fails or the commit fails, the reusable server kernel backs out the work unit.

This method guarantees that the index and data files are always committed together and that the committed copies are always consistent with one another.

Migrating Among Repositories

To migrate your authorization data to the Shared File System from minidisks, follow the instructions in .

---

[19] In fact, it would be a good idea to put the files for copy 1 on one physical DASD pack and the files for copy 2 on a different physical DASD pack.

[20] If you put the two files in two different servers, each server must be at least VM/ESA 1.1 or later.

Table 20. Migrating Authorization Data from Minidisks to SFS			
Step	**Description**	**Command**	**Page**
1	Make sure the server shuts down normally so that the two copies of authorization data are each internally consistent and identical to one another.	SERVER STOP	"SERVER STOP" on page 173
2	Move one copy to the desired Shared File System server(s) and directory(ies).	CMS's COPYFILE	n/a
3	Change the reusable server kernel's AUT_ configuration parameters to reflect the new names and locations of the authorization data.	Use XEDIT to change PROFILE RSK.	"Configuration Parameters" on page 65

Migrating from the Shared File System to minidisks is a little more complicated; follow the instructions in

Table 21. Migrating Authorization Data from SFS to Minidisks			
Step	**Description**	**Command**	**Page**
1	Duplicate your index and data files so that you have two identical copies of each (four files in all).	CMS's COPYFILE	n/a
2	Install the copies on the target minidisks.	CMS's COPYFILE	n/a
3	Using CMS Pipelines, an EXEC, XEDIT, or some other tool, make a file of the following format and content (this will be the initial log file): • RECFM F • LRECL 256 • Put one record in the file. The first twelve bytes of the record should be X'000000020000000200000000'. The content of the remainder of the record is unimportant. Install this file on the target minidisk.		n/a
4	Update your reusable server kernel configuration parameters to point to the new target repository and update the names of the index, data, and log files.	Use XEDIT to change PROFILE RSK.	"Configuration Parameters" on page 65

Parallelism

The reusable server kernel lets multiple threads read the authorization data simultaneously but requires updating threads to serialize and perform their work exclusively of all other threads (in other words, either multiple readers are allowed or one writer is allowed).

Administrative Commands

The reusable server kernel provides a service, called AUTH, which provides a command interface to many of the authorization APIs. This command set is useful in thses circumstances:

- Commands to manipulate the authorization database can appear in `PROFILE RSK` and be issued each time the server starts.
- An operator can manipulate the authorization database by sending authorization commands to the AUTH service through the `CP MSG` command or by typing them on the server console.

For more information on the authorization command set, see Chapter 14, "Command Descriptions," on page 77

Other Services' Use of Authorization

The presupplied services and line drivers are capable of using the authorization database as a way to protect their command sets. For example, the AUTH service -- that is, the implementer of the AUTH command set -- offers a means by which the server administrator can instruct it to examine the authorization database to determine whether a certain user is permitted to issue AUTH commands. The starting and stopping of author-supplied services can be similarly protected.

As shipped, all such controls are inactive, that is, no permission checking is in effect. The following sections describe how such authorization checking can be activated.

Overview

The basic idea is that certain services and line drivers interrogate a corresponding configuration parameter to decide whether to check authorizations for the command sets they implement. When a service or line driver's authorization configuration parameter is set `ON`, the service or line driver calls `ssAuthTestOperations` each time it handles a command. The purpose of this call is to determine whether the requesting user has permission to issue the prospective command. If the call to `ssAuthTestOperations` succeeds, the line driver or service will attempt the requested operation. Table 22 on page 40 generally illustrates how a line driver or calls `ssAuthTestOperations`.

Table 22. Line Driver and Service Calls to ssAuthTestOperations

Coordinate	Value
Object	Name of the service being manipulated.
User	The user ID attempting to manipulate the service.
Action	For a start, STRT. For a stop, STOP. For connection reporting, RPRT. For actual use thereof, EXEC.

Activation

To activate authorization checking for line drivers and services, perform the these initialization steps with respect to the authorization database:

Table 23. Activating Authorization Checking for Services and Line Drivers

Step	Task	Command	Description	Page
1	Create an object class to which objects representing services will belong.	AUTH CRECLASS	The name of the object class is not important, but operations STRT, STOP, RPRT, and EXEC must be defined on objects of the class.	"AUTH CRECLASS" on page 88
2	Create an authorization object corresponding to the service that will be protected.	AUTH CREOBJECT	You should create the new object as a member of the class you just created with AUTH CRECLASS. The name of the new object should match the name of the service as it was given in the ssServiceBind API call.	"AUTH CREOBJECT" on page 89
3	Grant privileges for each user who will be permitted to START the service.	AUTH PERMIT	Arrange for the user ID to have permission to perform the STRT operation on the object that represents the service.	"AUTH PERMIT" on page 96
4	Grant privileges for each user who will be permitted to STOP the service.	AUTH PERMIT	Arrange for the user ID to have permission to perform the STOP operation on the object that represents the service.	"AUTH PERMIT" on page 96
5	Grant privileges for each user who will be permitted to enable a line driver's connection reporting feature.	AUTH PERMIT	Arrange for the user ID to have permission to perform the RPRT operation on the object that represents the service.	"AUTH PERMIT" on page 96
6	Grant privileges for each user who will be permitted to use a given service.	AUTH PERMIT	Arrange for the user ID to have permission to perform the EXEC operation on the object that represents the service.	"AUTH PERMIT" on page 96

Once the authorization database has been set up, it remains to inform line drivers and services that they should actually *check* the authorization data you've configured. This is accomplished by using the CONFIG commands:

- To enable line drivers' checking of your newly-created authorization records, issue CONFIG AUTHCHECK_LD ON. When you do this, each line driver will handle a given user's START or STOP commands only if the authorization data permits it.
- To inform a given service that it should check your newly-created authorization records, set the service's appropriate configuration parameter (see Table 24 on page 42 and Table 31 on page 66).

Table 24. Authorization Configuration Parameters

Service	Parameter	Page
AUTH	AUTHCHECK_AUTH	"CONFIG AUTHCHECK_AUTH" on page 112
CACHE	AUTHCHECK_CACHE	"CONFIG AUTHCHECK_CACHE" on page 113
CMS	AUTHCHECK_CMS	"CONFIG AUTHCHECK_CMS" on page 114
CONFIG	AUTHCHECK_CONFIG	"CONFIG AUTHCHECK_CONFIG" on page 115
CP	AUTHCHECK_CP	"CONFIG AUTHCHECK_CP" on page 116
ENROLL	AUTHCHECK_ENROLL	"CONFIG AUTHCHECK_ENROLL" on page 117
MONITOR	AUTHCHECK_MONITOR	"CONFIG AUTHCHECK_MONITOR" on page 119
SERVER	AUTHCHECK_SERVER	"CONFIG AUTHCHECK_SERVER" on page 120
SGP	AUTHCHECK_SGP	"CONFIG AUTHCHECK_SGP" on page 121
TRIE	AUTHCHECK_TRIE	"CONFIG AUTHCHECK_TRIE" on page 122
USERID	AUTHCHECK_USERID	"CONFIG AUTHCHECK_USERID" on page 123
WORKER	AUTHCHECK_WORKER	"CONFIG AUTHCHECK_WORKER" on page 124

All of the aforementioned configuration parameters can be set in PROFILE RSK each time the server starts. For more information, see .

Chapter 6. Enrollment

One problem common to many servers is the notion of enrolling users. In the abstract, this problem is nothing more than implementing or exploiting some kind of indexed access method. Users' records are kept in a repository of some kind and inserted, removed, and retrieved using the chosen access method, the user identifiers serving as indices.

Recognizing this commonality, the reusable server kernel implements an indexed access method suitable for use in storing enrollment data. The server kernel offers an API for programmed manipulation of enrollment sets -- record insertion, deletion, and retrieval, to name a few operations -- and it offers a corresponding command set that lets the server operator perform these operations easily. The command set is implemented as a service, so it is available through any of the server kernel's line drivers - CONSOLE, MSG, and so on.

The reusable server kernel stores related enrollment records together in an enrollment *set*. Each enrollment set bears an eight-byte name; the server operator refers to an enrollment set by that name when he uses the ENROLL command set, and the server author refers to an enrollment set by that same name when he uses the enrollment API. The server kernel can manage multiple enrollment sets concurrently.

To ensure good performance, the reusable server kernel exploits VM Data Spaces to hold enrollment sets. When the server kernel is instructed to make an enrollment set ready for use, it reads the enrollment records from a Shared File System file into a VM Data Space, organizing them in the data space for quick access. Each enrollment set resides in its own data space, and a data space being used for enrollment contains nothing but records of that enrollment set. Note that the reusable server kernel's enrollment facility requires the underlying processor to support VM Data Spaces. Processors not offering VM Data Spaces cannot support the enrollment facility.

Because a data space can be up to 2 GB in size, and because z/VM lets a single virtual machine manage many such data spaces concurrently, the number of enrollment records the reusable server kernel can manage has no limit, practically speaking. The data structures used ensure that the server kernel can hold several hundred thousand enrollment records in a single data space without appreciable lookup, insertion, or replacement delays.

As the enrollment records change, the reusable server kernel appends information to the corresponding SFS file, said appended records being indicative of the changes that are occuring against the enrollment set. At an appropriate time, the operator or the server program itself indicates that it is time to commit the changes; in response to this, the server kernel uses CSL routine DMSCOMM to commit the changes to the SFS file. Each enrollment set's corresponding SFS file is open on its own work unit, each such work unit being used for no other purpose than I/O to a single enrollment file.

Eventually the server operator or server program determines that activity to an enrollment set is complete and instructs the reusable server kernel to unload the enrollment data. Theserver kernel closes the corresponding SFS file, deletes the data space, and the enrollment set is thereby closed. If the server program terminates and the enrollment set is still open, the server kernel closes it automatically before terminating, committing any uncommitted changes. If the Shared File System should ever indicate that it cannot commit changes, the reusable server kernel backs out the changes, using SFS's rollback support.

Because of the cumulative nature of the SFS file that holds an enrollment set, it is occasionally helpful to remove redundant information from such a file. An EXEC to perform such cleanup is provided. When an enrollment set is being cleaned, it cannot be in use for any other purpose; it must be unloaded prior to being cleaned and reloaded afterward.

Each enrollment record consists of a 64-byte key and a corresponding piece of enrollment data. The reusable server kernel imposes no structure on the enrollment data itself; the structure of the enrollment data is left to the server author. However, the server kernel does impose the restriction that an enrollment record cannot contain more than 65,450 bytes of data (this limit comes from the record-length limit of CMS file systems). Zero-length data is permitted on enrollment records.

Last, recognizing the utility of a general-purpose indexed access method capable of holding data on this scale, the reusable server kernel implements *transient* enrollment sets. A transient enrollment set is empty when opened, is never written to disk, and all memory of it is lost when it is closed. While it is open, though, all of the server kernel's indexing and retrieval facilities are available, and VM Data Spaces are exploited just as they are for permanent enrollment sets. This gives the server author a way to keep track of large numbers of tagged, transient data items concurrently. Said data items can be stored in an enrollment set, where the reusable server kernel keeps them in a VM Data Space until they are again requested by the server program. Note also that because transient enrollment data is never written to a CMS file, it is not necessary for the reusable server kernel to limit the data length quite so much. For transient enrollment sets, the amount of data that can be stored in a given record is limited to 16 MB - the maximum amount movable through the Move Long (MVCL) instruction.

Programming Interfaces

The server program can use the following programming interfaces to manipulate enrollment sets:

Table 25. Enrollment APIs

Programming Interface	Description	Page
ssEnrollCommit	Commit changes to an enrollment set.	"ssEnrollCommit — Commit Enrollment Set" on page 264
ssEnrollDrop	Close a permanent enrollment set, either committing or rolling back the uncommitted changes, or destroy a transient enrollment set.	"ssEnrollDrop — Drop Enrollment Set" on page 266
ssEnrollList	Generate a list of the enrollment sets currently loaded.	"ssEnrollList — List Enrollment Sets" on page 268
ssEnrollLoad	Load an enrollment set from an SFS file into a VM Data Space, or initialize a transient enrollment set.	"ssEnrollLoad — Load Enrollment Set" on page 270
ssEnrollRecordGet	Retrieve a record from an enrollment set.	"ssEnrollRecordGet — Get Enrollment Record" on page 272
ssEnrollRecordInsert	Insert a record into an enrollment set.	"ssEnrollRecordInsert — Insert Enrollment Record" on page 274
ssEnrollRecordList	Generate a list of the indicies of all the records in the enrollment set.	"ssEnrollRecordList — List Records In Enrollment Set" on page 276
ssEnrollRecordRemove	Remove a record from an enrollment set.	"ssEnrollRecordRemove — Remove Enrollment Record" on page 278

Operator Commands

The ENROLL service implements a set of operator commands:

Table 26. Enrollment Commands

Command	Description	Page
COMMIT	Commits changes to an enrollment set.	"ENROLL COMMIT" on page 152

Table 26. Enrollment Commands (continued)		
Command	**Description**	**Page**
DROP	Unloads an enrollment set from a data space.	"ENROLL DROP" on page 153
GET	Retrieves a record from an enrollment set.	"ENROLL GET" on page 154
INSERT	Inserts a record into an enrollment set.	"ENROLL INSERT" on page 155
LIST	Generates a summary of the loaded enrollment sets.	"ENROLL LIST" on page 156
LOAD	Loads an enrollment set into a data space.	"ENROLL LOAD" on page 157
RECLIST	Generates a list of the keys of the records in an enrollment set.	"ENROLL RECLIST" on page 158
REMOVE	Removes a record from an enrollment set.	"ENROLL REMOVE" on page 159

Chapter 7. Indexing by Prefixes

Overview

The reusable server kernel's enrollment API provides a simple indexed access method that lets the server author use a fully-formed index to return exactly one record whose key matches the supplied fully-formed index. This solves the enrollment problem well but ignores a large class of indexing problems relevant in server development. In particular, it ignores the problem of returning a set of records whose keys are matched by a prefix the caller supplies. This problem appears in many situations, such as telephone directory lookup or web page indexing.

The reusable server kernel contains APIs that let the server application build and interrogate indices that permit the retrieval of record sets according to lookup by prefix. For each such index, the reusable server kernel APIs provide insertion and lookup operations, identifying the inserted or retrieved records by record number (the indexing API holds onto record *numbers*, **not** records themselves). The reusable server kernel keeps each such index in its own VM Data Space and lets multiple RSK-based service machines access the indices concurrently. An index does not persist across invocations of the server program; the server must rebuild the index each time it starts.

More specifically, the provided APIs are:[21]

- `ssTrieCreate`: creates an index. The caller specifies a name for the index and the size (in pages) for the index. The reusable server kernel creates a data space to hold the index and returns the ASIT and ALET to the caller.
- `ssTrieDelete`: destroys an index. The reusable server kernel destroys the corresponding data space.
- `ssTrieRecordInsert`: the caller supplies the index name, a record number, and the key to be associated with the record number. The reusable server kernel inserts the record number into the index.
- `ssTrieRecordList`: the caller supplies an index name and a key prefix. The reusable server kernel searches the index and returns a list of all the record numbers whose corresponding keys match the prefix specified by the caller.

Example

Suppose a company phone book is contained in a CMS F-format file, with the 40-column employee name appearing in columns 36 to 75. An RSK-based phone directory lookup engine might read the phone file into memory and then form an index on the employee names. To index each record, the engine would call `ssTrieRecordInsert`, identifying the record by number and supplying the 40-column employee name field as the record's key. Once all records have been indexed, the server is ready to begin servicing lookup requests; given a prefix, the engine can call `ssTrieRecordList`, thereby retrieving the record numbers of all the records whose key matches the prefix of interest.

Index Sharing

An application using the trie APIs will probably work alone most of the time, that is, its indices will be private. In this manner of operation, the application creates the index by name and then refers to it by name when performing insertion and lookup operations.

However, the reusable server kernel does provide the basic structure necessary for the application to share an index among multiple virtual machines (for example, worker machines). When `ssTrieCreate` creates an index, it supplies the caller with the ASIT and ALET of the data space containing the index. If the application desires to share the index with (for example) a worker machine, it should call CSL routine

---

[21] The APIs take their name from the data structure used to implement the index. This data structure is called a *trie* (rhymes with *sky*) and is described, for example, in Aho, Hopcroft, and Ullman, *Data Structures and Algorithms*, Addison-Wesley, 1985, ISBN 0-201-00023-7.

DMSSPCP to permit the worker to access the index data space read/write and then it should send the ASIT to the worker. The worker should use DMSSPLA to generate its own ALET for the space and then call the trie APIs as appropriate, identifying the index by ALET. Note that the worker must have read/write access to the data space, even if it is performing only lookups. This is because the trie APIs use storage in the data space to implement necessary locking primitives.

The reusable server kernel makes no attempt to recover from program checks that will occur in worker machines if the owning virtual machine should delete the index. When deletion of an index (that is, a call to `ssTrieDelete`) is required, the application must take care to inform the workers and receive their acknowledgements prior to deleting the index.

No Record Deletion?

For reasons of complexity, there is no `ssTrieRecordDelete` function. If it becomes necessary to "delete a record", the application should simply ignore that record's number when it appears in the output of `ssTrieRecordList`.

Commands

A very simple built-in service, TRIE, offers a command, LIST, that can be used to display pertinent information about the indices the server has created. For each such index, the reusable server kernel displays the index name and ASIT, the index size, the amount of data space storage actually being used, the number of records being held, and the number of nodes in the trie.

There are no command equivalents for the `ssTrieRecordInsert` and `ssTrieRecordList` entry points.

Chapter 8. Anchors

The reusable server kernel lets the application set and query the value of an application-wide anchor word. This is similar in intent to CMS's ANCHOR macro and its `ThreadSetUserData` and `ThreadQueryUserData` CSL routines. Unlike ANCHOR, the reusable server kernel facility is callable. Unlike the thread functions, the reusable server kernel facility provides application-wide scope.

A server program would typically use the anchor services for holding the address of some server-wide control block. This control block would typically be acquired early in the server's life and the `ssAnchorSet` function would be called to record the address of this control block. When the address of the control block is required, the server can call `ssAnchorGet` to retrieve the control block's address.

Note also that `ssAnchorGet` returns the address and length of the buffer in which the server may place data to be accrued by the CP monitor (APPLDATA -- DIAG X'00DC').

The reusable server kernel does not use CSL routines `ThreadSetUserData` or `ThreadQueryUserData`. The server writer is free to use these routines as he wishes.

The ANCHOR macro works correctly only in virtual uniprocessor situations. It is not recommended for use in virtual multiprocessor situations.

Chapter 9. Memory Management

Fast, efficient allocation and release of primary storage (memory) is vital to the execution of a server program. CMS provides the CMSSTOR facility for storage management; CMSSTOR works very well for single-threaded, assembler-only, base-VCPU-only programs, but for multithreaded, parallel servers CMSSTOR shows its limits. In particular, the following characteristics of CMSSTOR are undesirable for server writers:

- *Base-only execution:* though the macro can be invoked from non-base processors, CMSSTOR actually runs on the base VCPU. This means that the base VCPU becomes a serialization point for the server.
- *Assembler only:* callable support is not provided.
- *Base address space only:* CMSSTOR is not capable of managing storage in a data space.

To overcome these difficulties, the reusable server kernel implements a "front end" for CMSSTOR whose purpose is to relieve these constraints. The following entry points are provided:

- `ssMemoryCreateDS:` creates a data space and prepares to manage the storage thereof. The caller sees the data space as a subpool.
- `ssMemoryAllocate:` allocates storage, either from a data space or the primary address space.
- `ssMemoryRelease:` releases storage.
- `ssMemoryDelete:` deletes a subpool and the corresponding data space.

For management of data space storage, the reusable server kernel storage management facility provides an interface that lets the caller see a data space as a subpool, as follows:

- To create a data space and assign a subpool name to it, the caller invokes `ssMemoryCreateDS`, passing it the subpool name to use and the size of the data space. Subject to any constraints imposed by the virtual machine's `XCONFIG ADDRSPACE` directory entry, the reusable server kernel creates the data space, prepares to manage the storage therein, and returns to the caller the new data space's ASIT and ALET.

  `ssMemoryCreateDS` accepts a storage key and option array on input and passes these directly to CSL routine DMSSPCC (Create Data Space). If the caller of `ssMemoryCreateDS` supplies a zero-length option array, `ssMemoryCreateDS` uses all of DMSSPCC's defaults, except that the data space is created SHARE.

  Regarding establishing addressability to the data space, `ssMemoryCreateDS` calls DMSSPLA with the WRITE and SYNCH options.

- To allocate and release storage in the data space, the caller uses `ssMemoryAllocate` and `ssMemoryRelease`, referring to the data space by its subpool name.
- To delete the data space, the caller uses `ssMemoryDelete`.

For the primary address space, the reusable server kernel storage management facility is a front-end for CMSSTOR, as follows:

- For each subpool name ever used in a call to (that is, "seen by") `ssMemoryAllocate`, the reusable server kernel keeps track of storage allocated through `ssMemoryAllocate` and storage released through `ssMemoryRelease`. In other words, for each subpool, the reusable server kernel maintains a free storage subpool cache that can be manipulated without serializing on the base VCPU.[22]
- When `ssMemoryAllocate` is called, it performs the following steps in an attempt to locate storage for the caller:

---

[22] In fact, non-trivial serialization occurs only when two VCPUs try to manipulate the same subpool.

Step	Description
1	The subpool's cache is checked, and if *max_bytes_needed* can be satisfied from there then the request completes.
2	CMSSTOR OBTAIN is consulted in variable fashion, the lower bound being the largest qualifying size available in the cache (or *min_bytes_needed*, if all cache pieces are too small) and the upper bound being *max_bytes_needed*.
3	The request is satisfied from either the result of CMSSTOR OBTAIN or whatever was available in the cache, whichever is larger.

- When ssMemoryRelease is called, the released storage is added to the appropriate subpool cache, and if the free storage in the cache is above the maximum free amount specified by the MEM&_MAXFREE configuration parameter, the cache is trimmed.
- When ssMemoryDelete is called, the cache for the named subpool is destroyed, all storage being released through SUBPOOL DELETE.

The application should not call SUBPOOL DELETE for subpools that have been manipulated through calls to ssMemoryAllocate and ssMemoryRelease; such an invocation will confuse the reusable server kernel. Use ssMemoryDelete instead.

After the application ends, the reusable server kernel issues ssMemoryDelete for each subpool cache remaining.

For more information on the forms of the subpool names used internally by the reusable server kernel, see Appendix F, "Reserved Names," on page 375.

Chapter 10. Worker Machines

In some server situations, a single virtual machine performing complex operations for lots of clients simultaneously is an inconvenient, risky, or unachievable proposition. For example, if the clients are submitting code for the server to run as the clients' proxy, it would be desirable for each such client submission to run in an environment where it cannot tamper with, harm, or even innocently interfere with the execution of other clients' similar submissions. Similarly, if the server must run code that is under test or is at risk for terminating abnormally, the server designer should have at his disposal a means for running such code in isolation. In some cases, performance of the server might even improve if client work could be distributed among a set of worker virtual machines, each such worker performing a dedicated function for multiple clients simultaneously or perhaps working alone on behalf of a single client. These are no doubt only a few of the possible scenarios where the ability to run some of the server's work in other virtual machines would be an attractive feature.

The reusable server kernel recognizes these situations and offers an API that lets the server author distribute work among sets of subordinate virtual machines. These subordinates, called *workers*, usually run on the same CP instance as the main server. Sets of subordinates are defined to the main server via operator commands, probably in PROFILE RSK. The server kernel establishes communication connections to workers in response to API calls made by service instances; however, the format and meaning of the data actually exchanged with workers is left to the server author. In addition, when the workers are running on the same CP instance as the main server, the server kernel uses the XAUTOLOG and FORCE commands to log on and log off workers as appropriate. Finally, it should be emphasized that the relationship with the worker machine is mediated entirely by the service instance. The server kernel never shunts data directly from a client to a worker or vice-versa.

Functional Overview

For organizational purposes, the server kernel organizes worker machines into groups called *classes*. The virtual machines making up a class are all functionally equivalent to one another as far as the server author is concerned. In other words, when a service instance needs help from a worker, any member of the class will do; the server author leaves it up to the server kernel to select a class member and establish a connection to it. The server kernel is able to manage multiple worker classes simultaneously.

To initiate a connection to a worker, a service instance calls entry point ssWorkerAllocate, specifying the class from which the server kernel is to select a worker machine and specifying some details about how the connection is to be allocated. In response to this call, the reusable server kernel evaluates the load on each worker in the class, selects the least-loaded member, and attempts to establish an IUCV connection to it. The service instance can influence the selection algorithm slightly; it can specify either that the server kernel should XAUTOLOG another worker only if all currently logged-on workers are full, or it can specify that the server kernel should route the new connection to an empty or newly-autologged worker if possible, resorting to multiple connections to a single worker only if the class is sufficiently active. When ssWorkerAllocate returns to its caller, either the connection to the worker is in place or all reasonable attempts to contact a worker have been exhausted.

Each member of a worker class -- in other words, each worker virtual machine -- has associated with it a maximum number of IUCV connections it can handle simultaneously. The server author or server operator specifies this limit via operator command when he adds the worker to the class. For the purpose of worker machine selection, the load being imposed on a given worker is taken to be the fraction of its IUCV capacity in use. For example, a worker capable of handling four IUCV connections but handling only two at the moment is considered by the server kernel to be 50% utilized, while if that worker were handling only one IUCV connection at the moment, it would be considered to be 25% utilized. The load distribution algorithm selects the least-loaded machine, using round-robin to break ties.

If the caller requests it, the reusable server kernel can set alternate user ID and security label (seclabel) information for the worker as part of selecting the worker. To be able to set a worker's alternate user ID and seclabel, the controlling virtual machine must have permission to issue Diagnose X'D4'. See *z/VM: CP*

Programming Services for more information. If you attempt to use the reusable server kernel's alternate user ID machinery and your virtual machine does not have the privilege necessary to issue Diagnose X'D4', your virtual machine will take a program check. It is your responsibility to recover from this. Also note that the reusable server kernel always uses the subcode X'04' form of Diagnose X'D4'.

Once the connection to the worker is established, the service instance communicates with the worker using the `ssClient` APIs and CMS IPC, just as it would communicate with a client. More specifically, `ssWorkerAllocate` returns a C-block that represents the connection between the service instance and the worker. To write to the worker, the service instance uses `ssClientDataPut` followed by a CMS IPC message telling the server kernel that it has generated new data to be sent to the worker. Reading from the worker is similar; after it sees a CMS IPC message informing it that new data are available, the service instance calls `ssClientDataGet` to retrieve what the worker sent.

When a service instance is done using a worker, it notifies the reusable server kernel via CMS IPC, just as it would do to notify a server kernel line driver that it had finished with a client. The IPC message causes the server kernel to sever the IUCV connection to the worker. In the event that the worker terminates the connection first, the service instance is notified and must acknowledge the connection loss, just as it must respond to a line driver when it learns of the loss of communication to a client.

Server Configuration Considerations

The worker API uses IUCV to move data between the main server and the workers, and when the workers are running on the same CP instance as the main server, the worker API employs the CP XAUTOLOG and FORCE commands to start and stop worker machines. The following configuration considerations apply:

- The main server must be permitted to `IUCV CONNECT` to each worker machine. There are many ways to arrange this. Perhaps the simplest way is to insert `IUCV ALLOW` into the CP directory entry for each worker machine. Any method that lets the connection proceed is just fine.
- If the workers are running on the same CP instance as the main server, the main server virtual machine must be permitted to XAUTOLOG and FORCE worker machines. XAUTOLOG requires class A or B or an entry in the CP directory entry of each worker machine. FORCE requires CP class A.

Distributing Worker Machines

Some installations might choose to employ a single system image (SSI) cluster or the VM/Pass-Through Facility (PVM) to distribute IUCV and thereby run worker machines on systems other than the local CP. For example, specialized hardware might be available on some other processor, and a worker machine might be placed there to handle requests originating from other systems.

On a per-class basis, the server operator decides whether the server kernel is to manage workers as local or distributed. If the class is specified to be local, the server kernel employs XAUTOLOG and FORCE to log workers on and off as necessary. If the class is specified as distributed, the server kernel skips all such management steps, merely attempting `IUCV CONNECT` and returning an error if the connection attempt fails.

When a class is specified as distributed, the server operator or server designer is responsible for making sure that the worker machines are autologged at an appropriate time and that they are reset if errors or abends occur. A system management tool such as IBM Operations Manager for z/VM can be used for this purpose.

When the server kernel issues `IUCV CONNECT` to connect to a worker machine, it does so in a manner that can be distributed to other systems if CP is appropriately configured. To make this work in a non-SSI environment, the system administrator must specify `DISTRIBUTE IUCV YES` in the CP system configuration file (SYSTEM CONFIG). He must also make sure that the IUCV carrier (for example, PVM) is working properly. Within an SSI cluster, IUCV is automatically available among the member systems, regardless of the `DISTRIBUTE IUCV` configuration. However, to connect to a worker machine on a system that is part of the same ISFC collection but is not a member of the same SSI cluster, `DISTRIBUTE IUCV YES` must be specified.

API Details

To allocate a connection to a worker machine, the service instance calls `ssWorkerAllocate`, passing it a few pieces of information:

- The address of its own C-block

- The worker class in which the connection should be allocated

- An indication of how the server kernel is to select a worker:

  - The instance can ask that the server kernel attempt to minimize the number of worker machines logged on, routing connections to logged-on, not-completely-full workers whenever possible, or

  - The instance can ask that the server kernel route connections to empty or not-yet-logged-on workers whenever possible, choosing partially-busy, already-logged-on workers only when necessary.

- An integer specifying the number of workers the server kernel should try before giving up and returning failure to the caller.

- Optional alternate user ID and seclabel information.

Subject to these parameters, the server kernel selects a worker machine and tries to establish a connection to it. If the attempt fails, the server kernel will retry a small number of times, and if the worker proves unreachable, the server kernel will record this fact (so it can skip the worker when it handles subsequent `ssWorkerAllocate` calls) and move to another worker. The server kernel will iterate in this way until either the caller's specified number of tries expires or the whole worker class proves unreachable. Normally the retry strategy is not a factor - the usual case will be that the worker will be waiting for work and will accept the server kernel's `IUCV CONNECT` request immediately.

When `ssWorkerAllocate` returns to the calling instance, it supplies two pieces of information that are crucial to the instance's being able to interact with the assigned worker:

- It supplies a three-byte unsigned binary integer that uniquely identifies the connection to the worker. This integer is called the *connection ID*. This integer is returned in an unsigned four-byte buffer, the uppermost byte of said buffer always being zero.

- It supplies the address of a C-block that represents the connection to the worker. This is called the *worker C-block*.

To detect activity on the worker connection, the instance issues `QueueReceiveBlock` against its **line driver queue**, just as it normally does. Recall that under normal circumstances, this API call completes when the instance's line driver sends a message to the instance, informing the instance that something significant has happened with respect to its client. When using the worker API, though, the instance needs to be aware that messages indicative of *worker* activity will *also* arrive on its line driver queue. The instance can detect that a received IPC message is indicative of worker activity by examining the *message type* field of the received IPC message. A message indicative of worker activity contains X'01' as the high-order byte of the message type; the lower three bytes of the type field are the 24-bit connection ID returned by `ssWorkerAllocate`. Thus the instance can wait for either client activity or worker activity with a single call to `QueueReceiveBlock`, and the arriving message will tell the instance whether it's the client or a worker that needs attention.

To exchange data with the worker, the instance calls the `ssClient` APIs just as usual, using the *ss_cli_iam_instance* qualifier. Data are moved between the instance and the worker in the same manner as they are moved between instance and client. When the instance must send an IPC message to the "worker line driver" -- for example, to inform the server kernel that it has used `ssClientDataPut` to queue data for transmission to the worker -- it forms the instance-to-line-driver message just as it would for any line driver interaction and then transmits the IPC message to the queue handle appearing in the *worker* C-block. The server kernel receives the message and operates on the worker connection accordingly.

The Worker C-Block

The worker C-block contains a few fields that will be of special interest to the service instance. These fields are:

- A queue handle that represents the queue to which the instance should transmit CMS IPC messages relevant to the connection to the worker.
- A line driver key that should be used as the key in any such transmitted messages.
- The *vc_userid* field of the worker C-block contains the user ID of the worker virtual machine.

Further, certain fields in the worker C-block are zero because they are irrelevant in the context of a connection to a worker machine. For example, a worker C-block does not contain a pointer to an S-block.

Operator Commands

The reusable server kernel supplies a service, WORKER, which lets the server operator manipulate worker classes. The commands are given in the following table.

Command	Description	Page
Table 27. WORKER Commands		
Command	**Description**	**Page**
WORKER ADD	Lets the operator add a worker machine to a worker class, specifying the number of IUCV connections the worker machine is capable of handling simultaneously. This command would usually be found in PROFILE RSK, though the operator is free to issue it while the server is running.	"WORKER ADD" on page 203
WORKER CLASSES	Displays the existing worker machine classes and some brief status information about each class.	"WORKER CLASSES" on page 204
WORKER DELCLASS	Deletes an entire worker class. Normally this just means that any instances connected to workers in the class would receive an IPC message asking them to stop their activity. The FORCE option will cause the server kernel to sever the IUCV connections, to inform the instances that communication to the workers has been lost, and to CP FORCE any workers running disconnected. When DELCLASS processing completes, the worker class is no longer available for use.	"WORKER DELCLASS" on page 205
WORKER DELETE	Operates on a single worker machine in a manner similar to DELCLASS.	"WORKER DELETE" on page 206
WORKER DISTRIBUTE	Informs the server kernel that a worker class should be managed as if its worker machines are distributed across systems.	"WORKER DISTRIBUTE" on page 207
WORKER MACHINES	Displays a table of status information about the machines in a given class.	"WORKER MACHINES" on page 208
WORKER RESET	Clears any persistent error information the server kernel may have remembered about worker machines. This restores the workers to usable status and is useful after manual intervention has resolved a problem with a given worker machine or class of worker machines.	"WORKER RESET" on page 210
WORKER STATUS	Displays a table of status information about each worker connection existing at the moment.	"WORKER STATUS" on page 211

Writing a Worker Machine Program

IBM does not supply a program to run in the worker machine. The server author must write this program, being aware of the following configuration and execution considerations:

- The worker machine's CP directory entry and profiles must be configured so that the worker machine will start itself completely if autologged. If the worker machine is running a CMS-based program, `IPL CMS PARM AUTOCR` is appropriate in the worker's CP directory entry and the worker's `PROFILE EXEC` should be rigged so that the worker program starts automatically. If the worker program is running under some other operating system, the other operating system's corresponding mechanisms should be employed.

- The server kernel will attempt to `IUCV CONNECT` to the worker machine, using RSKWORK as the first eight bytes of the user data area of its connection parameter list. If the worker program is CMS-based, this means that the worker program will need to issue `HNDIUCV SET` to identify an exit named RSKWORK. When the server kernel attempts to connect, the worker program's RSKWORK exit routine will be driven. The worker program should respond with `CMSIUCV ACCEPT`.

- The format and meaning of the data exchanged on the IUCV connection is up to the server author.

- Eventually it will be time to bring down the IUCV connection. The server kernel will `IUCV SEVER` if the service instance instructs it that the relationship between the instance and the worker is to be ended; in this case the worker program should respond with `IUCV SEVER`. If the worker machine is the one that decides when the connection is over, it should issue `IUCV SEVER` and the server kernel will respond with its own `IUCV SEVER`, reflecting the connection loss to the service instance.

- If the main server is configured such that it might route multiple IUCV connections to a worker simultaneously, the worker program should be prepared to handle multiple IUCV connections simultaneously.

- The worker program should not use `IUCV SEND,TYPE=2WAY`, `IUCV QUIESCE`, or `IUCV RESUME`. The server kernel is not prepared to handle these and will respond with `IUCV SEVER`.

Finally, it is interesting to note that the reusable server kernel itself could be used as the base for a program to be run in the worker machine. The server kernel's IUCV line driver is capable of being the recipient of IUCV activity generated by the server kernel's worker API.

Chapter 11. Run-Time Environment

To facilitate the writing of well-performing programs and to provide high-performance interprocedure linkage, the reusable server kernel implements its own procedure linkage convention. The reusable server kernel entry points themselves (for example, `ssSgpStart`) all expect to be driven using this convention, and routines provided by the server writer (for example, RSKMAIN, service entry points, thread entry points, and so on) are all driven by the reusable server kernel using this convention. This convention greatly reduces the need to call a storage management interface to allocate and release save areas and local variable storage. This keeps overhead down, letting procedure linkage happen without excessive SVCs or other calls. [23]

Associated with each thread is a chain of control blocks known as *dynamic storage area frames* or *stack frames*. Each stack frame is at least 4 KB in size. Contained in each frame is a *frame header* and one or more *dynamic storage areas* (DSAs). The anchor for this chain of DSA frames is held in a control block called the *run-time anchor block* (RAB). An example is shown in .

---

[23] The linkage resembles the linkage used among internal entry points in the CMS Application Multitasking kernel.

DSA Frame

```
       ┌──────────────────────────────────┐
    0  │        A (next DSA frame)         │
       ├──────────────────────────────────┤
    4  │      Size (bytes) of this frame   │
       ├──────────────────────────────────┤
    8  │    64-byte save area for use by   │
       │      frame overflow handler       │
       ├──────────────────────────────────┤
   72  │        First DSA in frame         │
       ├──────────────────────────────────┤
   ??  │         Next DSA in frame         │
       │      (etc. for several DSAs)      │
       ├──────────────────────────────────┤
   ??  │         Last DSA in frame         │
       ├──────────────────────────────────┤
   ??  │           Unused space            │
       └──────────────────────────────────┘
```

Dynamic Storage Area (DSA)

```
       ┌──────────────────────────────────┐
    0  │            Unused word            │
       ├──────────────────────────────────┤
    4  │        A (previous DSA)           │
       ├──────────────────────────────────┤
    8  │          A (next DSA)             │
       ├──────────────────────────────────┤
   12  │    60-byte save area for R14...R12│
       ├──────────────────────────────────┤
   72  │           Unused area             │
       ├──────────────────────────────────┤
   84  │  Next available byte in DSA frame │
       ├──────────────────────────────────┤
   88  │           Unused area             │
       ├──────────────────────────────────┤
  120  │   Procedure's automatic storage   │
       └──────────────────────────────────┘
```

Run-Time Anchor Block (RAB)

```
       ┌──────────────────────────────────┐
    0  │        A (first DSA frame)        │
       ├──────────────────────────────────┤
    4  │    A (first byte past frame end)  │
       ├──────────────────────────────────┤
    8  │     A (frame overflow handler)    │
       └──────────────────────────────────┘
```

Figure 4. Run-Time Environment Control Blocks

The register contents at procedure entry are described in .

Table 28. Register Contents at Procedure Entry	
Register	**Description**
R1	Pointer to an OS Type I parameter list. The entries in this list are addresses of the actual parameter values.
R12	Pointer to the RAB, organized as shown above.
R13	Pointer to a DSA, organized as shown above.
R14	Return address.
R15	Called procedure's entry address.

When a procedure is entered, it uses the save area pointed to by R13 in the usual OS fashion (STM R14,R12,12(R13)). It then computes the size of the DSA it needs (120 bytes plus amount of automatic storage needed) and compares that to the amount left in the frame; this comparison is done by adding the amount needed to the next available byte (NAB) in the caller's save area and comparing that to the

frame end field in the RAB. If there is enough space in the frame, the new DSA is built starting at the byte pointed to by the NAB field in the current DSA, and this new DSA is chained to the caller's DSA in the usual OS fashion. If not enough space is left, then the frame overflow handler is called to add a new frame to the end of the frame list (the frame overflow handler's address is in the RAB). The frame overflow handler is cognizant of the registers used during procedure entry and returns with the registers set such that the linkage processing can continue as if no overflow had occurred.

When a procedure exits, it unchains its save area, restores the caller's registers (including the caller's R13, which comes from the *previous DSA pointer* field in the exiting procedure's DSA), and returns to the caller through BR R14.

The reusable server kernel provides PL/X and assembler macros implementing these entry and exit conventions. For PL/X, the macros are invoked through the OPTIONS clause on the PROCEDURE statement. For assembler, the macros are invoked directly by the assembler programmer. *The assembler programmer must ensure that the amount of DSA storage he requests is an integral number of doublewords.* An example is shown in Figure 5 on page 61 and Figure 6 on page 62.

```
@PROCESS ENVIRONMENT(VM/ESAOS) OPT(MAX);

/* illustration of linkage convention */

sstest: procedure
(
 pl_epptr,                   /* A(eplist)  */
 pl_tpptr,                   /* A(tplist)  */
 pl_scptr                    /* A(scblock) */
)
options
(
 id                      /* generates identifier      */
 reentrant               /* no static data, please    */
 amode(31)               /* AMODE 31                  */
 rmode(any)              /* can live anywhere         */
 datareg(13)             /* R13 locates automatic storage */
 savearea(120)           /* size of fixed part of DSA   */
 stack('SSPRLG','SSEPIL')  /* entry and exit macros      */
);

/* note BYVALUE because the pointer values we want are */
/* in the array pointed to by R1 */
declare sstest entry
(
 pointer(31) byvalue,
 pointer(31) byvalue,
 pointer(31) byvalue
)
external as ('RSKMAIN');

declare
 pl_epptr    pointer(31),   /* pointer to eplist         */
 pl_tpptr    pointer(31),   /* pointer to tplist         */
 pl_scptr    pointer(31);   /* pointer to SCBLOCK        */

respecify (r12) restricted;  /* stay away from RAB pointer   */

/* body of procedure goes here */

end sstest;
```

Figure 5. PL/X Linkage

```
***
*
*  Illustration of linkage convention
*
***
*
* Procedure entry:
*RSKMAIN
  CSECT ,                       Declare CSECTRSKMAIN
  AMODE 31                      Establish AMODERSKMAIN
  RMODE ANY                     Establish RMODE
        STM   R14,R12,12(R13)        Save registers
        LR    R11,R15                R11 is base register
        USING RSKMAIN,R11            Establish addressability
        LA    R0,DSASIZE             R0 = size of DSA needed
        SSPRLG                       R1,R2 -> new DSA, R0 = new NAB
        LR    R15,R13                R15 -> caller DSA
        LR    R13,R2                 R13 -> my DSA
        ST    R15,4(,R13)            Write my backward pointer
        ST    R13,8(,R15)            Write caller's forward pointer
        LM    R15,R2,16(R15)         Restore R15-R2
*
* Your code goes in here… stay away from R11-R13.  R14
* and R15 can be used as needed for calls to other routines.
*
* Note that your automatic storage area (the storage you
* requested via R0 when you called SSPRLG) starts at offset
* X'78' into the save area returned by SSPRLG.
*
*
* Procedure exit (note RC is in R15):
*
        L     R13,4(,R13)            R13 -> caller's DSA
        LA    R0,DSASIZE             Size of DSA I used
        SSEPIL                       Release it
        L     R14,12(,R13)           Get return address
        LM    R0,R12,20(R13)         Restore rest of registers
        BR    R14                    Return to caller
*
* Other stuff
* Note DSASIZE is a multiple of 8 bytes!
DSASIZE EQU   200                    200-120 = 80 bytes of local vars
        REGEQU                       Register equates
*
        END
```

Figure 6. Assembler Linkage

Like all other routines, the server entry point RSKMAIN is driven using this linkage convention. The parameter list array passed to RSKMAIN through R1 is organized as described in Table 29 on page 62.

Table 29. Parameter List Array for RSKMAIN	
Offset	**Usage**
0	Pointer to the extended parameter list with which CMS invoked the module.
4	Pointer to the tokenized parameter list with which CMS invoked the module.
8	Pointer to the SCBLOCK for the module, if the module is a nucleus extension.

The reusable server kernel uses CMS Application Multitasking's support for custom language run-time environments to implement its convention for procedure linkage. BKWRTE MODULE is the language environment manager for the reusable server kernel and needs to be present in the file mode search order when the server module starts. CMS loads BKWRTE as a nucleus extension prior to giving control to the server module. BKWRTE must remain loaded as a nucleus extension for the life of the server program.

Chapter 12. Initialization and Profiles

This chapter describes the flow of control during server execution and describes how to set up `PROFILE RSK`. For descriptions of the various command sets, see Chapter 14, "Command Descriptions," on page 77.

To accomplish most of the work of initializing and configuring the server, the server author writes a Rexx exec, `PROFILE RSK`. In this exec the server author supplies commands necessary to configure the server, start it, and wait for its completion. The reusable server kernel runs `PROFILE RSK` as part of server startup.

Most of the work done in `PROFILE RSK` is accomplished through `ADDRESS RSK` and command sets implemented by the reusable server kernel. These command sets fall into a few broad categories:

- `CONFIG` commands, meant to set certain configuration parameters needed by the reusable server kernel during execution.
- `SGP` commands, meant to manipulate storage groups.
- `AUTH` commands, meant to provide a means for manipulating the authorization database.
- `CACHE` commands, meant to provide a means for configuring file caches.
- `ENROLL` commands, meant to manipulate enrollment data.
- `WORKER` commands, meant to define pools of worker machines.
- Line driver commands, meant to manipulate line drivers and the relationships between line drivers and services.

Flow of Control

The general flow of control during the execution of the server is illustrated in Figure 7 on page 64. The execution of the server has these general stages:

Step	Description
1	The module begins, and the reusable server kernel performs some rudimentary initialization.
2	The reusable server kernel passes control to RSKMAIN, the server entry point provided by the server author.
3	RSKMAIN performs whatever setup is needed, including binding its services through calls to `ssServiceBind`.
4	RSKMAIN calls `ssServerRun` to begin the server.

Step	Description
5	ssServerRun passes control to PROFILE RSK. The processing in PROFILE RSK proceeds in several stages, as follows: 1. The profile may perform appropriate initialization. 2. The profile issues several CONFIG commands to set configuration parameters for the reusable server kernel. 3. The profile issues the RUNSERV command to begin the execution of the server. In response to RUNSERV, the reusable server kernel brings up line drivers and makes APIs available for use. When RUNSERV returns, the reusable server kernel is ready for operation. 4. The profile issues any AUTH, CACHE, or other commands necessary to configure the server. 5. The profile issues one or more line driver START commands to start services. At this point the server is running. 6. The profile issues the WAITSERV command to wait for the server to end. 7. The profile may perform appropriate termination activities. 8. The profile returns to its caller.
6	ssServerRun returns to RSKMAIN. The return and reason code from ssServerRun indicate whether the server was able to be started.
7	RSKMAIN performs termination processing.
8	RSKMAIN returns to the reusable server kernel, supplying a return code.
9	The reusable server kernel performs termination and returns to CMS. The return code supplied to CMS by the server module is the return code of RSKMAIN.

```
CMS
   └───► Server module
           1. Init anchor block
           2. Establish subcom handler
           3. Establish CONFIG command
           4. Call RSKMAIN
                 a. Server-wide initialization
                 b. Bind services
                 c. Call ssServerRun
                      (1) Invoke PROFILE RSK
                             (a) Issue CONFIG command
                             (b) Issue  RUNSERV command
                                    ((1)) Load user ID mapping file
                                    ((2)) Init cradle subsystems
                                    ((3)) Return to PROFILE RSK
                             (c) Issue AUTH, SGP, CACHE commands
                             (d) Issue line driver START command
                             (e) Issue WAITSERV command
                      (2) Return to RSKMAIN
                 d. Server-wide termination
                 e. Return to caller
           5. Remove subcom handler
           6. Perform cleanup
           7. Return to CMS
```

Figure 7. Flow of Control

Execution Conditions within RSKMAIN

RSKMAIN has only two reusable server kernel APIs at its disposal:

- `ssServiceBind`, to bind services.
- `ssServerRun`, to start the server and wait for its completion.

No other APIs are permitted. Attempts to call them will produce unpredictable results.

PROFILE RSK

Shortly after the server module begins execution, `PROFILE RSK` gets control. This is just a Rexx exec that performs initialization, configures the server, starts it, waits for it to end, and then performs termination functions.

For the server author's convenience, any parameters present on the command line used to invoke the server module are passed to `PROFILE RSK` such that they can be retrieved with `parse arg`.

In general, anything one can do from Rexx is permitted in `PROFILE RSK`. However, here are some things to keep in mind:

- Some `CONFIG` commands are usable only before RUNSERV while others are usable anytime. For more information, see Table 31 on page 66.
- All of the rest of the commands sets (for example, AUTH) are usable only between RUNSERV and WAITSERV, that is, only while the server is running. Attempts to use these commands at other times produce RC=-3.

For a sample of `PROFILE RSK`, see Appendix A, "Sample PROFILE RSK," on page 361.

Starting and Stopping

Table 30 on page 65 illustrates the syntax for the RUNSERV and WAITSERV commands. Issue these from Rexx using `ADDRESS RSK`.

Table 30. RUNSERV and WAITSERV Commands			
Command	**Usage**	**Syntax**	**Notes**
RUNSERV	Used within `PROFILE RSK` to start the server.	▶▶─ RUNSERV ─▶◀	Return codes: **0** Server started OK **x** Some other situation
WAITSERV	Used within `PROFILE RSK` to wait for the server to stop.	▶▶─ WAITSERV ─▶◀	Return codes: **0** Server terminated normally **x** Some other situation

Configuration Parameters

The reusable server kernel defines certain *configuration parameters* so that the server author or system programmer can control the manner in which the server behaves. These configuration parameters are manipulated by a command, `CONFIG`, which is useful in `PROFILE RSK`. `CONFIG` is issued through `ADDRESS RSK`. Most `CONFIG` commands are useful only prior to issuing RUNSERV, but some are useful anytime.

The parameters and their meanings are given in Table 31 on page 66. For definitions of the commands used to manipulate these parameters, see Chapter 14, "Command Descriptions," on page 77.

In truth, CONFIG is a *service* meant for the manipulation of configuration variables. This means that a command such as MSG START CONFIG could be used to permit remote manipulation of configuration variables.

Table 31. Configuration Variables			
Variable	**Function**	**When?**	**Notes**
AUT_CACHE	Sets the number of rows of authorization data to cache.	Anytime.	Specify *rows* as a positive integer.
AUT_DATA_1	Sets the name of copy 1 of the authorization data file.	Pre-RUNSERV	
AUT_DATA_2	Sets the name of copy 2 of the authorization data file.	Pre-RUNSERV	Ignored when AUT_LOCATION is SFS.
AUT_FREE	Sets the maximum number of row buffers to keep on the free row buffer list.	Anytime.	Specify *rows* as a positive integer.
AUT_INDEX_1	Sets the name of copy 1 of the authorization index file.	Pre-RUNSERV	
AUT_INDEX_2	Sets the name of copy 2 of the authorization index file.	Pre-RUNSERV	Ignored when AUT_LOCATION is SFS.
AUT_LOCATION	Sets the repository for the authorization data.	Pre-RUNSERV	
AUT_LOG	Sets the name of the authorization logfile.	Pre-RUNSERV	Ignored when AUT_LOCATION is SFS.
AUTHCHECK_AUTH	Sets whether the AUTH service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_CACHE	Sets whether the CACHE service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_CMS	Sets whether the CMS service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_CONFIG	Sets whether the CONFIG service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_CP	Sets whether the CP service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_ENROLL	Sets whether the ENROLL service will perform authorization checking for its commands.	Anytime.	

Variable	Function	When?	Notes
AUTHCHECK_LD	Sets whether line drivers will perform authorization checking for START or STOP commands.	Anytime.	
AUTHCHECK_MONITOR	Sets whether the MONITOR service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_SERVER	Sets whether the SERVER service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_SGP	Sets whether the SGP service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_TRIE	Sets whether the TRIE service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_USERID	Sets whether the USERID service will perform authorization checking for its commands.	Anytime.	
AUTHCHECK_WORKER	Sets whether the WORKER service will perform authorization checking for its commands.	Anytime.	
MEM_MAXFREE	Sets the maximum number of pages that should be kept preallocated by the reusable server kernel storage manager for any one subpool.	Anytime.	Specify *pages* as a positive integer.
MON_PRODUCT_ID	Sets the 16-byte product identifier the reusable server kernel will use when it invokes DIAG X'00DC' to identify the server's APPLDATA monitor buffer.	Pre-RUNSERV	
MON_USER_SIZE	Sets the size of the application monitor buffer.	Pre-RUNSERV	The address of the application monitor buffer is returned by ssAnchorGet.
MON_KERNEL_ROWS	Sets the number of rows the kernel monitor buffer will contain.	Pre-RUNSERV	
MSG_NOHDR	Sets whether the MSG/SMSG line driver will use CP's MSGNOH command to issue replies.	Anytime.	
NOMAP_APPC	Sets whether the APPC line driver will pass an unmappable user ID to an instance.	Anytime.	
NOMAP_IUCV	Sets whether the IUCV line driver will pass an unmappable user ID to an instance.	Anytime.	

Table 31. Configuration Variables (continued)

Table 31. Configuration Variables (continued)

Variable	Function	When?	Notes
NOMAP_MSG	Sets whether the MSG/SMSG line driver will pass an unmappable user ID to an instance.	Anytime.	
NOMAP_TCP	Sets whether the TCP line driver will pass an unmappable user ID to an instance.	Anytime.	
NOMAP_UDP	Sets whether the UDP line driver will pass an unmappable user ID to an instance.	Anytime.	
NOMAP_SPOOL	Sets whether the SPOOL line driver will pass an unmappable user ID to an instance.	Anytime.	
RSCS_USERID	Sets the user ID of the RSCS machine the SPOOL driver should use.	Anytime.	
SGP_FILE	Sets the name of the storage group definition file.	Pre-RUNSERV	
SPL_CATCHER	Sets the user ID to which the SPOOL line driver will CP TRANSFER spool files it is unable to decode.	Anytime.	
SPL_INPUT_FT	Sets the file type of reader files the SPOOL line driver will recognize as service input.	Anytime.	
SPL_OUTPUT_FT	Sets the file type of punch files the SPOOL line driver will generate in response to service output.	Anytime.	
SRV_THREADS	Sets the maximum number of threads of a service a parallelizing line driver will attempt to run simultaneously.	Anytime.	
UMAP_FILE	Sets the name of the user ID mapping file.	Anytime.	
VM_CONSOLE	Sets whether the console line driver will pass unrecognized command lines to CMS for execution.	Anytime.	
VM_MSG	Sets whether the MSG/SMSG line driver will pass unrecognized messages to CMS for execution.	Anytime.	
VM_SUBCOM	Sets whether the SUBCOM line driver will pass unrecognized messages to CMS for execution.	Anytime.	
VM_SPOOL	Sets whether the SPOOL line driver will pass unrecognized input to CMS for execution.	Anytime.	

Storage Group Definition File

The storage groups known to the reusable server kernel are recorded in the file whose name is given in configuration variable SGP_FILE. Each time an API call that changes the storage group configuration executes successfully, the reusable server kernel rewrites the file. Thus storage group definitions persist across invocations of the server program.

This file is not meant for manual manipulation. It should be manipulated only with the appropriate API calls or administration commands.

This file must be present when the reusable server kernel starts. If it is not present, the reusable server kernel will not start. To create the first-ever configuration file, just use XEDIT to make a one-record, V-format file whose only record contains an asterisk as its first character. The reusable server kernel will ignore this record and realize that no storage groups are defined.

User ID Mapping Facility

Frequently the reusable server kernel translates *(nodeid,userid)* pairs to single-token user IDs. This mapping is part of the scheme by which the reusable server kernel presents single-token user IDs to service instances. For example, the spool file line driver translates the origin node and origin user ID of a request file into a single-token user ID and passes that single-token user ID to a service instance. Similarly, the TCP/IP line driver translates the client's IP address into a single-token user ID. [24] Both these translations are done through a translation database called the *user ID mapping file*. The user ID mapping data is kept in a file whose name is given in configuration variable UMAP_FILE.

The reusable server kernel loads the mapping file into storage when the server starts and uses the in-storage copy for translations. The command USERID RELOAD is available for reloading the in-storage copy from disk. This lets the server operator change the mapping while the server is running.

Each time the reusable server kernel needs to translate a *(userid,nodeid)* pair to a single-token user ID, the translation is done according to the rules in the mapping file. The translation scan goes from top to bottom through the file, stopping at the first matching entry. The entries can contain wildcards to ease the handling of groups of users (nodes, and so forth). The rules for wildcard use are the same as the rules for wildcards in CMS Application Multitasking's IPC message keys and event keys.

The syntax rules for the user ID mapping file are illustrated in Appendix B, "Sample User ID Mapping File," on page 365 contains a sample user ID mapping file.

The mapping file must be present when the server starts; the server will not start without it.

---

[24] For TCP/IP, *nodeid* is the IP address, and *userid* is ∗.

Chapter 13. Monitor Data

While the server runs, the reusable server kernel uses CP's APPLDATA facility (Diagnose X'00DC') to accrue monitor data. The monitor data support is arranged so that both the reusable server kernel itself and the server application can generate monitor data concurrently.

The monitor data facility works like this:

- As part of setting up the server virtual machine's CP directory entry, the system administrator must insert OPTION APPLMON so that the server virtual machine will be permitted to produce monitor data.

- In PROFILE RSK prior to the RUNSERV command, the server author places CONFIG commands to set the values of the MON_PRODUCT_ID and MON_KERNEL_ROWS configuration variables. These variables control the following things:

  - The value of MON_PRODUCT_ID is the product ID the reusable server kernel uses when it invokes Diagnose X'00DC' to identify each monitor buffer.

  - The value of MON_KERNEL_ROWS is the number of monitor rows the server kernel should allocate for its own purposes. The minimum and default value is 36 rows.

- Just after RUNSERV, the server kernel allocates one or more monitor buffers according to the configuration parameters specified, using Diagnose X'00DC' to identify each monitor buffer. If an error occurs in trying to identify a monitor buffer, the server kernel will write a message to the server console, specifying the Diagnose X'00DC' return code produced by CP. The server administrator will need to interpret the return code and take appropriate action.

- While the server runs, the server kernel employs rows of the monitor buffer to log information pertinent to the use of various resources (memory subpools, for example). Monitor data is produced for a resource for only as long as the resource exists; when the resource is deleted, the monitor row is marked free and might be reused later for some other resource.

- If the server application wants to produce its own monitor data, it can call entry point ssAnchorGet to retrieve the address and length of the portion of the monitor buffer reserved for application use.

- The application can store information into the application portion of the monitor buffer, and the values stored in the buffer will be picked up by CP as APPLDATA.

- As part of server shutdown, the server kernel invokes Diagnose X'00DC' again to retract the monitor buffers.

Monitor Buffer Organization

The first part of each monitor buffer is reserved for use by the server kernel. This reserved portion is organized into records called *monitor rows*. The first eight bytes of each row tell the kind of data accruing in that row, according to .

Table 32. Monitor Data Rows

Identifier	Type of Row
KERNEL	Kernel information
SERVICE	Service information
LINEDRV	Line driver information
AUTH	Authorization information
SGP	Storage group information
MEM	Memory information
ENROLL	Enrollment information

Table 32. Monitor Data Rows (continued)

Identifier	Type of Row
CACHE	File cache row
TRIE	Trie API row
WORKER	Worker API row
$UNUSED	Unused row

After the area used by the server kernel comes the application portion of the monitor buffer. The application can use `ssAnchorGet` to retrieve the address and length of this area.

The sections below describe the organizations of the server kernel's monitor buffer rows.

Kernel Row

The kernel row gives basic information about the organization of the monitor area. There is only one kernel row per monitor buffer. In each monitor buffer, the kernel row is the first row in the buffer.

Table 33. KERNEL Monitor Row

Offset	Length	Data Type	Usage
0	8	CHAR	String "KERNEL"
8	8	CHAR	Blanks (X'40')
16	4	INT	Number of rows
20	4	INT	Size of row (bytes)
24	4	INT	Size of application portion
28	4	INT	Reserved for IBM

Service Row

A service row accumulates information about the operation of a specific service.

Table 34. SERVICE Monitor Row

Offset	Length	Data Type	Usage
0	8	CHAR	String "SERVICE"
8	8	CHAR	Service name
16	4	INT	Reserved for IBM
20	4	INT	Number of completed transactions
24	8	INT	Total bytes from clients
32	8	INT	Total bytes to clients

Line Driver Row

A line driver row accumulates information about the operation of a specific line driver.

Table 35. LINEDRV Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "LINEDRV"
8	8	CHAR	Service name
16	4	INT	Reserved for IBM
20	4	INT	Number of completed transactions
24	8	INT	Total bytes from clients
32	8	INT	Total bytes to clients

Authorization Row

The authorization row accumulates information about the operation of the authorization API.

Table 36. AUTH Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "AUTH"
8	8	CHAR	Unused
16	4	INT	Number of permits
20	4	INT	Number of inquiries
24	4	INT	Number of rows retrieved
28	4	INT	Number of row cache hits

Storage Group Row

A storage group row accumulates information about the operation of a particular storage group.

Note that times are accrued only when I/O is performed through DIAG X'00A4'.

Table 37. SGP Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "SGP"
8	8	CHAR	Storage group name
16	4	INT	Reserved for IBM

20	4	INT	I/O technique:
			
			0
			    Diag X'A4'
			1
			    Diag X'0250'
			2
			    VM Data Spaces

24	4	INT	Number of reads
28	8	INT	Pages read
36	8	INT	Time spent reading (STCK)

Table 37. SGP Monitor Row (continued)			
Offset	**Length**	**Data Type**	**Usage**
44	4	INT	Number of writes
48	8	INT	Pages written
56	8	INT	Time spent writing (STCK)

Memory Row

A memory row accumulates information about the operation of a particular subpool.

Table 38. MEM Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "MEM"
8	8	CHAR	Subpool name
16	4	INT	Free storage in server kernel cache
20	4	INT	Amount currently in use through ssMemoryAllocate
24	4	INT	Calls to ssMemoryAllocate
28	8	INT	Total taken through ssMemoryAllocate
36	4	INT	Calls to ssMemoryRelease
40	8	INT	Total returned through ssMemoryRelease
48	4	INT	Times extended through CMSSTOR
52	8	INT	Total taken through CMSSTOR
60	4	INT	Times depleted through CMSSTOR
64	8	INT	Total returned through CMSSTOR

Enrollment Row

An enrollment row accumulates information about the operation of a particular enrollment set.

Table 39. ENROLL Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "ENROLL"
8	8	CHAR	Enrollment set name
16	4	INT	Number of records in set
20	4	INT	Bytes in use holding records
24	4	INT	Count of insertions
28	4	INT	Count of removals
32	4	INT	Count of retrievals

Cache Row

The cache row accumulates information about the operation of the file caching API.

Table 40. CACHE Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "CACHE"
8	8	CHAR	Cache name
16	4	INT	Cache size in bytes
20	4	INT	Bytes in use
24	4	INT	Files in cache
28	4	INT	Number of opens
32	4	INT	Number of hits
36	4	INT	Number of discards

Trie Row

The trie row accumulates information about the operation of the trie API.

Table 41. TRIE Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "TRIE"
8	8	CHAR	Trie name
16	4	INT	Last free trie byte
20	4	INT	Next free trie byte
24	4	INT	Records indexed
28	4	INT	Internal node count
32	4	INT	Number of lookups done
36	4	INT	Number of records returned

A trie's monitor data is maintained only in the virtual machine that owns the trie and is updated only when the owning virtual machine performs an operation against the trie.

Worker Row

The worker row accumulates information about the operation of the worker machine API.

The worker row is updated every 30 seconds as long as there is activity through the worker API (if no calls to the worker API happen, the row does not get updated). The worker row contains information about the three most active worker classes, as measured by total number of worker connections since the server started. The information in the worker row can be trusted if the STCK field of the row is nonzero. While the row is being recomputed, the STCK field is set to zero. There is no guarantee that the classes will be mentioned in the row in order of their activity - the most active class might appear in the "class 3" slot, for example.

Table 42. WORKER Monitor Row			
Offset	**Length**	**Data Type**	**Usage**
0	8	CHAR	String "WORKER"
8	8	CHAR	Unused

Offset	Length	Data Type	Usage
16	8	DWORD	STCK of last monitor row update
24	8	CHAR	Class name 1
32	4	INT	Total connections to class 1
36	4	INT	Connections right now to class 1
40	8	CHAR	Class name 2
48	4	INT	Total connections to class 2
52	4	INT	Connections right now to class 2
56	8	CHAR	Class name 3
64	4	INT	Total connections to class 3
68	4	INT	Connections right now to class 3

Table 42. WORKER Monitor Row (continued)

Chapter 14. Command Descriptions

NOT-PI

This chapter contains information that is NOT intended to be used as Programming Interfaces of z/VM.

This chapter describes commands made available by the set of services shipped as part of the reusable server kernel:

Table 43. Command Subsets

Subset	Description
APPC	Provides a means of controlling the APPC/VM line driver.
AUTH	Provides a means of manipulating the authorization database.
CACHE	Provides a means of manipulating file caches.
CMS	Provides a means of issuing CMS commands.
CONFIG	Provides a means of manipulating configuration parameters.
CONSOLE	Provides a means of manipulating the console line driver.
CP	Provides a means of issuing CP commands.
ENROLL	Provides a means of manipulating enrollment data.
IUCV	Provides a means of manipulating the IUCV line driver.
MONITOR	Provides a means of displaying monitor rows.
MSG	Provides a means of manipulating the MSG/SMSG line driver.
SERVER	Provides a means of controlling the execution of the server.
SGP	Provides a means of manipulating storage groups.
SPOOL	Provides a means of manipulating the SPOOL line driver.
SUBCOM	Provides a means of manipulating the SUBCOM line driver.
TCP	Provides a means of manipulating the TCP/IP line driver.
TRIE	Provides a means of manipulating tries.
UDP	Provides a means of manipulating the UDP/IP line driver.
USERID	Provides a means of manipulating the user ID mapping file.
WORKER	Provides a means of manipulating worker machine pools.

In truth, each of these command sets is implemented as a reusable server kernel service of the same name. Said services all expect record-oriented input and they all produce record-oriented output. This means that they can be sourced by any of the reusable server kernel's record-oriented line drivers. In addition, these services can be sourced by the bulk data line drivers if the client program takes responsibility for managing the data stream in record-oriented fashion (see Table 8 on page 12).

To set up the particular sourcing arrangement you want, use PROFILE RSK. For an example of a PROFILE RSK that establishes several sourcing arrangements for each of these services, see Appendix A, "Sample PROFILE RSK," on page 361.

In addition to the specific messages listed in the command descriptions that follow, any of these commands might produce any of these messages:

BKW0000I Operation completed OK.
BKW0001E Not authorized.
BKW0002E Enter a command.
BKW0003E Syntax error.
BKW0004E Unrecognized command.

For more information about messages, see Appendix H, "Messages," on page 393.

Syntax, Message, and Response Conventions

The following topics provide information on the conventions used in syntax diagrams and in examples of messages and responses.

How to Read Syntax Diagrams

Special diagrams (often called *railroad tracks*) are used to show the syntax of external interfaces.

To read a syntax diagram, follow the path of the line. Read from left to right and top to bottom.

- The ►►── symbol indicates the beginning of the syntax diagram.
- The ──► symbol, at the end of a line, indicates that the syntax diagram is continued on the next line.
- The ►── symbol, at the beginning of a line, indicates that the syntax diagram is continued from the previous line.
- The ──►◄ symbol indicates the end of the syntax diagram.

Within the syntax diagram, items on the line are required, items below the line are optional, and items above the line are defaults. See the examples in Table 44 on page 78.

Table 44. Examples of Syntax Diagram Conventions	
Syntax Diagram Convention	**Example**
Keywords and Constants A keyword or constant appears in uppercase letters. In this example, you must specify the item KEYWORD as shown. In most cases, you can specify a keyword or constant in uppercase letters, lowercase letters, or any combination. However, some applications may have additional conventions for using all-uppercase or all-lowercase.	►►─ KEYWORD ─►◄
Abbreviations Uppercase letters denote the shortest acceptable abbreviation of an item, and lowercase letters denote the part that can be omitted. If an item appears entirely in uppercase letters, it cannot be abbreviated. In this example, you can specify KEYWO, KEYWOR, or KEYWORD.	►►─ KEYWOrd ─►◄

Table 44. Examples of Syntax Diagram Conventions (continued)
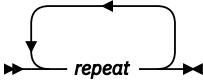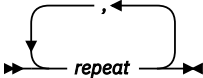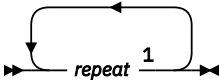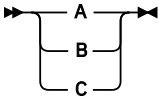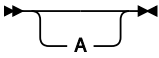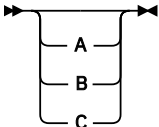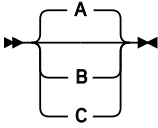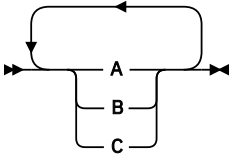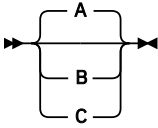
Syntax Diagram Convention	Example
Symbols You must specify these symbols exactly as they appear in the syntax diagram.	***** Asterisk **:** Colon **,** Comma **=** Equal Sign **-** Hyphen **()** Parentheses **.** Period
Variables A variable appears in highlighted lowercase, usually italics. In this example, *var_name* represents a variable that you must specify following KEYWORD.	►►— KEYWOrd —— *var_name* —►◄
Repetitions An arrow returning to the left means that the item can be repeated. A character within the arrow means that you must separate each repetition of the item with that character. A number (1) by the arrow references a syntax note at the bottom of the diagram. The syntax note tells you how many times the item can be repeated. Syntax notes may also be used to explain other special aspects of the syntax.	 Notes: [1] Specify *repeat* up to 5 times.
Required Item or Choice When an item is on the line, it is required. In this example, you must specify A. When two or more items are in a stack and one of them is on the line, you must specify one item. In this example, you must choose A, B, or C.	
Optional Item or Choice When an item is below the line, it is optional. In this example, you can choose A or nothing at all. When two or more items are in a stack below the line, all of them are optional. In this example, you can choose A, B, C, or nothing at all.	

Table 44. Examples of Syntax Diagram Conventions (continued)

Syntax Diagram Convention	Example
Defaults When an item is above the line, it is the default. The system will use the default unless you override it. You can override the default by specifying an option from the stack below the line. In this example, A is the default. You can override A by choosing B or C.	
Repeatable Choice A stack of items followed by an arrow returning to the left means that you can select more than one item or, in some cases, repeat a single item. In this example, you can choose any combination of A, B, or C.	
Syntax Fragment Some diagrams, because of their length, must fragment the syntax. The fragment name appears between vertical bars in the diagram. The expanded fragment appears in the diagram after a heading with the same fragment name. In this example, the fragment is named "A Fragment."	 **A Fragment**

Examples of Messages and Responses

Although most examples of messages and responses are shown exactly as they would appear, some content might depend on the specific situation. The following notation is used to show variable, optional, or alternative content:

xxx
: Highlighted text (usually italics) indicates a variable that represents the data that will be displayed.

[ ]
: Brackets enclose optional text that might be displayed.

{ }
: Braces enclose alternative versions of text, one of which will be displayed.

|
: The vertical bar separates items within brackets or braces.

...
: The ellipsis indicates that the preceding item might be repeated. A vertical ellipsis indicates that the preceding line, or a variation of that line, might be repeated.

APPC LIST

```
▶▶ APPC ── LIST ▶◀
```

Purpose

Lists the subtasks associated with the APPC/VM line driver.

Operands

None

Options

None

Usage Notes

The output form is:

```
Subtask ServName T ExitName Capacity InUse Threads Waiters
------- -------- - -------- -------- ----- ------- -------
0       ECHO     G BKWG0000 40       0     1       0
```

The columns have the following meanings:

Subtask
The numeric identifier of the subtask.

ServName
The name of the service involved.

T
The type of APPC/VM resource, as follows:

G
APPC/VM global resource

L
APPC/VM local resource

P
APPC/VM private resource

ExitName
The name of the CMSIUCV exit the server kernel opened. Also known as the *transaction program name*.

Capacity
The number of concurrent clients the subtask can handle.

InUse
The number of clients currently being handled.

Threads
The number of CMS threads working on behalf of this subtask.

Waiters
The number of clients whose conversations are waiting to be accepted (unhandled connection pending interrupts).

Messages and Return Codes

BKW0201E Subtask not found.

APPC QUERY

```
►► APPC ── QUERY ── subtaskid ►◄
```

Purpose

Queries a specific APPC/VM subtask.

Operands

subtaskid
    The identifier of the subtask to query.

Options

None

Usage Notes

The output form is:

```
Instance C-Block  Userid   IPVMID  LUName            BytesIn   BytesOut
-------- -------  ------   ------  ------            -------   --------
1        01AFD1B8 BKW      WADEB   *USERID:WADEB     0         0
```

The columns have the following meanings:

Instance
    The numeric identifier of the instance.

C-Block
    The address of the instance's C-block.

Userid
    The mapped user ID of the client.

IPVMID
    The security user ID of the client.

LUName
    The name of the LU at which the client resides.

BytesIn
    The number of bytes the client has sent the instance.

BytesOut
    The number of bytes the instance has sent the client.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

APPC REPORT

```
►►─── APPC ──── REPORT ──┬── ON ──┬──►◄
                          └── OFF ─┘
```
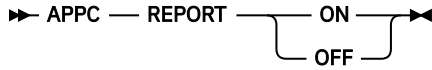
Purpose

Toggles reporting state for the APPC line driver.

Operands

ON
　　Turns reporting on.

OFF
　　Turns reporting off.

Options

None

Usage Notes

When reporting is on, the APPC line driver issues the following messages to describe client activity:

• BKW1704I

• BKW1705I

• BKW1706I

• BKW1707I

For more information, see .

Messages and Return Codes

None

APPC START

```
►►── APPC ── START ── servicename ──┬──────────── GLOBAL ── srv_threads ── servicename ───────┬──►◄
                                    │        ┌── GLOBAL ──┐                                    │
                                    └────────┼── LOCAL ───┼──┬──────────┬──────────────────────┘
                                             └── PRIVATE ─┘  └ maxclients ┴──┬───────┬──┘
                                                                            └── tpn ─┘
```
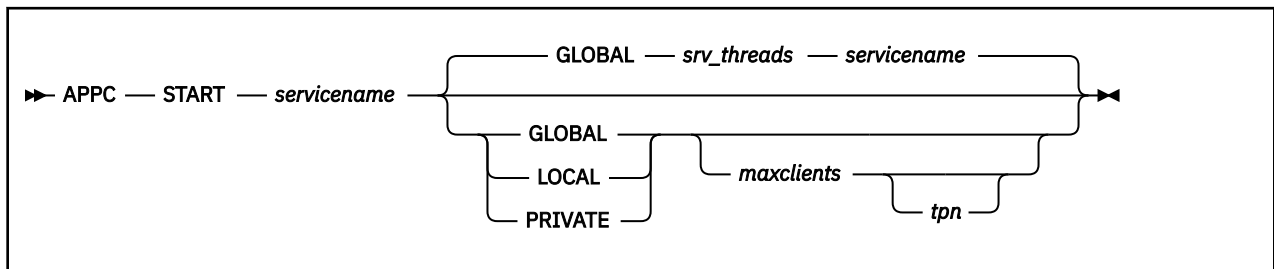
Purpose

Starts a service, connecting it to the APPC/VM line driver.

Operands

servicename
    The name of the service to start, as specified on a call to `ssServiceBind`.

GLOBAL
    The transaction program should be registered as an APPC/VM global resource.

LOCAL
    The transaction program should be registered as an APPC/VM local resource.

PRIVATE
    The transaction program should be registered as an APPC/VM private resource.

maxclients
    The maximum number of clients this subtask should be permitted to serve concurrently.

tpn
    The transaction program name the APPC/VM line driver should use.

Options

GLOBAL
    The transaction program should be registered as an APPC/VM global resource.

srv_threads
    The current value of configuration parameter SRV_THREADS.

servicename
    The name of the service being started.

Usage Notes

1. To register a global or local resource, the server virtual machine's CP directory entry must be appropriately configured.

2. To register a private resource, $SERVER$ NAMES must be set up correctly.

3. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

    For more information, see *z/VM: Connectivity*.

Messages and Return Codes

BKW0005E Out of storage.
BKW0200E Service not found.
BKW0205E Prefix already in use.

BKW0206E Service INIT routine failed - RC=&1 RE=&2.
BKW0207E Start of self is prohibited.
BKW1607E Client count must be greater than zero.
BKW1608E Unable to HNDIUCV SET.
BKW1609E Unable to create controlling thread.
BKW1700E (Resource &1) CMSIUCV CONNECT to *IDENT RC=&2
BKW1702E Unable to identify APPC/VM resource.

APPC STOP

```
►►── APPC ──── STOP ──── subtaskid ──────────────────►◄
                                └──── NOW ────┘
```
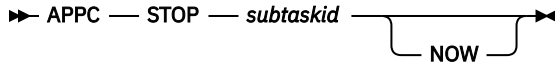
Purpose

Stops a specific APPC/VM subtask, optionally denying currently-connected clients the privilege of completing their operations.

Operands

subtaskid
    The identifier of the subtask to stop.

Options

NOW
    Stop the subtask without letting current clients complete normally.

Usage Notes

None

Messages and Return Codes

BKW0201E Subtask not found.
BKW1600I Instance STOP requested.
BKW1606E Wait expired for STOP.

AUTH CRECLASS

```
▶▶── AUTH ── CRECLASS ── class ──┬── operation ──┬──▶◀
                                 └◀──────────────┘
```

Purpose

Creates an object class in the authorization database.

Operands

class
   The name of the class to be created.

operation
   The name of an operation to be defined on objects of this class.

Options

None

Usage Notes

For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

BKW0005E Out of storage.
BKW0007E RC=&1 RE=&2 from routine &3
BKW0800E The class specified already exists
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files

AUTH CREOBJECT

▶▶── AUTH ── CREOBJECT ── *object* ── *class* ──▶◀

Purpose

Creates an object class in the authorization database.

Operands

object
> The name of the object to be created.

class
> The name of the class to which the object is to belong.

Options

None

Usage Notes

For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

BKW0005E Out of storage.
BKW0007E RC=&1 RE=&2 from routine &3
BKW0800E The class specified already exists
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0805E The class specified does not exist
BKW0806E The object specified already exists

AUTH DELCLASS

```
►►── AUTH ── DELCLASS ── class ──┬──────────────────────────┬──►◄
                                 └─( ──┬──────────────┬──────┘
                                       └─ OBJECTSONLY ─┘
```
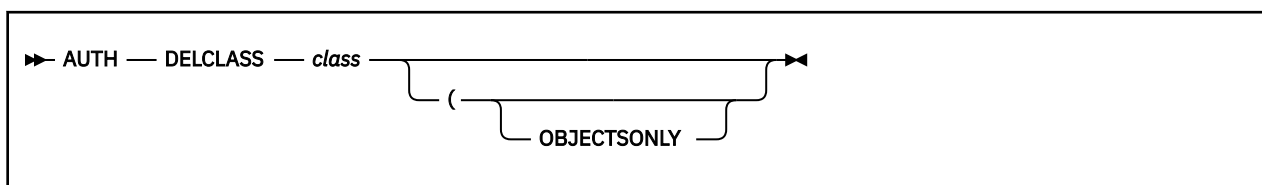
Purpose

Deletes the objects of a given class.

Operands

class
>   The class for which objects are to be deleted.

Options

OBJECTSONLY
>   Delete the objects for the class, but leave the class itself in the authorization database.

Usage Notes

1. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

2. If OBJECTSONLY is omitted, then the class itself is also deleted from the authorization database.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0805E The class specified does not exist
BKW0807E At least one of the options specified is unrecognized

AUTH DELOBJECT

```
►►── AUTH ── DELOBJECT ── object ──────────────────────────►◄
                                 └─( ──────────────┘
                                    └─ RULESONLY ─┘
```
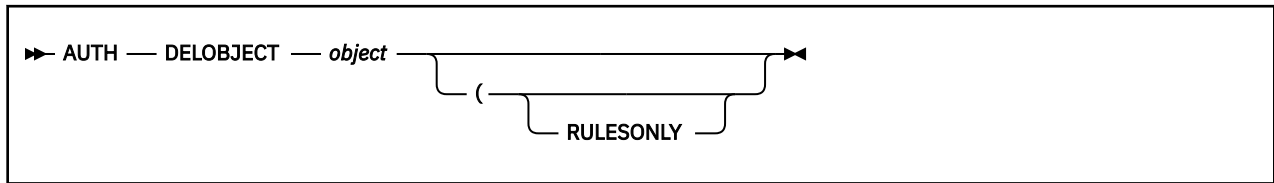
Purpose

Deletes the authorization rules for a given object.

Operands

object
> The object for which rules are to be deleted.

Options

RULESONLY
> Delete the rules for the object, but leave the object itself in the authorization database.

Usage Notes

1. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.
2. If RULESONLY is omitted, then the object itself is also deleted from the authorization database.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0807E At least one of the options specified is unrecognized
BKW0808E The object specified does not exist

AUTH DELUSER

```
►►── AUTH ──── DELUSER ──── userid ──────────────────►◄
                                    └──── class ────┘
```
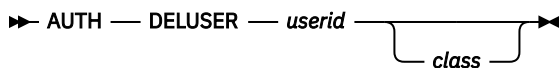
Purpose

Deletes authorization rules for a user.

Operands

userid
     The user ID for which authorization rules are to be deleted.

class
     The class from which *userid*'s rules are to be deleted.

Options

None

Usage Notes

1. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

2. If *class* is not specified, then *userid*'s rules for all classes are deleted.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0807E At least one of the options specified is unrecognized
BKW0810E No rules exist for the userid specified

AUTH LISTCLASS

```
►►── AUTH ── LISTCLASS ──┬───────*───────┬──►◄
                         └── match_key ──┘
```

Purpose

Lists the classes defined in the authorization data.

Operands

match_key
>   The key a class ID must match in order for it to show up in the output.

Options

None

Usage Notes

1. *match_key* is expressed using the CMS Application Multitasking syntax for IPC and event keys.

2. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

3. Output from this command appears as follows:

```
For class: File
 R     W

For class: Dir
 R     W     NR    NW

For class: Service
 STRT  STOP  EXEC
```

   The output just cites each class and then follows the citation with a list of the operations defined on it.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0805E The class specified does not exist
BKW0807E At least one of the options specified is unrecognized
BKW0813E No classes exist for the match key specified

AUTH LISTOBJECT

```
►►─ AUTH ── LISTOBJECT ── class_id ──┬──────*──────┬─►◄
                                     └── match_key ─┘
```
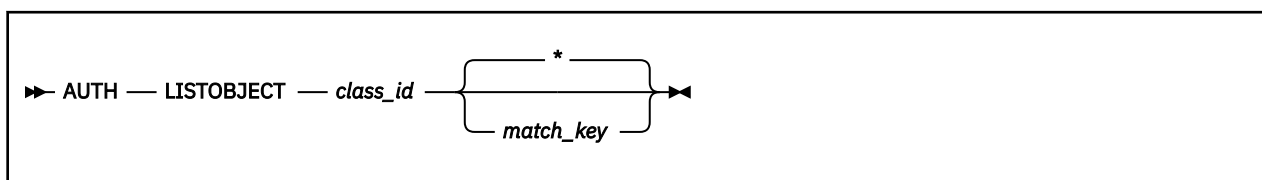
Purpose

Lists the objects belonging to a specified class.

Operands

match_key
> The key an object name must match in order for it to show up in the output.

Options

None

Usage Notes

1. Operand *match_key* is expressed using the CMS Application Multitasking syntax for IPC and event keys.

2. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

3. Output from this command appears as follows:

```
For class: Service
ECHO
SGEXER
HTTP
AUTH
CACHE
CONFIG
ENROLL
MONITOR
SERVER
SGP
USERID
CP
CMS
```

> The name of the class appears, followed by a list of the names of the objects in the class.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0805E The class specified does not exist
BKW0807E At least one of the options specified is unrecognized
BKW0814E No objects exist for the match key specified

AUTH MODCLASS

```
▶▶─ AUTH ── MODCLASS ── class ──┬── operation ──┬─▶◀
                                └──────◀───────┘
```

Purpose

Adds operations to the definition of an existing object.

Operands

class
> The name of the class to be modified.

operation
> The name of an operation to be defined on objects of this class.

Options

None

Usage Notes

For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

BKW0005E Out of storage.
BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0805E The class specified does not exist
BKW0812E Operation limit for the class specified has been exceeded

AUTH PERMIT

```
>>── AUTH ── PERMIT ── userid ── object ──┬── operation ──┬──┬──────── ( ── ADD ───────┬──><
                                          └──────◄────────┘  │                          │
                                                             │          ┌── ADD ───┐    │
                                                             └── ( ──────┼── REMOVE ─┼───┘
                                                                         └── REPLACE ─┘
```
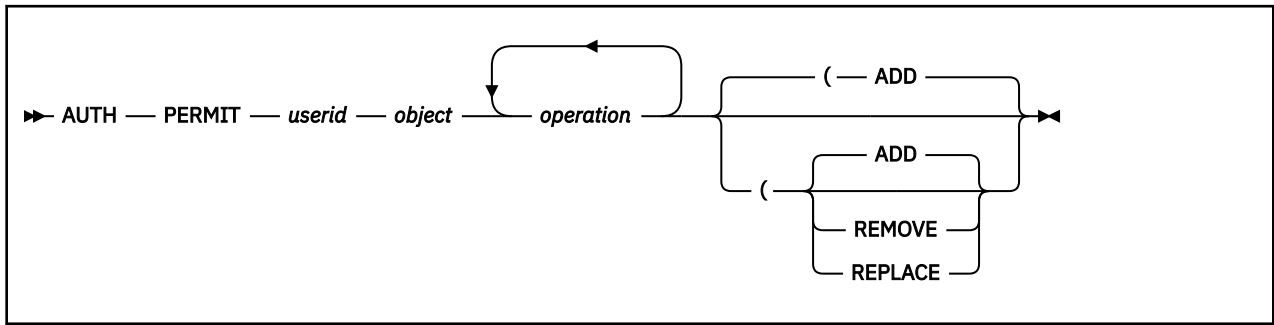
Purpose

Controls the operations a user can perform on an object.

Operands

userid
 The user ID to which this rule is to apply.

object
 The object to which this rule is to apply.

operation
 An operation defined on this object.

Options

ADD
 This rule is to be added to *userid*'s permissions for *object*.

REMOVE
 This rule is to be removed from *userid*'s permissions for *object*.

REPLACE
 This rule is to replace *userid*'s permissions for *object*.

Usage Notes

For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

BKW0005E Out of storage.
BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0808E The object specified does not exist

AUTH QOBJECT

```
▶▶── AUTH ── QOBJECT ── object ─────────────▶◀
                          └─ userid ─┘
```
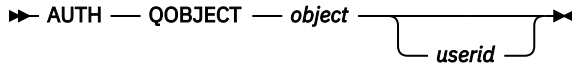
Purpose

Inquires about the permitted operations associated with a given object.

Operands

object
> The object for which rules are to be displayed.

userid
> The user ID for which rules are to be displayed.

Options

None

Usage Notes

1. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

2. If *userid* is supplied, then only *userid*'s rules for *object* are displayed.

3. If *userid* is omitted, then all rules for *object* are displayed.

4. Output from this command is as follows:

```
For object: SGP
 For userid: RANDOPM
  STRT STOP

 For userid: BKW
  STRT STOP EXEC
```

The output identifies the user IDs for whom there are rules in the data and for each such user ID the output lists the permitted operations.

Messages and Return Codes

BKW0005E Out of storage.
BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0803E Too many operations or options specified
BKW0808E The object specified does not exist
BKW0815E No userids exist for the object specified
BKW0816E No rules exist for the userid specified

AUTH RELOAD

```
►► AUTH ── RELOAD ►◄
```

Purpose

Causes the authorization API to reset its attempts to use the authorization database.

Operands

None

Options

None

Usage Notes

For support information, see "ssAuthReload — Reload Authorization Data" on page 240.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3
BKW0801E Unable to read the authorization files
BKW0802E Unable to write to the authorization files
BKW0811E Unable to open the authorization files

BKWENRCP

```
                            BKWENRDB ── *
►► ── BKWENRCP ── fn ──┬───────────────────────┬── ►◄
                       │                   * ─┐ │
                       └── ft ──┬───────────┤ │
                                ├── fm ────┤
                                └── dirid ─┘
```
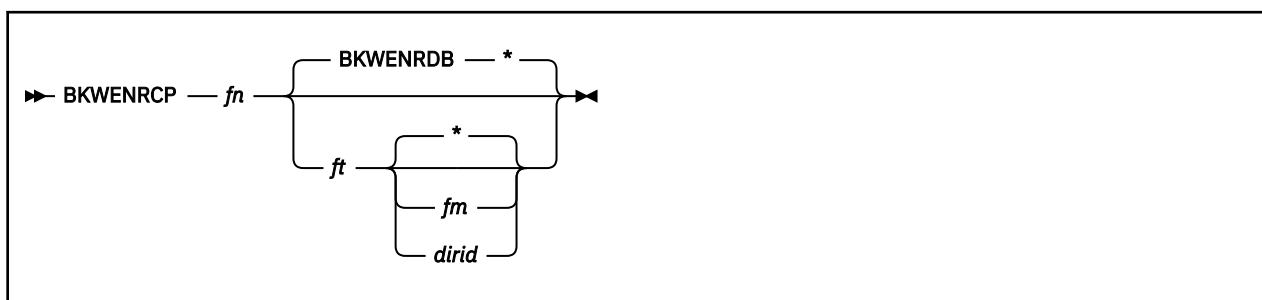
Purpose

Removes redundant information from the SFS file holding an enrollment set.

Operands

set_name
>   The name of the set to be interrogated.

fn
>   The file name of the SFS enrollment file.

ft
>   The file type of the SFS enrollment file.

fm
>   The file mode of the SFS enrollment file.

dirid
>   The directory name of the SFS enrollment file.

Options

None

Usage Notes

1. BKWENRCP is an EXEC, not an *internal* command provided by the reusable server kernel (such as the ENROLL command set).

2. To be processed by BKWENRCP, the SFS file containing the enrollment set must not be active -- that is, the corresponding enrollment set must be dropped through ENROLL DROP before BKWENRCP can work.

3. The output is written to the A file mode in a file whose file name matches *fn* and whose file type is BKWENRCP.

4. If your enrollment set is very large, a large virtual machine might be required to process it.

Messages and Return Codes

The return codes produced by BKWENRCP all come from CMS Pipelines. For more information, see *z/VM: CMS Pipelines User's Guide and Reference*.

CACHE CREATE

```
►►─ CACHE ── CREATE ── cname ──────────►◄
                           └─ size ─┘
```

Purpose

Creates a file cache.

Operands

cname
    The name of the file cache to be created.

size
    The size of the file cache, in pages.

Options

None

Usage Notes

1. The name *cname* is used directly in a call to `ssMemoryCreateDS` and therefore must be unique among all storage subpool names.

2. The cache size *size* is given in pages. It must be greater than zero and less than or equal to 524288. The size you specify is rounded up to the next 16-page boundary. If you do not specify a size, a size of 16 MB is used.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

CACHE DELETE

```
►►─ CACHE ── DELETE ── cname ─►◄
```

Purpose

Deletes a file cache.

Operands

cname
    The name of the file cache to be deleted.

Options

None

Usage Notes

1. Once deletion starts, no more new files will be cached.
2. The deletion completes after the last file is closed.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

CACHE LIST

```
►► CACHE ── LIST ►◄
```

Purpose

Lists the set of file caches.

Operands

None

Options

None

Usage Notes

The output form is:

```
Name      Size       InUse      FileCount  Opens    Hits
----      ----       -----      ---------  -----    ----
CACHE1    16384000   433567     421        1633     1185
CACHE2    32768000   2236541    28         4532     4158
```

The columns have the following meanings:

Column
    Meaning

Name
    Name of cache

Size
    Cache size in bytes

InUse
    Bytes in use in cache

FileCount
    Number of files in cache

Opens
    Number of file opens processed

Hits
    Number of cache hits on opens

Messages and Return Codes

BKW1500E No file caches found.

CMS

▶▶── CMS ── *cms_command_string* ──▶◀

Purpose

Provides a means of issuing CMS commands.

Operands

cms_command_string
  The command string to pass to CMS.

Options

None

Usage Notes

The command is issued by passing it to the CMS subcommand environment.

Messages and Return Codes

BKW1000I RC=&1 from CMS.

CONFIG AUT_CACHE

▶▶── CONFIG ── AUT_CACHE ── *rows* ──▶◀

Purpose

Sets the number of authorization rows that will be cached.

Operands

rows
    The number of rows to be cached.

Options

None

Usage Notes

For *rows*, specify a positive integer.

Messages and Return Codes

None

CONFIG AUT_DATA_1

▶▶─ CONFIG ── AUT_DATA_1 ── *filespec* ─▶◀

Purpose

Sets the name of copy 1 of the authorization data file.

Operands

filespec
 The name of copy 1 of the authorization data file.

Options

None

Usage Notes

1. For *filespec*, any string acceptable to DMSOPEN is acceptable.
2. Changing this parameter has no effect after PROFILE RSK has issued RUNSERV.

Messages and Return Codes

None

CONFIG AUT_DATA_2

▶▶── CONFIG ── AUT_DATA_2 ── *filespec* ──▶◀

Purpose

Sets the name of copy 2 of the authorization data file.

Operands

filespec
The name of copy 2 of the authorization data file.

Options

None

Usage Notes

1. For *filespec*, any string acceptable to DMSOPEN is acceptable.
2. Changing this parameter has no effect after PROFILE RSK has issued RUNSERV.
3. This parameter is ignored when AUT_LOCATION is set to SFS.

Messages and Return Codes

None

CONFIG AUT_FREE

▶▶── CONFIG ── AUT_FREE ── *rows* ──▶◀

Purpose

Sets the maximum number of free buffers that will be retained for the purpose of caching authorization rows.

Operands

rows
　　The maximum number of row buffers to retain.

Options

None

Usage Notes

For *rows*, specify a positive integer.

Messages and Return Codes

None

CONFIG AUT_INDEX_1

```
►►─ CONFIG ── AUT_INDEX_1 ── filespec ─►◄
```

Purpose

Sets the name of copy 1 of the authorization index file.

Operands

filespec
    The name of copy 1 of the authorization index file.

Options

None

Usage Notes

1. For *filespec*, any string acceptable to DMSOPEN is acceptable.
2. Changing this parameter has no effect after PROFILE RSK has issued RUNSERV.

Messages and Return Codes

None

CONFIG AUT_INDEX_2

▶▶── CONFIG ── AUT_INDEX_2 ── *filespec* ──▶◀

Purpose

Sets the name of copy 2 of the authorization index file.

Operands

filespec
> The name of copy 2 of the authorization index file.

Options

None

Usage Notes

1. For *filespec*, any string acceptable to DMSOPEN is acceptable.
2. Changing this parameter has no effect after PROFILE RSK has issued RUNSERV.
3. This parameter is ignored when AUT_LOCATION is set to SFS.

Messages and Return Codes

None

CONFIG AUT_LOCATION

```
▶▶── CONFIG ── AUT_LOCATION ──┬── MINIDISK ──┬──▶◀
                              └─── SFS ───────┘
```

Purpose

Sets the repository type of the authorization database.

Operands

MINIDISK
   The authorization database is stored on CMS minidisks.

SFS
   The authorization database is stored in the CMS Shared File System.

Options

None

Usage Notes

Changing this parameter has no effect after PROFILE RSK has issued RUNSERV.

Messages and Return Codes

None

CONFIG AUT_LOG

▶▶— CONFIG —— AUT_LOG —— *filespec* —▶◀

Purpose

Sets the name of the authorization logfile.

Operands

filespec
    The name of the authorization logfile.

Options

None

Usage Notes

1. Changing this parameter has no effect after `PROFILE  RSK` has issued `RUNSERV`.
2. For *filespec*, any syntax acceptable to `DMSOPEN` may be used.
3. This parameter is ignored when `AUT_LOCATION` is set to `SFS`.

Messages and Return Codes

None

CONFIG AUTHCHECK_AUTH

```
►►─ CONFIG ── AUTHCHECK_AUTH ──┬── ON ──┬──►◄
                               └── OFF ──┘
```
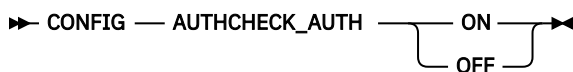
Purpose

Controls whether the AUTH commands will be subject to authorization checking.

Operands

ON
　　Authorization checking will be performed.

OFF
　　Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_CACHE

▶▶— CONFIG —— AUTHCHECK_CACHE ——┬— ON —┬—◀▶
                                └— OFF —┘

Purpose

Controls whether the CACHE commands will be subject to authorization checking.

Operands

ON
> Authorization checking will be performed.

OFF
> Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_CMS

```
►►── CONFIG ── AUTHCHECK_CMS ──┬── ON ──┬──►◄
                               └── OFF ──┘
```

Purpose

Controls whether the CMS service will perform authorization checking.

Operands

ON
> Authorization checking will be performed.

OFF
> Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_CONFIG

```
▶▶── CONFIG ── AUTHCHECK_CONFIG ──┬── ON ──┬──▶◀
                                  └── OFF ──┘
```

Purpose

Controls whether the CONFIG commands will be subject to authorization checking.

Operands

ON
Authorization checking will be performed.

OFF
Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_CP

```
▶▶─ CONFIG ── AUTHCHECK_CP ─┬─ ON ─┬─◀
                            └─ OFF ─┘
```

Purpose

Controls whether the CP service will perform authorization checking.

Operands

ON
    Authorization checking will be performed.

OFF
    Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_ENROLL

```
►►─ CONFIG ── AUTHCHECK_ENROLL ──┬─ ON ─┬─►◄
                                  └─ OFF ─┘
```

Purpose

Controls whether the ENROLL service will perform authorization checking.

Operands

ON
   Authorization checking will be performed.

OFF
   Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_LD

```
►►── CONFIG ── AUTHCHECK_LD ──┬── ON ──┬──►◄
                             └── OFF ──┘
```

Purpose

Controls whether line driver commands will be subject to authorization checking.

Operands

ON
Authorization checking will be performed.

OFF
Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_MONITOR

```
▶▶── CONFIG ── AUTHCHECK_MONITOR ──┬── ON ──┬──▶◀
                                   └── OFF ──┘
```

Purpose

Controls whether the MONITOR service will perform authorization checking.

Operands

ON
    Authorization checking will be performed.

OFF
    Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_SERVER

```
►►── CONFIG ── AUTHCHECK_SERVER ──┬── ON ──┬──►◄
                                  └── OFF ──┘
```

Purpose

Controls whether the SERVER commands will be subject to authorization checking.

Operands

ON
   Authorization checking will be performed.

OFF
   Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_SGP

```
►►— CONFIG —— AUTHCHECK_SGP ——┬— ON —┬—►◄
                              └— OFF —┘
```

Purpose

Controls whether the SGP commands will be subject to authorization checking.

Operands

ON
    Authorization checking will be performed.

OFF
    Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_TRIE

```
►►─ CONFIG ── AUTHCHECK_TRIE ──┬── ON ──┬──►◄
                               └─ OFF ──┘
```

Purpose

Controls whether the TRIE service will perform authorization checking.

Operands

ON
    Authorization checking will be performed.

OFF
    Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_USERID

```
▶▶─ CONFIG ── AUTHCHECK_USERID ─────── ON ──────▶◀
                                  └─ OFF ─┘
```

Purpose

Controls whether the USERID commands will be subject to authorization checking.

Operands

ON
    Authorization checking will be performed.

OFF
    Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG AUTHCHECK_WORKER

```
►►── CONFIG ── AUTHCHECK_WORKER ──┬── ON ──┬──►◄
                                  └── OFF ──┘
```
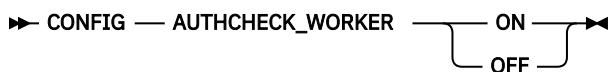
Purpose

Controls whether the WORKER commands will be subject to authorization checking.

Operands

ON
    Authorization checking will be performed.

OFF
    Authorization checking will not be performed.

Options

None

Usage Notes

For more information, see "Other Services' Use of Authorization" on page 40 and Table 31 on page 66.

Messages and Return Codes

None

CONFIG MEM_MAXFREE

▶▶─ CONFIG ── MEM_MAXFREE ── *pages* ─▶◀

Purpose

Sets the maximum number of pages that the reusable server kernel storage manager will retain for a given subpool before returning storage from that subpool to CMS.

Operands

pages
    The maximum number of pages to retain.

Options

None

Usage Notes

For *pages*, specify a positive integer.

Messages and Return Codes

None

CONFIG MON_KERNEL_ROWS

```
▶▶─ CONFIG ── MON_KERNEL_ROWS ── rows ─▶◀
```

Purpose

Sets the number of monitor data rows the reusable server kernel defines.

Operands

rows
    The number of rows to define.

Options

None

Usage Notes

1. You must choose *rows* in range [36..55000].
2. The reusable server kernel rounds *rows* to the next higher multiple of 55. For example, if you ask for 75 rows you will actually get 110.

Messages and Return Codes

None

CONFIG MON_PRODUCT_ID

▶▶── CONFIG ── MON_PRODUCT_ID ── *identifier* ──▶◀

Purpose

Sets the product identifier the reusable server kernel will use when it invokes Diagnose X'00DC' to start APPLDATA monitor data collection.

Operands

identifier
    The 16-byte identifier to use.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG MON_USER_SIZE

```
►►— CONFIG —— MON_USER_SIZE —— bytes ►◄
```

Purpose

Sets the size of the monitor buffer the reusable server kernel will reserve for application use.

Operands

bytes
   The number of bytes to reserve.

Options

None

Usage Notes

This command is obsolete. No matter what you ask for, you will now get 3952 bytes of user monitor buffer, which is the largest amount of user data the server kernel can put in a single monitor buffer.

Messages and Return Codes

None

CONFIG MSG_NOHDR

```
►►─ CONFIG ── MSG_NOHDR ──┬── ON ──┬─►◄
                          └── OFF ──┘
```
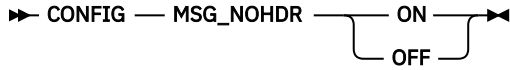
Purpose

Controls whether the MSG/SMSG line driver will use the MSGNOH command to reply to a client.

Operands

ON
MSGNOH will be used.
OFF
MSG will be used.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG NOMAP_APPC

```
►►─ CONFIG ── NOMAP_APPC ──┬── ON ──┬─►◄
                           └── OFF ──┘
```

Purpose

Controls whether the APPC line driver will pass unmappable user IDs to a service instance.

Operands

ON
Unmappable user IDs will be passed.

OFF
Unmappable user IDs will be rejected.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG NOMAP_IUCV

```
▶▶── CONFIG ── NOMAP_IUCV ──┬── ON ──┬──▶◀
                            └── OFF ──┘
```
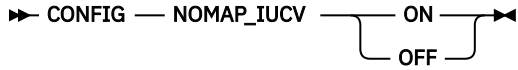
Purpose

Controls whether the IUCV line driver will pass unmappable user IDs to a service instance.

Operands

ON
    Unmappable user IDs will be passed.

OFF
    Unmappable user IDs will be rejected.

Options

None

Usage Notes

If NOMAP_IUCV is **ON**, unmappable user IDs will be passed as user ID $UNKNOWN.

Messages and Return Codes

None

CONFIG NOMAP_MSG
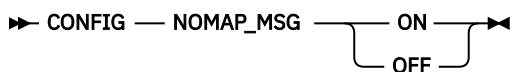
```
►►─ CONFIG ── NOMAP_MSG ──┬── ON ──┬─►◄
                          └── OFF ──┘
```

Purpose

Controls whether the MSG/SMSG line driver will pass unmappable user IDs to a service instance.

Operands

ON
    Unmappable user IDs will be passed.

OFF
    Unmappable user IDs will be rejected.

Options

None

Usage Notes

If NOMAP_MSG is **ON**, unmappable user IDs will be passed as user ID $UNKNOWN.

Messages and Return Codes

None

CONFIG NOMAP_SPOOL
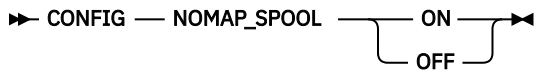
```
►►─ CONFIG ── NOMAP_SPOOL ──┬── ON ──┬─►◄
                            └── OFF ──┘
```

Purpose

Controls whether the SPOOL line driver will pass unmappable user IDs to a service instance.

Operands

ON
  Unmappable user IDs will be passed.

OFF
  Unmappable user IDs will be rejected.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG NOMAP_TCP

```
►►── CONFIG ── NOMAP_TCP ──┬── ON ──┬──►◄
                           └── OFF ──┘
```

Purpose

Controls whether the TCP line driver will pass unmappable user IDs to a service instance.

Operands

ON
 Unmappable user IDs will be passed.

OFF
 Unmappable user IDs will be rejected.

Options

None

Usage Notes

If NOMAP_TCP is **ON**, unmappable user IDs will be passed as user ID $UNKNOWN.

Messages and Return Codes

None

CONFIG NOMAP_UDP

```
►►─ CONFIG ── NOMAP_UDP ──┬── ON ──┬─►◄
                          └── OFF ──┘
```

Purpose

Controls whether the UDP line driver will pass unmappable user IDs to a service instance.

Operands

ON
    Unmappable user IDs will be passed.

OFF
    Unmappable user IDs will be rejected.

Options

None

Usage Notes

If NOMAP_UDP is **ON**, unmappable user IDs will be passed as user ID $UNKNOWN.

Messages and Return Codes

None

CONFIG RSCS_USERID

```
►►── CONFIG ── RSCS_USERID ── userid ──►◄
```

Purpose

Sets the user ID of the virtual machine in which the SPOOL and MSG/SMSG line drivers will assume RSCS is running.

Operands

userid
> The user ID of the RSCS machine.

Options

None

Usage Notes

Most installations will tailor `PROFILE  RSK` so that it issues CMS's `IDENTIFY` command, parses the response so as to obtain the user ID of the RSCS machine, and then issues an appropriate `CONFIG RSCS_USERID` command.

Messages and Return Codes

None

CONFIG SGP_FILE

```
▶▶─ CONFIG ── SGP_FILE ── filespec ─▶◀
```

Purpose

Sets the name of the storage group configuration file.

Operands

filespec
   The string identifying the storage group configuration file.

Options

None

Usage Notes

1. For *filespec*, any string acceptable to DMSOPEN is acceptable.
2. Changing this parameter has no effect after PROFILE RSK has issued RUNSERV.

Messages and Return Codes

None

CONFIG SPL_CATCHER

```
►►─ CONFIG ── SPL_CATCHER ── userid ─►◄
```

Purpose

Controls the user ID to which the SPOOL driver will transfer spool files it is unable to decode.

Operands

userid
>   The user ID to which the SPOOL driver will transfer files it is unable to decode.

Options

None

Usage Notes

1. The SPOOL line driver is able to decode files sent in NETDATA (aka SENDFILE NEW) or DISK DUMP (aka SENDFILE OLD) formats. All other formats are undecodable.
2. If *userid* is ⋆, the reusable server kernel will leave such files in the server's reader in USER HOLD status.

Messages and Return Codes

None

CONFIG SPL_INPUT_FT

```
►►── CONFIG ──── SPL_INPUT_FT ──── filetype ──►◄
```

Purpose

Controls the file type the SPOOL driver will recognize as input for a service.

Operands

filetype
The file type the SPOOL line driver will recognize.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG SPL_OUTPUT_FT

▶▶── CONFIG ── SPL_OUTPUT_FT ── *filetype* ──▶◀

Purpose

Controls the file type the SPOOL driver will produce as output from a service.

Operands

filetype
    The file type the SPOOL line driver will produce.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG SRV_THREADS

▶▶ CONFIG —— SRV_THREADS —— *threads* ▶◀

Purpose

Controls the number of threads on which a given line driver will attempt to run a given service.

Operands

threads
　　The maximum number of threads on which a given line driver will attempt to run a given service.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG UMAP_FILE

```
►►─ CONFIG ── UMAP_FILE ── filespec ─►◄
```

Purpose

Sets the name of the user ID mapping file.

Operands

filespec
    The string identifying the user ID mapping file.

Options

None

Usage Notes

For *filespec*, any string acceptable to DMSOPEN is acceptable.

Messages and Return Codes

None

CONFIG VM_CONSOLE

```
▶▶── CONFIG ── VM_CONSOLE ──┬── ON ──┬─▶◀
                           └── OFF ──┘
```
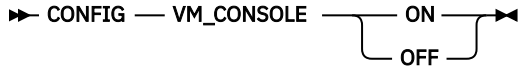
Purpose

Controls whether the console line driver will pass unrecognized input to CMS for execution.

Operands

ON
    The console driver will pass unrecognized input to CMS.

OFF
    The console driver will not pass unrecognized input to CMS.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG VM_MSG
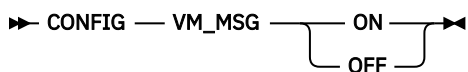
```
►►── CONFIG ── VM_MSG ──┬── ON ──┬──►◄
                        └── OFF ──┘
```

Purpose

Controls whether the MSG/SMSG line driver will pass unrecognized input to CMS for execution.

Operands

ON
 The MSG/SMSG driver will pass unrecognized input to CMS.

OFF
 The MSG/SMSG driver will not pass unrecognized input to CMS.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG VM_SPOOL

```
▶▶─ CONFIG ── VM_SPOOL ──┬── ON ──┬─▶◀
                         └── OFF ──┘
```

Purpose

Controls whether the SPOOL line driver will pass unrecognized input to CMS for execution.

Operands

ON
   The SPOOL driver will pass unrecognized input to CMS.

OFF
   The SPOOL driver will not pass unrecognized input to CMS.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONFIG VM_SUBCOM

```
►►── CONFIG ── VM_SUBCOM ──┬── ON ──┬──►◄
                           └── OFF ──┘
```
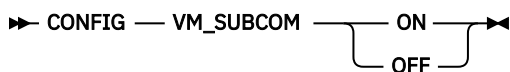
Purpose

Controls whether the SUBCOM line driver will pass unrecognized input to CMS for execution.

Operands

ON
　　The SUBCOM driver will pass unrecognized input to CMS.

OFF
　　The SUBCOM driver will not pass unrecognized input to CMS.

Options

None

Usage Notes

None

Messages and Return Codes

None

CONSOLE LIST

```
▶▶── CONSOLE ──── LIST ──▶◀
```

Purpose

Lists the subtasks associated with the console line driver.

Operands

None

Options

None

Usage Notes

This command displays information about the services started through the console line driver. The output form is:

```
Subtask  Service  Prefix   Instances
-------  -------  ------   ---------
0        CONSOLE  CONSOLE  1
1        SERVER   SERVER   1
```

The columns have the following meanings:

Subtask
    The numeric identifier of the subtask.

Service
    The name of the started service.

Prefix
    The prefix used to send input to the service.

Instances
    The number of instances of the service the line driver is controlling.

Messages and Return Codes

None

CONSOLE QUERY

```
►►─ CONSOLE ── QUERY ── subtaskid ─►◄
```

Purpose

Queries a specific console subtask.

Operands

subtaskid
    The identifier of the subtask to query.

Options

None

Usage Notes

This command displays information about all of the instances of the requested subtask. The output form is:

```
Instance C-block  ThreadID Userid    BytesIn    BytesOut
-------- -------  -------- ------    -------    --------
1        01EE0F5C 16       *         175        446
```

In this output, the columns have the following meanings:

Instance
    The numeric identifier of the instance.

C-block
    The address of the instance's C-block.

ThreadID
    The CMS thread ID of the thread on which the instance is running.

Userid
    The user ID of the client affiliated with the instance.

BytesIn
    The number of bytes the client has provided to the instance.

BytesOut
    The number of bytes the instance has provided to the client.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

CONSOLE START

```
►►─── CONSOLE ─── START ─── servicename ──────────────►◄
                                      └─ prefix ─┘
```

Purpose

Starts a service, connecting it to the console line driver.

Operands

servicename
> The name of the service to start, as specified on a call to `ssServiceBind`.

prefix
> The prefix that will identify commands that should be sent to this service.

Options

None

Usage Notes

1. If *prefix* is not specified, the value of *servicename* is used for the prefix.

2. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

Messages and Return Codes

BKW0005E Out of storage.
BKW0200E Service not found.
BKW0205E Prefix already in use.
BKW0206E Service INIT routine failed - RC=&1 RE=&2.
BKW0207E Start of self is prohibited.

CONSOLE STOP

```
►►─ CONSOLE ── STOP ── subtaskid ──────────────►◄
                              └─ NOW ─┘
```

Purpose

Stops a specific console subtask, optionally denying currently-connected clients the privilege of completing their operations.

Operands

subtaskid
    The identifier of the subtask to stop.

Options

NOW
    Stop the subtask without letting current clients complete normally.

Usage Notes

None

Messages and Return Codes

BKW0201E Subtask not found.
BKW0202E Stop of self is prohibited.
BKW0203I Subtask asked to STOP.
BKW0204I Subtask killed.

CP

```
►►── CP ── cp_command_string ──►◄
```

Purpose

Provides a means of issuing CP commands.

Operands

cp_command_string
>   The command string to pass to CP.

Options

None

Usage Notes

The command is issued by passing it to CP through DIAG X'08'.

Messages and Return Codes

BKW0900I RC=&1 from CP.
BKW0901E CP response was truncated.
BKW0902E CP command was too long.

ENROLL COMMIT

```
▶▶── ENROLL ── COMMIT ── set_name ──▶◀
```

Purpose

Commits changes to the named enrollment set.

Operands

set_name
    The name of the set to be committed.

Options

None

Usage Notes

For more information, see .

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

ENROLL DROP
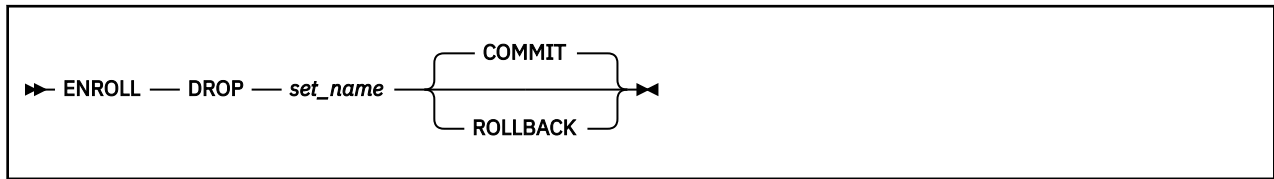
```
►►─── ENROLL ── DROP ── set_name ──┬── COMMIT ──┬──►◄
                                   └── ROLLBACK ──┘
```

Purpose

Commits changes to the named enrollment set.

Operands

set_name
> The name of the set to be committed.

COMMIT
> The uncommitted changes should be committed.

ROLLBACK
> The uncommitted changes should be rolled back.

Options

None

Usage Notes

For more information, see .

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

ENROLL GET

```
▶▶── ENROLL ── GET ── set_name ── key ──◀◀
```

Purpose

Retrieves a record from an enrollment set.

Operands

set_name
 The name of the set to be interrogated.

key
 The key of the record to be retrieved.

Options

None

Usage Notes

1. Due to parsing considerations, *key* must not contain a left parenthesis or a space.
2. For more information, see "Usage Notes" on page 272.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

ENROLL INSERT

```
▶▶── ENROLL ── INSERT ── set_name ── key ── data ──▶◀
```

Purpose

Inserts or replaces a record in an enrollment set.

Operands

set_name
  The name of the set to be updated.

key
  The key of the record to be inserted.

data
  The data to be inserted.

Options

None

Usage Notes

1. Due to parsing considerations, *key* must not contain a left parenthesis or a space.

2. The record is inserted with method *ss_enr_insert_replace*.

3. For more information, see "Usage Notes" on page 274.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

ENROLL LIST

```
►► ENROLL ── LIST ►◄
```

Purpose

Generates a list of the loaded enrollment sets.

Operands

None

Options

None

Usage Notes

For more information, see "ssEnrollList — List Enrollment Sets" on page 268.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

ENROLL LOAD

►► ENROLL ── LOAD ── *set_name* ─┬─ DISK ─┬─ *size* ── *filename* ►◄
                                 └─ MEM ──┘

Purpose

Loads an enrollment set from the Shared File System, or initializes a transient enrollment set.

Operands

set_name
    The name of the set to be loaded.

DISK
    This is a permanent enrollment set.

MEM
    This is a transient enrollment set.

size
    The data space size to use, in pages.

filename
    The file specification of the Shared File System file to be used.

Options

None

Usage Notes

For more information, see "ssEnrollLoad — Load Enrollment Set" on page 270.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

ENROLL RECLIST

```
▶▶─── ENROLL ──── RECLIST ──── set_name ─▶◀
```

Purpose

Generates a list of the keys of the records stored in the named enrollment set.

Operands

set_name
   The name of the set to be interrogated.

Options

None

Usage Notes

For more information, see "Usage Notes" on page 276.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

ENROLL REMOVE

▶▶── ENROLL ── REMOVE ── *set_name* ── *key* ──◀◀

Purpose

Removes a record from an enrollment set.

Operands

set_name
  The name of the set to be updated.

key
  The key of the record to be removed.

Options

None

Usage Notes

1. Due to parsing considerations, *key* must not contain a left parenthesis or a space.
2. For more information, see "Usage Notes" on page 278.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

IUCV LIST

```
▶▶─ IUCV ── LIST ─◀◀
```

Purpose

Lists the subtasks associated with the IUCV line driver.

Operands

None

Options

None

Usage Notes

The output form is:

```
Subtask ServName ExitName Capacity InUse Threads Waiters
------- -------- -------- -------- ----- ------- -------
0       ECHO     ECHO     40       0     1       0
```

The columns have the following meanings:

Subtask
The numeric identifier of the subtask.

ServName
The name of the started service.

ExitName
The name of the IUCV exit for this subtask.

Capacity
The number of clients this subtask can handle concurrently.

InUse
The number of clients currently connected.

Threads
The number of threads available to service clients of this subtask.

Waiters
The number of clients waiting to be serviced.

Messages and Return Codes

BKW0201E Subtask not found.

IUCV QUERY

```
▶▶── IUCV ── QUERY ── subtaskid ──▶◀
```

Purpose

Queries a specific IUCV subtask.

Operands

subtaskid
    The identifier of the subtask to query.

Options

None

Usage Notes

The output form is:

```
Instance C-Block  Userid   BytesIn   BytesOut
-------- -------  ------   -------   --------
32       01D2E6DC RICHARD  22        22
```

The columns have the following meanings:

Instance
    The numeric identifier of the instance.

C-Block
    The address of the C-block for this client.

Userid
    The mapped user ID of the client.

BytesIn
    The number of bytes the IUCV line driver has queued for the instance.

BytesOut
    The number of bytes the instance has queued for the IUCV line driver to transmit to the client.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

IUCV REPORT

```
▶▶ IUCV ── REPORT ──┬── ON ──┬── ▶◀
                    └── OFF ──┘
```

Purpose

Toggles reporting state for the IUCV line driver.

Operands

ON
> Turns reporting on.

OFF
> Turns reporting off.

Options

None

Usage Notes

When reporting is on, the IUCV line driver issues the following messages to describe client activity:

- BKW1602I
- BKW1603I
- BKW1604I
- BKW1605I

For more information, see "IUCV Line Driver Messages" on page 409.

Messages and Return Codes

None

IUCV START

```
▶▶─ IUCV ── START ── servicename ─┬──────────────────────────────┬─▶◀
                                  └─ maxclients ─┬──────────────┬─┘
                                                 └─ exitname ───┘
```
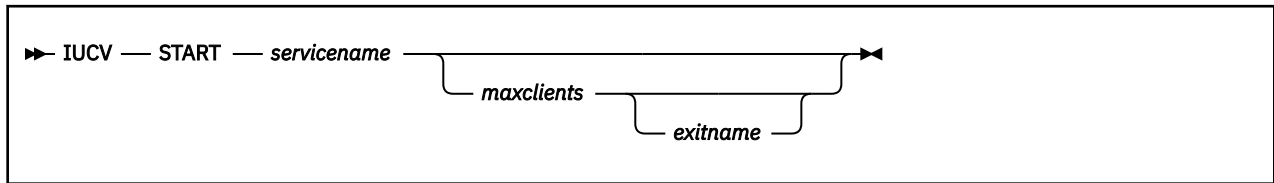
Purpose

Starts a service, connecting it to the IUCV line driver.

Operands

servicename
>   The name of the service to start, as specified on a call to `ssServiceBind`.

maxclients
>   The maximum number of concurrent clients permitted for the subtask.

exitname
>   The `HNDIUCV` exit name to be used for the subtask.

Options

None

Usage Notes

1. If *maxclients* is not specified, the current value of configuration parameter SRV_THREADS is used.

2. If *exitname* is not specified, the value of *servicename* is used.

3. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

Messages and Return Codes

BKW0200E Service not found.
BKW0207E Start of self is prohibited.
BKW1607E Client count must be greater than zero.
BKW1608E Unable to HNDIUCV SET.
BKW1609E Unable to create controlling thread.

IUCV STOP

```
►►─ IUCV ── STOP ── subtaskid ─────────────────►◄
                           ├─ NOW ──────┤
                           └─ instance ─┘
```
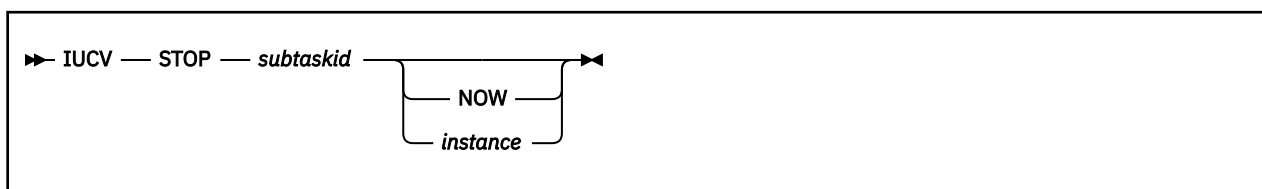
Purpose

Stops a specific IUCV subtask, optionally denying currently-connected clients the privilege of completing their operations, or stops a specific client and affiliated instance.

Operands

subtaskid
    The identifier of the subtask to stop.

instance
    The number of the instance to stop.

Options

NOW
    Stop the subtask without letting current clients complete normally.

Usage Notes

1. If NOW is specified, the subtask is stopped immediately and clients are not given the opportunity to finish their work.

2. If *instance* is specified, only that specific connection is terminated.

Messages and Return Codes

BKW0201E Subtask not found.
BKW1600I Instance STOP requested.
BKW1606E Wait expired for STOP.

MONITOR DISPLAY

```
►►── MONITOR ── DISPLAY ──┬──────────────────┬──►◄
                          └─ type ──┬───────┬─┘
                                    └─ name ─┘
```
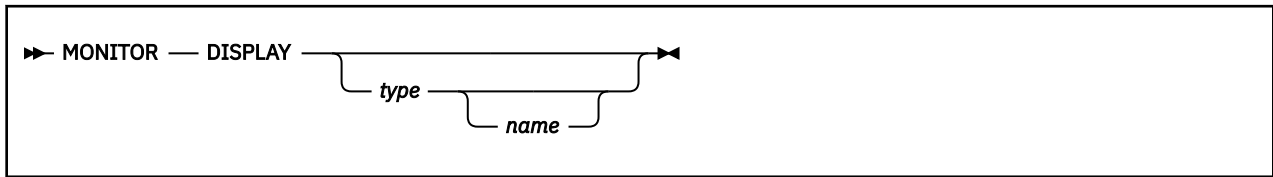
Purpose

Displays one or more rows of monitor data.

Operands

type
    The type of monitor row to display.

size
    The name of a specific monitor row of the given type.

Options

None

Usage Notes

1. If *type* is not specified, all monitor rows are displayed.

2. If only *type* is specified, all rows of the specified type are displayed.

3. If both *type* and *name* are specified, the specific row described is displayed.

4. For each qualifying monitor row, the display consists simply of the address and length of the row and the storage at those locations.

Messages and Return Codes

BKW1400E Matching monitor row not found.

MONITOR USER

```
▶▶─ MONITOR ── USER ─▶◀
```

Purpose

Displays the user monitor buffer.

Operands

None

Options

None

Usage Notes

The display consists simply of the address and length of the user monitor buffer and the storage at those locations.

Messages and Return Codes

None

MSG LIST

```
►►── MSG ─── LIST ──►◄
```

Purpose

Lists the subtasks associated with the MSG/SMSG line driver.

Operands

None

Options

None

Usage Notes

This command displays information about the services started through the MSG/SMSG line driver. The output form is:

```
Subtask  Service  Prefix   Instances
-------  -------  ------   ---------
0        MSG      MSG      1
1        SERVER   SERVER   1
```

The columns have the following meanings:

Subtask
    The numeric identifier of the subtask.

Service
    The name of the started service.

Prefix
    The prefix used to send input to the service.

Instances
    The number of instances of the service the line driver is controlling.

Messages and Return Codes

None

MSG QUERY

```
►►── MSG ── QUERY ── subtaskid ──►◄
```

Purpose

Queries a specific MSG/SMSG subtask.

Operands

subtaskid
    The identifier of the subtask to query.

Options

None

Usage Notes

This command displays information about all of the instances of the requested subtask. The output form is:

```
Instance C-block  ThreadID Userid   BytesIn   BytesOut
-------- -------  -------- ------   -------   --------
1        01EE0F5C 16       BKW      175       446
```

In this output, the columns have the following meanings:

Instance
    The numeric identifier of the instance.

C-block
    The address of the instance's C-block.

ThreadID
    The CMS thread ID of the thread on which the instance is running.

Userid
    The user ID of the client affiliated with the instance.

BytesIn
    The number of bytes the client has provided to the instance.

BytesOut
    The number of bytes the instance has provided to the client.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

MSG START

```
▶▶─ MSG ── START ── servicename ──┬──────────┬──▶◀
                                  └─ prefix ─┘
```
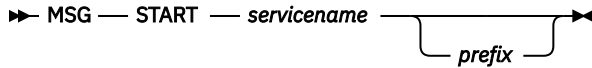
Purpose

Starts a service, connecting it to the MSG/SMSG line driver.

Operands

servicename
    The name of the service to start, as specified on a call to `ssServiceBind`.

prefix
    The prefix that will identify commands that should be sent to this service.

Options

None

Usage Notes

1. If *prefix* is not specified, then the value of *servicename* is used for the prefix.

2. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

Messages and Return Codes

BKW0005E Out of storage.
BKW0200E Service not found.
BKW0205E Prefix already in use.
BKW0206E Service INIT routine failed - RC=&1 RE=&2.
BKW0207E Start of self is prohibited.

MSG STOP

```
►►── MSG ── STOP ── subtaskid ─┬──────────┬──►◄
                               └── NOW ────┘
```
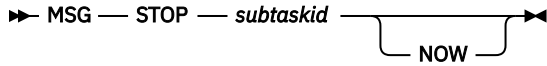
Purpose

Stops a specific MSG/SMSG subtask, optionally denying currently-connected clients the privilege of completing their operations.

Operands

subtaskid
    The identifier of the subtask to stop.

Options

NOW
    Stop the subtask without letting current clients complete normally.

Usage Notes

None

Messages and Return Codes

BKW0201E Subtask not found.
BKW0202E Stop of self is prohibited.
BKW0203I Subtask asked to STOP.
BKW0204I Subtask killed.

SERVER SERVICES

```
▶▶ SERVER ── SERVICES ▶◀
```

Purpose

Displays a summary of the bound services.

Operands

None

Options

None

Usage Notes

This command causes the reusable server kernel to display a list of the bound services with some descriptive information about each service. The output form is:

```
Service  S-block  Type Init     Service  Term    Count
-------  -------  ---- ----     -------  ----    -----
USERID   01EFEF40 N    00000000 81E94530 81E94D18 0
SERVER   01EFEF70 N    00000000 81E94530 81E94D18 1
CONFIG   01EFEFA0 N    00000000 81E94530 81E94D18 1
CONSOLE  01EFEFD0 LDSS 81E93478 81E939C8 81E94408 1
```

The meanings of the columns are:

Service
> The name of the bound service.

S-block
> The address of the service's S-block.

Type
> The type of the bound service. Types are:
>
> **N**
>> Normal service
>
> **LD**
>> Line driver
>
> **LDSS**
>> Self-sourced line driver

Init
> The address of the service's initialization routine.

Service
> The address of the service's service routine.

Term
> The address of the service's termination routine.

Count
> The number of line drivers that have started this service.

Messages and Return Codes

None

SERVER MONITOR

```
▶▶─ SERVER ── MONITOR ─▶◀
```

Purpose

Gives information about the Diagnose X'00DC' monitor buffers.

Operands

None

Options

None

Usage Notes

For each monitor buffer, this command tells the user the:

- Location of the monitor buffer
- Size of the monitor buffer
- Number of rows in the monitor buffer
- Number of free rows in the monitor buffer

Messages and Return Codes

BKW0301I Monitor buffer at &1.&2, &3 rows, &4 free

SERVER STOP

```
▶▶─ SERVER ── STOP ─▶◀
```

Purpose

Stops the server and the reusable server kernel.

Operands

None

Options

None

Usage Notes

Issuing this command is equivalent to calling entry point `ssServerStop`. Both of these facilities cause `WAITSERV` to complete.

Messages and Return Codes

None

SGP CREATE
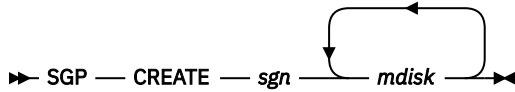
Purpose

Creates a storage group.

Operands

sgn
> The number of the storage group to create.

mdisk
> The device number of a minidisk to be used for the storage group.

Options

None

Usage Notes

For more information, see "Usage Notes" on page 293.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

SGP DELETE

```
▶▶ SGP ── DELETE ── sgn ◀▶
```

Purpose

Deletes a storage group.

Operands

sgn
    The number of the storage group to delete.

Options

None

Usage Notes

For more information, see .

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

SGP LIST

```
▶▶ SGP ── LIST ▶◀
```

Purpose

Displays a list of the known storage groups.

Operands

None

Options

None

Usage Notes

1. This command causes the reusable server kernel to display a list of the known storage groups. The output format is:

```
SGrp Name      Blocks     IOMode Status
---- ----      ------     ------ ------
2    main      4000       blk-rw 40000000
5    spare     82400      blk-ro 20000000
```

The meanings of the columns are:

SGrp
The storage group number.

Name
The name of the storage group.

Blocks
The total number of 4 KB blocks in the storage group.

IOMode
The mode in which the storage group was started.

off
not started

blk-ro
block mode read-only

blk-rw
block mode read-write

Status
Status bits

X'80000000'
Stop is in progress

X'40000000'
I/O using VM Data Spaces

X'20000000'
I/O using DIAG X'250'

2. For more information, see "Usage Notes" on page 299 and "Usage Notes" on page 302.

Messages and Return Codes

BKW0005E Out of storage.
BKW0007E RC=&1 RE=&2 from routine &3

SGP MDLIST

```
▶▶── SGP ── MDLIST ── sgn ──▶◀
```

Purpose

Displays specific information about the minidisks of a storage group.

Operands

sgn
    The number of the storage group to interrogate.

Options

None

Usage Notes

1. This command causes the reusable server kernel to display a list of the minidisks associated with a
   given storage group. The output format is:

   ```
   VDev Blocks
   ---- ------
   1004 34006
   0FC2 14200
   ```

   The meanings of the columns are:

   VDev
       The device number of the minidisk.

   Total
       The number of 4 KB blocks on the minidisk.

2. For more information, see "Usage Notes" on page 302.

Messages and Return Codes

BKW0005E Out of storage.
BKW0007E RC=&1 RE=&2 from routine &3

SGP START

```
►►─ SGP ── START ── sgn ── groupname ─┬──────────────┬─┬── DS ──┬─►◄
                                      │   BLOCKRW    │ │        │
                                      ├── BLOCKRO ───┤ ├── DS ──┤
                                      └── BLOCKRW ───┘ └─ NODS ─┘
```

Purpose

Starts a specific storage group.

Operands

sgn
　　The number of the storage group to start.

groupname
　　The symbolic name to be assigned to the storage group.

BLOCKRO
　　The storage group should be started in block mode read-only.

BLOCKRW
　　The storage group should be started in block mode read-write.

DS
　　The reusable server kernel should attempt to use VM Data Spaces for I/O.

NODS
　　The reusable server kernel should not attempt to use VM Data Spaces for I/O.

Options

None

Usage Notes

For more information, see "Usage Notes" on page 306.

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

SGP STOP

```
▶▶ SGP —— STOP —— sgn ▶◀
```

Purpose

Stops a specific storage group.

Operands

sgn
    The number of the storage group to stop.

Options

None

Usage Notes

For more information, see .

Messages and Return Codes

BKW0007E RC=&1 RE=&2 from routine &3

SPOOL LIST

```
►► SPOOL ── LIST ►◄
```

Purpose

Lists the subtasks associated with the SPOOL line driver.

Operands

None

Options

None

Usage Notes

This command displays information about the services started through the spool line driver. The output form is:

```
Subtask  Service  Prefix   Instances
-------  -------  ------   ---------
0        SPOOL    SPOOL    1
1        SERVER   SERVER   1
```

The columns have the following meanings:

Subtask
    The numeric identifier of the subtask.

Service
    The name of the started service.

Prefix
    The file name used to send input to the service.

Instances
    The number of instances of the service the line driver is controlling.

Messages and Return Codes

None

SPOOL QUERY

▶▶── SPOOL ── QUERY ── *subtaskid* ──▶◀

Purpose

Queries a specific SPOOL subtask.

Operands

subtaskid
    The identifier of the subtask to query.

Options

None

Usage Notes

This command displays information about all of the instances of the requested subtask. The output form is:

```
Instance C-block  ThreadID Userid    BytesIn   BytesOut
-------- -------  -------- ------    -------   --------
1        01EE0F5C 16       BKW       175       446
```

In this output, the columns have the following meanings:

Instance
    The numeric identifier of the instance.

C-block
    The address of the instance's C-block.

ThreadID
    The CMS thread ID of the thread on which the instance is running.

Userid
    The user ID of the client affiliated with the instance.

BytesIn
    The number of bytes the client has provided to the instance.

BytesOut
    The number of bytes the instance has provided to the client.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

SPOOL START

```
►►── SPOOL ── START ── servicename ── spoolfn ──►◄
```

Purpose

Starts a service, connecting it to the SPOOL line driver.

Operands

servicename
The name of the service to start, as specified on a call to `ssServiceBind`.

spoolfn
The file name of spool files that should be directed to this service.

Options

None

Usage Notes

1. If *prefix* is not specified, then the value of *servicename* is used for the prefix.
2. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

Messages and Return Codes

BKW0005E Out of storage.
BKW0200E Service not found.
BKW0205E Prefix already in use.
BKW0206E Service INIT routine failed - RC=&1 RE=&2.
BKW0207E Start of self is prohibited.

SPOOL STOP

```
►►── SPOOL ── STOP ── subtaskid ──┬──────────┬──►◄
                                  └── NOW ──┘
```

Purpose

Stops a specific SPOOL subtask, optionally denying currently-connected clients the privilege of completing their operations.

Operands

subtaskid
　　The identifier of the subtask to stop.

Options

NOW
　　Stop the subtask without letting current clients complete normally.

Usage Notes

None

Messages and Return Codes

BKW0201E Subtask not found.
BKW0202E Stop of self is prohibited.
BKW0203I Subtask asked to STOP.
BKW0204I Subtask killed.

SUBCOM LIST

```
►► SUBCOM ── LIST ►◄
```

Purpose

Lists the subtasks associated with the SUBCOM line driver.

Operands

None

Options

None

Usage Notes

This command displays information about the services started through the SUBCOM line driver. The output form is:

```
Subtask  Service  Prefix  Instances
-------  -------  ------  ---------
0        SUBCOM   SUBCOM  1
1        SERVER   SERVER  1
```

The columns have the following meanings:

Subtask
   The numeric identifier of the subtask.

Service
   The name of the started service.

Prefix
   The prefix used to send input to the service.

Instances
   The number of instances of the service the line driver is controlling.

Messages and Return Codes

None

SUBCOM QUERY

```
►►── SUBCOM ── QUERY ── subtaskid ──►◄
```

Purpose

Queries a specific SUBCOM subtask.

Operands

subtaskid
>   The identifier of the subtask to query.

Options

None

Usage Notes

This command displays information about all of the instances of the requested subtask. The output form is:

```
Instance C-block  ThreadID Userid    BytesIn   BytesOut
-------- -------  -------- ------    -------   --------
1        01EE0F5C 16       *         175       446
```

In this output, the columns have the following meanings:

Instance
>   The numeric identifier of the instance.

C-block
>   The address of the instance's C-block.

ThreadID
>   The CMS thread ID of the thread on which the instance is running.

Userid
>   The user ID of the client affiliated with the instance.

BytesIn
>   The number of bytes the client has provided to the instance.

BytesOut
>   The number of bytes the instance has provided to the client.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

SUBCOM START

```
▶▶─ SUBCOM ── START ── servicename ──┬──────────┬──▶◀
                                      └─ prefix ─┘
```
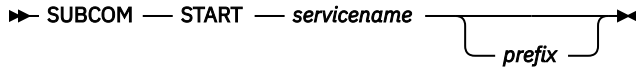
Purpose

Starts a service, connecting it to the SUBCOM line driver.

Operands

servicename
> The name of the service to start, as specified on a call to `ssServiceBind`.

prefix
> The prefix that will identify commands that should be sent to this service.

Options

None

Usage Notes

1. If *prefix* is not specified, the value of *servicename* is used for the prefix.

2. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

Messages and Return Codes

BKW0005E Out of storage.
BKW0200E Service not found.
BKW0205E Prefix already in use.
BKW0206E Service INIT routine failed - RC=&1 RE=&2.
BKW0207E Start of self is prohibited.

SUBCOM STOP

```
▶▶─ SUBCOM ── STOP ── subtaskid ──────────────▶◀
                              └─ NOW ─┘
```
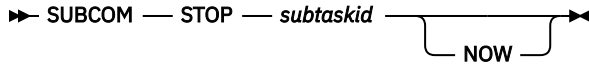
Purpose

Stops a specific SUBCOM subtask, optionally denying currently-connected clients the privilege of completing their operations.

Operands

subtaskid
> The identifier of the subtask to stop.

Options

NOW
> Stop the subtask without letting current clients complete normally.

Usage Notes

None

Messages and Return Codes

BKW0201E Subtask not found.
BKW0202E Stop of self is prohibited.
BKW0203I Subtask asked to STOP.
BKW0204I Subtask killed.

TCP LIST

```
▶▶── TCP ── LIST ──◀◀
```

Purpose

Lists the subtasks associated with the TCP/IP line driver.

Operands

None

Options

None

Usage Notes

This command displays information about the services started through the TCP/IP line driver. The output form is:

```
Subtask ServName BPort Adapter_Address TCPStack Sokts InUse Thrds
------- -------- ----- --------------- -------- ----- ----- -----
2       WEBSERV  80    0.0.0.0         TCPIP    100   17    31
4       WEBADMIN 90    9.117.32.29     TCPIP    50    4     13
```

The columns have the following meanings:

Subtask
    The numeric identifier of the subtask.

ServName
    The name of the started service.

BPort
    The port number to which the service is bound.

Adapter_Address
    The adapter address to which the port is bound.

TCPStack
    The user ID of the TCP/IP virtual machine through which this subtask's TCP activity is taking place.

Sokts
    The number of sockets available to the subtask.

InUse
    The number of sockets currently in use.

Thrds
    The number of CMS threads servicing this subtask.

Messages and Return Codes

BKW0201E Subtask not found.

TCP QUERY

```
►► TCP ── QUERY ── subtaskid ►◄
```

Purpose

Queries a specific TCP/IP subtask.

Operands

subtaskid
> The identifier of the subtask to query.

Options

None

Usage Notes

The output form is:

```
Instance C-Block  Userid      RPort Remote_Host      BytesIn    BytesOut
-------- -------  ------      ----- -----------      -------    --------
2        030F0210 PAUL        1401  9.130.79.171     165        32436
5        030F0500 FRED        833   9.117.32.29      8223       11234385
```

The columns and their meanings are:

Instance
> The numeric identifier of this instance.

C-Block
> The address of the instance's C-block.

Userid
> The mapped user ID of the client being served by this instance, as produced by the `ssUseridMap`.

RPort
> The port number through which the client's connection is exiting the client computer.

Remote_Host
> The IP address of the client computer.

BytesIn
> The number of bytes received from the client so far.

BytesOut
> The number of bytes sent to the client so far.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

TCP REPORT

```
►► TCP ── REPORT ──┬── ON ──┬──►◄
                   └── OFF ──┘
```
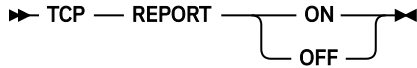
Purpose

Toggles reporting state for the TCP/IP line driver.

Operands

ON
 Turns reporting on.

OFF
 Turns reporting off.

Options

None

Usage Notes

When reporting is on, the TCP/IP line driver issues the following messages to describe client activity:

- BKW0500I
- BKW0501I
- BKW0502I
- BKW0504I

For more information, see "TCP and UDP Line Driver Messages" on page 397.

Messages and Return Codes

None

TCP START
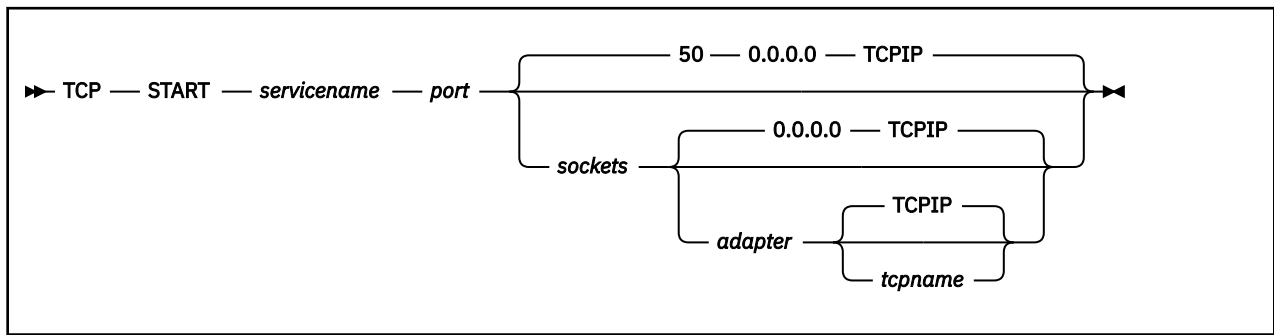
Purpose

Starts a service, connecting it to the TCP line driver.

Operands

servicename
> The name of the service to start, as specified on a call to `ssServiceBind`.

port
> The port number on which the reusable server kernel should make the service available.

sockets
> The number of sockets the reusable server kernel should make available for this port.

adapter
> The IP address of the adapter over which you want this service to accept requests (specify `0.0.0.0` to mean "any of this VM system's adapters").

tcpname
> The name of the TCP/IP service machine through which the reusable server kernel should access the TCP/IP network.

Options

None

Usage Notes

1. Operand *port* must be between 1 and 65535 inclusive.

2. Operand *sockets* must be between 50 and 2000 inclusive.

3. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

Messages and Return Codes

BKW0200E Service not found.
BKW0207E Start of self is prohibited.
BKW0513E Port number must be in range [0..65535].
BKW0514E Socket count must be in range [50..2000].
BKW0005E Out of storage.
BKW0516E Creation of subtask controller thread failed.
BKW0517E Creation of TCP/IP socket group failed.
BKW0518E Creation of listen socket failed.
BKW0519E Setting listen socket to SO_REUSEADDR failed.

BKW0520E Setting listen socket to nonblocking failed.
BKW0521E bind() for listen socket failed.
BKW0522E listen() for listen socket failed.

TCP STOP

```
▶▶─ TCP ── STOP ── subtaskid ──────────────▶◀
                          └─ NOW ─┘
```

Purpose

Stops a specific TCP/IP subtask, optionally denying currently-connected clients the privilege of completing their operations.

Operands

subtaskid
    The identifier of the subtask to stop.

Options

NOW
    Stop the subtask without letting current clients complete normally.

Usage Notes

None

Messages and Return Codes

BKW0201E Subtask not found.
BKW0523I Instance STOP requested.
BKW0524E Wait expired for STOP.

TRIE LIST

```
►► TRIE ── LIST ►◄
```

Purpose

Lists the tries created by this virtual machine.

Operands

None

Options

None

Usage Notes

The output form is:

```
Name      ASIT            LastFree NextFree Nodes     Records
----      ---------------- -------- -------- -----     -------
D0000001 7690F9000000001E 7FFFFFFF 0F4585B8 3050166   421008
D0000002 7690F88000000008 3FFFFFFF 2B934EEC 8697007   421008
```

The columns have the following meanings:

Name
   The trie name supplied by the creator.

ASIT
   The ASIT of the data space containing the trie.

LastFree
   The address of the last byte of the trie data space.

NextFree
   The address of the next free byte in the trie data space.

Nodes
   The number of nodes in the trie.

Records
   The number of record numbers being held onto by the trie.

Messages and Return Codes

BKW1900E No tries found.

UDP LIST

▶▶ ── UDP ── LIST ──▶◀

Purpose

Lists the subtasks associated with the UDP/IP line driver.

Operands

None

Options

None

Usage Notes

This command displays information about the services started through the UDP/IP line driver. The output form is:

```
Subtask ServName BPort Adapter_Address TCPStack InProg Thrds
------- -------- ----- --------------- -------- ------ -----
2       MYSERV   85    0.0.0.0         TCPIP    17     31
4       MYADMIN  95    9.117.32.29     TCPIP    4      13
```

The columns have the following meanings:

Subtask
    The numeric identifier of the subtask.

ServName
    The name of the started service.

BPort
    The port number to which the service is bound.

Adapter_Address
    The adapter address to which the port is bound.

TCPStack
    The user ID of the TCP/IP virtual machine through which this subtask's UDP activity is taking place.

InProg
    The number of transactions in progress at the moment.

Thrds
    The number of CMS threads servicing this subtask.

Messages and Return Codes

BKW0201E Subtask not found.

UDP QUERY

```
▶▶── UDP ── QUERY ── subtaskid ──▶◀
```

Purpose

Queries a specific UDP/IP subtask.

Operands

subtaskid
    The identifier of the subtask to query.

Options

None

Usage Notes

The output form is:

```
Instance C-Block  Userid    RPort Remote_Host     BytesIn    BytesOut
-------- -------  ------    ----- -----------     -------    --------
2        030F0210 PAUL       1401 9.130.79.171    165        0
5        030F0500 FRED        833 9.117.32.29     8223       0
```

The columns and their meanings are:

Instance
    The numeric identifier of this instance.

C-Block
    The address of the instance's C-block.

Userid
    The mapped user ID of the client being served by this instance, as produced by the `ssUseridMap`.

RPort
    The port number through which the client's connection is exiting the client computer.

Remote_Host
    The IP address of the client computer.

BytesIn
    The number of bytes received from the client so far.

BytesOut
    The number of bytes sent to the client so far.

Messages and Return Codes

BKW0201E Subtask not found.
BKW0208I Subtask is handling no clients.

UDP REPORT

```
▶▶ UDP ── REPORT ──┬── ON ──┬── ▶◀
                   └── OFF ──┘
```
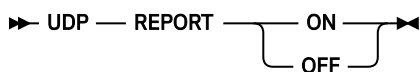
Purpose

Toggles reporting state for the UDP/IP line driver.

Operands

ON
    Turns reporting on.

OFF
    Turns reporting off.

Options

None

Usage Notes

When reporting is on, the UDP/IP line driver issues the following messages to describe client activity:

- BKW0500I
- BKW0501I
- BKW0502I
- BKW0504I

For more information, see "TCP and UDP Line Driver Messages" on page 397.

Messages and Return Codes

None

UDP START

```
▶▶ UDP ── START ── servicename ── port ──┬──────────────────────────────────────┬──▶◀
                                          │         0.0.0.0 ── TCPIP ──          │
                                          │                                      │
                                          │                ┌── TCPIP ──┐         │
                                          └── adapter ──────┤           ├─────────┘
                                                            └── tcpname ─┘
```
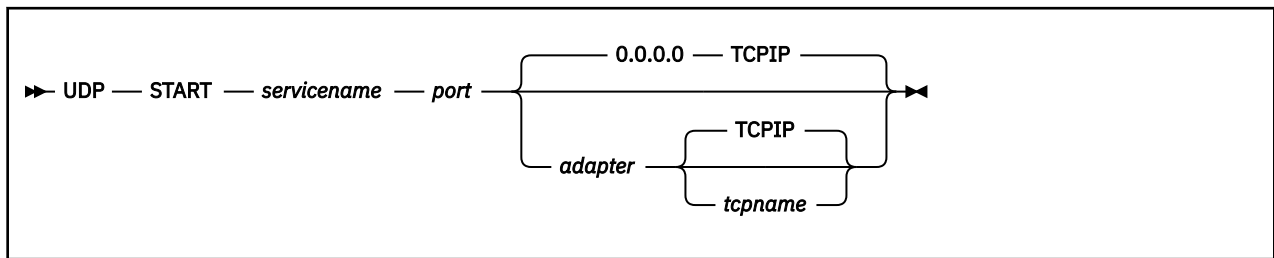
Purpose

Starts a service, connecting it to the UDP line driver.

Operands

servicename
  The name of the service to start, as specified on a call to `ssServiceBind`.

port
  The port number on which the reusable server kernel should make the service available.

adapter
  The IP address of the adapter over which you want this service to accept requests (specify `0.0.0.0` to mean "any of this VM system's adapters").

tcpname
  The name of the TCP/IP service machine through which the reusable server kernel should access the TCP/IP network.

Options

None

Usage Notes

1. Operand *port* must be between 1 and 65535 inclusive.

2. The started service is identified by a number called the *subtask ID*. Use this identifier to refer to the started service in future commands.

Messages and Return Codes

BKW0200E Service not found.
BKW0207E Start of self is prohibited.
BKW0513E Port number must be in range [0..65535].
BKW0514E Socket count must be in range [50..2000].
BKW0005E Out of storage.
BKW0516E Creation of subtask controller thread failed.
BKW0517E Creation of TCP/IP socket group failed.
BKW0518E Creation of listen socket failed.
BKW0519E Setting listen socket to SO_REUSEADDR failed.
BKW0520E Setting listen socket to nonblocking failed.
BKW0521E bind() for listen socket failed.
BKW0522E listen() for listen socket failed.

UDP STOP

```
▶▶── UDP ── STOP ── subtaskid ──────────────▶◀
                            └── NOW ──┘
```

Purpose

Stops a specific UDP/IP subtask, optionally denying currently-connected clients the privilege of completing their operations.

Operands

subtaskid
    The identifier of the subtask to stop.

Options

NOW
    Stop the subtask without letting current clients complete normally.

Usage Notes

None

Messages and Return Codes

BKW0201E Subtask not found.
BKW0523I Instance STOP requested.
BKW0524E Wait expired for STOP.

USERID MAP

```
►►── USERID ──── MAP ──── line_driver_name ──── node ──── user ─►◄
```

Purpose

Interrogates the user ID mapping file.

Operands

line_driver_name
>   The name of the line driver whose mapping is being interrogated.

node
>   The nodename as known to the specified line driver.

user
>   The user ID as known to the specified line driver.

Options

None

Usage Notes

The mapping is interrogated and the result displayed.

Messages and Return Codes

BKW0401I &1 &2 &3 maps to &4
BKW0402E RC=&1 RE=&2 mapping &3 &4 &5

USERID RELOAD

```
►►── USERID ──── RELOAD ──►◄
```

Purpose

Reloads the user ID mapping file.

Operands

None

Options

None

Usage Notes

The user ID mapping file is reloaded from whatever file is nominated by configuration parameter UMAP_FILE.

Messages and Return Codes

BKW0400E Reload failed - DMSOPEN or DMSREAD RC=&1 RE=&2.

WORKER ADD

```
►►─ WORKER ── ADD ── classname ── userid ──┬─────1─────┬── ►◄
                                           └─ capacity ─┘
```
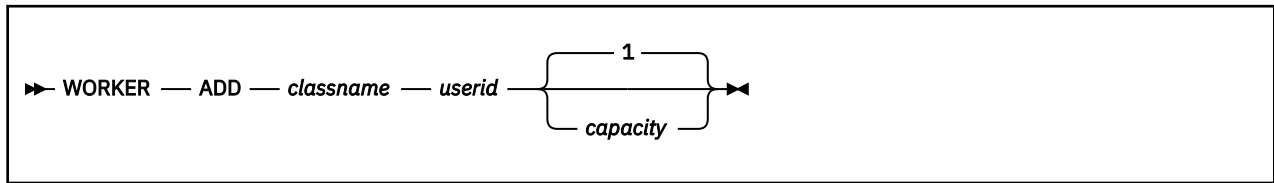
Purpose

Adds a worker machine to a worker class, creating the class if the class does not yet exist.

Operands

classname
> The name of the worker class to which the worker machine should be added.

userid
> The user ID of the worker virtual machine.

capacity
> The number of IUCV connections the worker machine is capable of handling concurrently.

Options

1
> The worker is capable of handling one connection at a time.

Usage Notes

1. Case is significant in class names.
2. Do not add a given worker virtual machine to more than one worker class. Unpredictable results will occur.

Messages and Return Codes

BKW1800E Worker machine is already in the specified class.

WORKER CLASSES

```
►► WORKER ── CLASSES ►◄
```

Purpose

Displays summary information about the worker classes.

Operands

None

Options

None

Usage Notes

The output format is:

```
Class    D Machines   C-Limit    C-InUse
-----    - --------   -------    -------
cgiserv  n 2          2          0
```

The columns have the following meanings:

Column
    Meaning

Class
    Name of class

D
    Whether workers are being managed as if they might be distributed on other nodes

    y
        Managed as if distributed

    n
        Managed as if local

Machines
    Number of worker machines

C-Limit
    Total number of connections permitted

C-InUse
    Number of connections at the moment

Messages and Return Codes

BKW1803E No worker classes defined.

WORKER DELCLASS

```
►►── WORKER ── DELCLASS ── classname ──────────►◄
                                    └─ FORCE ─┘
```

Purpose

Deletes a worker class, requesting instances to close their connections to the workers therein.

Operands

classname
    The name of the worker class being deleted.

FORCE
    The server kernel should forcibly sever the IUCV connections to the workers in the class.

Options

None

Usage Notes

1. Case is significant in class names.
2. If FORCE is not specified, the server kernel sends each instance a message asking it to end its connections with its workers in the affected class. Each instance is expected to finish up quickly and end its connection.
3. If FORCE is specified, the server kernel will IUCV SEVER all connections to workers in the class and inform each affected instance that its connections to those workers have been lost. After this, each worker machine found to be running disconnected will be forced off through CP FORCE.

Messages and Return Codes

BKW1802E Worker class not found.

WORKER DELETE

```
►►─── WORKER ──── DELETE ──── userid ─────────────────►◄
                                    └─── FORCE ───┘
```

Purpose

Deletes a single worker machine from its class.

Operands

userid
>   The user ID of the worker virtual machine.

FORCE
>   The server kernel should forcibly break any existing IUCV connections to the worker machine.

Options

None

Usage Notes

1. If FORCE is not specified, the server kernel sends each affected instance a message asking it to end its connections with the worker. The instances are expected to finish up quickly and end their connections to the worker.

2. If FORCE is specified, the server kernel will IUCV SEVER all connections to the worker and inform each affected instance that its connections to the worker have been lost. After this, if the worker is found to be running disconnected, it will be forced off through CP FORCE.

Messages and Return Codes

BKW1801E Worker machine not found.

WORKER DISTRIBUTE

```
►►── WORKER ── DISTRIBUTE ── classname ──┬── ON ──┬──►◄
                                         └── OFF ──┘
```

Purpose

Controls whether the reusable server kernel will attempt to manage a worker class as if the worker machines were located on other systems.

Operands

classname
　　The name of the worker class to which the command applies.

ON
　　Manage as if distributed.

OFF
　　Manage as if local.

Options

None

Usage Notes

1. Case is significant in class names.
2. When you set **DISTRIBUTE OFF** for a class, the reusable server kernel manages the workers as if they were running on the same instance of CP as the server itself. More specifically, the reusable server kernel uses the XAUTOLOG and FORCE commands to control the workers in the class. For example, if the server kernel determines that another worker needs to be logged on, it will issue XAUTOLOG to log on the new worker.
3. When you set **DISTRIBUTE ON** for a class, the reusable server kernel manages the workers as if they might be running on other systems. In particular, the reusable server kernel suppresses any attempts it might make to use the XAUTOLOG or FORCE commands to manage the worker machines in the class. Instead, responsibility for managing the machines is left to the server operator or system programmer.

Messages and Return Codes

BKW1802E Worker class not found.

WORKER MACHINES

```
▶▶─ WORKER ── MACHINES ── classname ─▶◀
```

Purpose

Displays a table of status information about worker machines in a given class.

Operands

classname
    The name of the class for which worker status should be displayed.

Options

None

Usage Notes

1. Case is significant in class names.

2. The output form is:

```
Machine  State    S Capacity   InUse
-------  -----    - --------   -----
MPT002   -        0 1          0
```

The columns have the following meanings:

Column
    Meaning

Machine
    The user ID of the worker machine

State
    What CP QUERY USER reports about the worker machine, or - if the worker is not logged on

S
    The status of the worker machine, as follows:

    0
        Seems usable

    1
        Repeated FORCE-XAUTOLOG cycles did not bring this worker to life

    2
        Tried to XAUTOLOG this worker but could not do so - possible insufficient privilege to use XAUTOLOG command

    3
        Unrecoverable error trying to IUCV CONNECT

    4
        Tried to reset worker through CP FORCE but command failed - possible insufficient privilege to use FORCE

    5
        CP FORCE succeeded but virtual machine did not log off - worker machine appears hung

Capacity
    The number of IUCV connections the worker can handle concurrently

InUse
> The number of IUCV connections to the worker right now

Messages and Return Codes

BKW1802E Worker class not found.
BKW1805E No worker machines found.

WORKER RESET

```
►►─ WORKER ── RESET ── classname ──────────────►◄
                            └─ userid ─┘
```

Purpose

Resets the status information the server kernel retains about a worker machine.

Operands

classname
    The name of the class to be reset.

userid
    The specific worker machine whose status is to be reset.

Options

None

Usage Notes

1. Case is significant in class names.
2. This command is meant to be used after manual intervention has supposedly resolved the problems the server kernel has detected in trying to use a worker machine or a class of worker machines. For example, the system administrator might have omitted the IUCV ALLOW statements in the workers' CP directory entries, and when the server attempted to use those workers, it found it could not connect to them. Once the CP directory has been repaired, WORKER RESET can be used to wipe out the server kernel's memory of the difficulty.
3. If *userid* is omitted, the status for all machines in the class is reset.

Messages and Return Codes

BKW1801E Worker machine not found.
BKW1802E Worker class not found.

WORKER STATUS

```
►►── WORKER ── STATUS ──┬──────────┬──►◄
                        └ classname ┘
```

Purpose

Displays information about the current set of connections to worker machines.

Operands

classname
   The name of the worker class for which status information should be displayed.

Options

None

Usage Notes

1. Case is significant in class names.
2. The output form is:

   ```
   Class    Machine  W-CBlock  I-CBlock  I-Service
   -----    -------  --------  --------  ---------
   cgiserv  MPT001   03FF3048  03FE21F8  HTTP
   ```

   The columns and their meanings are:

   Column
      Meaning
   Class
      The worker class involved
   Machine
      The worker machine to which the connection leads
   W-CBlock
      The address of the worker C-block
   I-CBlock
      The address of the instance C-block
   I-Service
      The service with which the instance is affiliated

Messages and Return Codes

BKW1802E Worker class not found.
BKW1804E No worker connections found.

Chapter 15. Function Descriptions

This chapter describes application programming interfaces (APIs) provided as part of the reusable server kernel. To review, the APIs can be partitioned into a number of subsets:

Table 45. Programming Interfaces

Subset	Description
Anchor	Provides a means for manipulating an anchor word.
Authorization	Provides a means for manipulating an authorization database.
Cache	Provides a means for manipulating cached files.
Client	Provides a means for manipulating buffers of client data.
Enroll	Provides a means for manipulating enrollment data.
Memory	Provides a means for manipulating memory.
Server	Provides a means for starting and stopping the server.
Service	Provides a means for identifying services.
Storage group	Provides a means for manipulating storage groups.
Tries	Provides a means for manipulating tries.
User ID	Provides a means for mapping user IDs.
Worker	Provides a means for connecting to a worker machine.

Programmers should be aware of the these restrictions regarding the use of these APIs:

- `RSKMAIN` can call only `ssServiceBind` and `ssServerRun`.
- `ssServiceBind` can be called only by `RSKMAIN` and only before `ssServerRun`.
- `ssServerRun` can be called only by `RSKMAIN` and only once.

Note: Failure to adhere to these restrictions could cause unpredictable results.

ssAnchorGet — Get Anchor Value

```
ssAnchorGet

    retcode
    reascode
    anchor
    monbufptr
    monbufsize
```

Purpose

Retrieves the value of the application-wide anchor word and the address and size of the application monitor data area.

Operands

ssAnchorGet
　　is the name of the function being invoked.

retcode
　　(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssAnchorGet`.

reascode
　　(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssAnchorGet`.

anchor
　　(output,INT,4) is a signed four-byte binary output variable to hold the returned anchor value.

monbufptr
　　(output,POINTER,4) is a signed four-byte binary output variable to hold the address of the application monitor buffer.

monbufsize
　　(output,INT,4) is a signed four-byte binary output variable to hold the size of the application monitor buffer.

Usage Notes

1. If the application-wide anchor word has not yet been set, this routine returns zero as the value of the anchor word.

2. The value returned in *monbufsize* is the value of the MON_USER_SIZE configuration variable.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_anc_rc_success	*ss_anc_re_success*	`ssAnchorGet` completed successfully

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMANC MACRO
PL/X	SSPLXANC COPY

ssAnchorSet — Set Anchor Value

```
ssAnchorSet

    retcode
    reascode
    anchor
```

Purpose

Sets the value of the application-wide anchor word.

Operands

ssAnchorSet
>   is the name of the function being invoked.

retcode
>   (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAnchorSet.

reascode
>   (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAnchorSet.

anchor
>   (input,INT,4) is a signed four-byte binary input variable holding the new anchor value.

Usage Notes

None

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_anc_rc_success	*ss_anc_re_success*	ssAnchorSet completed successfully

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMANC MACRO
PL/X	SSPLXANC COPY

ssAuthCreateClass — Create an Object Class

```
ssAuthCreateClass

    retcode
    reascode
    class_id
    operation_count
    operation_array
```

Purpose

Creates a class in the authorization rule base.

Operands

ssAuthCreateClass
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssAuthCreateClass`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssAuthCreateClass`.

class_id
    (input,CHAR,8) is a character string holding the identifier of the new class.

operation_count
    (input,INT,4) is a signed four-byte binary input variable holding the number of operations defined on the class.

operation_array
    (input,CHAR,4*operation_count) is an array of character strings holding the operations defined on the class.

Usage Notes

For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	`ssAuthCreateClass` completed successfully
ss_aut_rc_error	*ss_aut_re_bad_count*	*operation_count* out of range
ss_aut_rc_error	*ss_aut_re_out_of_storage*	Not enough storage available
ss_aut_rc_error	*ss_aut_re_exists*	Class already exists
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed

Return Code	Reason Code	Meaning
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthCreateObject — Create an Object

```
ssAuthCreateObject

    retcode
    reascode
    object_name
    object_name_length
    class_id
```

Purpose

Creates an object in the authorization rule base, assigning the object to the specified class.

Operands

ssAuthCreateObject
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssAuthCreateObject`.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssAuthCreateObject`.

object_name
> (input,CHAR,*object_name_length*) is a character string holding the name of the object.

object_name_length
> (input,INT,4) is a signed four-byte binary input variable holding the length of *object_name*.

class_id
> (input,INT,4) is a signed four-byte binary input variable holding the identifier of the class to which the object belongs.

Usage Notes

For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthCreateObject completed successfully
ss_aut_rc_error	*ss_aut_re_bad_obj_length*	*object_name_length* out of range
ss_aut_rc_error	*ss_aut_re_out_of_storage*	Not enough storage available
ss_aut_rc_error	*ss_aut_re_no_class*	Class does not exist
ss_aut_rc_error	*ss_aut_re_exists*	Object already exists
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed

Return Code	Reason Code	Meaning
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthDeleteClass — Delete a Class

```
ssAuthDeleteClass

    retcode
    reascode
    class_id
    option_count
    option_array
```

Purpose

Deletes the objects in a class, and optionally deletes the class.

Operands

ssAuthDeleteClass
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    ssAuthDeleteClass.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    ssAuthDeleteClass.

class_id
    (input,CHAR,8) is a character string holding the identifier of the class to be deleted.

option_count
    (input,INT,4) is a signed four-byte binary input variable holding the number of options in *option_array*.

option_array
    (input,INT,4*option_count) is an array of signed four-byte binary input variables holding the deletion
    options.

Usage Notes

1. These options are recognized:

    ss_aut_objects_only
        Delete only the class's objects

    ss_aut_objects_and_class
        Delete the class and the class's objects (default)

2. For more information on the naming conventions and other limits for the authorization API, see
    "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthDeleteClass completed successfully
ss_aut_rc_error	*ss_aut_re_bad_count*	*option_count* is out of range
ss_aut_rc_error	*ss_aut_re_bad_option*	At least one element of *option_array* is unrecognized

ssAuthDeleteClass

Return Code	Reason Code	Meaning
ss_aut_rc_error	*ss_aut_re_no_class*	Class does not exist
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthDeleteObject — Delete an Object

```
ssAuthDeleteObject

    retcode
    reascode
    object_name
    object_name_length
    option_count
    option_array
```

Purpose

Deletes the rules associated with an object, and optionally deletes the object.

Operands

ssAuthDeleteObject
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthDeleteObject.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthDeleteObject.

object_name
> (input,CHAR,*object_name_length*) is a character string holding the name of the object.

object_name_length
> (input,INT,4) is a signed four-byte binary input variable holding the length of *object_name*

option_count
> (input,INT,4) is a signed four-byte binary input variable holding the number of options in *option_array*.

option_array
> (input,INT,4*option_count) is an array of signed four-byte binary input variables holding the options to be applied to the deletion.

Usage Notes

1. These deletion options are recognized:

   ss_aut_rules_only
   > Delete only the object's rules

   ss_aut_rules_and_object
   > Delete the object and all its rules (default)

2. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthDeleteObject completed successfully

ssAuthDeleteObject

Return Code	Reason Code	Meaning
ss_aut_rc_error	*ss_aut_re_bad_obj_length*	*object_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_count*	*option_count* is out of range
ss_aut_rc_error	*ss_aut_re_bad_option*	Unrecognized option in *option_array*
ss_aut_rc_error	*ss_aut_re_no_object*	Object does not exist
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthDeleteUser — Delete a User

```
ssAuthDeleteUser

    retcode
    reascode
    user_name
    user_name_length
    class_name
    option_count
    option_array
```

Purpose

Deletes rules associated with a given user.

Operands

ssAuthDeleteUser
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthDeleteUser.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthDeleteUser.

user_name
> (input,CHAR,*user_name_length*) is a character string holding the name of the user.

user_name_length
> (input,INT,4) is a signed four-byte binary input variable holding the length of *user_name*

class_name
> (input,CHAR,8) is the name of the class from which rules should be deleted.

option_count
> (input,INT,4) is a signed four-byte binary input variable holding the number of deletion options specified.

option_array
> (input,INT,4*option_count*) is an array of signed four-byte binary input variables holding the deletion options.

Usage Notes

1. If no deletion options are specified, or if option *ss_aut_all_classes* is specified, then every rule applicable to the named user is deleted.

2. If *ss_aut_specific_class* is specified in the options array, then the only rules deleted are those that both apply to objects belonging to class *class_name* and mention the named user.

3. To adjust a given user's rules for a specific object, use routine *ssAuthPermitUser*.

4. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthDeleteUser completed successfully
ss_aut_rc_error	*ss_aut_re_bad_user_length*	*user_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_count*	*option_count* out of range
ss_aut_rc_error	*ss_aut_re_bad_option*	Unrecognized option in *option_array*
ss_aut_rc_error	*ss_aut_re_no_user*	No rules exist for *user_name*
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthListClasses — List Classes

```
ssAuthListClasses

    retcode
    reascode
    match_key
    match_key_length
    classes_expected
    class_buffer
    classes_returned
```

Purpose

Returns a list of classes.

Operands

ssAuthListClasses
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthListClasses.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthListClasses.

match_key
(input,CHAR,*match_key_length*) is an input character string holding the match key.

match_key_length
(input,INT,4) is a signed four-byte binary input variable holding the length of the match key.

classes_expected
(input,INT,4) is a signed four-byte binary input variable holding the number of eight-byte class names that will fit in *class_buffer*.

class_buffer
(output,CHAR,140*classes_expected*) is an output buffer into which the list of classes and their defined operations is to be placed.

classes_returned
(output,INT,4) is a signed four-byte binary output variable to hold the number of classes defined.

Usage Notes

1. ssAuthListClasses returns a list of the classes whose names match the match key specified by the caller. The operations defined on those classes are also returned.

2. The key expressed in *match_key* is expressed according to the CMS Application Multitasking syntax for IPC and event match keys.

3. Each class returned consumes 140 bytes in the output buffer, as follows:

   Offset.Length
     Usage

0.8
Class name

8.4
Number of operations

12.128
Operations (4 bytes each)

4. If the actual number of classes defined is greater than *classes_expected*, then the actual number of classes defined is returned in *classes_returned*, as many class names as will fit are filled into the output buffer, and a warning return and reason code are produced.

5. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthListClasses completed successfully
ss_aut_rc_error	*ss_aut_re_bad_count*	*classes_expected* is out of range
ss_aut_rc_warning	*ss_aut_re_too_many*	Some class names did not fit into the output buffer
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthListObjects — List Objects in Class

```
ssAuthListObjects

    retcode
    reascode
    class_id
    match_key
    match_key_length
    object_names_expected
    object_name_buffer_pointers
    object_name_buffer_sizes
    object_name_lengths
    object_names_returned
```

Purpose

Generates a list of the names of the objects belonging to a given class.

Operands

ssAuthListObjects
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthListObjects.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthListObjects.

class_id
    (input,CHAR,8) is a character string holding the class to be interrogated.

match_key
    (input,CHAR,*match_key_length*) is an input character string holding the match key.

match_key_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of the match key.

object_names_expected
    (input,INT,4) is a signed four-byte binary input variable holding the number of elements in the *object_name_buffer_pointers*, *object_name_buffer_sizes*, and *object_name_lengths* arrays.

object_name_buffer_pointers
    (input,POINTER,4**object_names_expected*) is an array of pointers to buffers to hold the returned object names.

object_name_buffer_sizes
    (input,INT,4**object_names_expected*) is an array of signed four-byte binary input variables holding the sizes of the buffers pointed to by the elements of *object_name_buffer_pointers*.

object_name_lengths
    (output,INT,4**object_names_expected*) is an array of signed four-byte binary output variables to hold the lengths of the returned object names.

object_names_returned
    (output,INT,4) is a signed four-byte binary output variable to hold the actual number of object names matching the supplied key.

Usage Notes

1. This function returns the names of the objects belonging to class *class_id* and matching key *match_key*.
2. The key expressed in *match_key* is expressed according to the CMS Application Multitasking syntax for IPC and event match keys.
3. If the actual number of objects selected by *match_key* is greater than *object_names_expected*, then the actual number of objects selected is returned in *object_names_returned*, as many object names as will fit are filled into the output arrays, and a warning return and reason code are produced.
4. If an object name does not fit into the buffer described by its pair of elements from the *object_name_buffer_pointers* and *object_name_buffer_sizes* arrays, then the actual length of the object name is returned in the corresponding element of the *object_name_lengths*, as much of the object name as will fit is returned in the object name buffer, and a warning return and reason code are produced.
5. If both of the above-mentioned warning conditions are encountered, the reason code will indicate that more object names were available than would fit in the output arrays (in other words, the truncated object name condition will not be visible through reason code).
6. For more information on the naming conventions and other limits for the authorization API, see .

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthListObjects completed successfully
ss_aut_rc_error	*ss_aut_re_bad_count*	*object_names_expected* out of range
ss_aut_rc_error	*ss_aut_re_no_class*	Class does not exist
ss_aut_rc_warning	*ss_aut_re_too_many*	More object names were available than caller expected
ss_aut_rc_warning	*ss_aut_re_trunc*	One or more returned object names was truncated
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthModifyClass — Modify an Object Class

```
ssAuthModifyClass

    retcode
    reascode
    class_id
    operation_count
    operation_array
```

Purpose

Adds operations to an existing object class.

Operands

ssAuthModifyClass
>   is the name of the function being invoked.

retcode
>   (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthModifyClass.

reascode
>   (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthModifyClass.

class_id
>   (input,CHAR,8) is a character string holding the identifier of the class being modified.

operation_count
>   (input,INT,4) is a signed four-byte binary input variable holding the number of operations to be added to the class.

operation_array
>   (input,CHAR,4*operation_count) is an array of character strings holding the operations to be added to the class.

Usage Notes

1. Use this function when it becomes necessary to define one or more new operations on a class (and therefore on all objects belonging to it).
2. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthModifyClass completed successfully
ss_aut_rc_error	*ss_aut_re_bad_count*	*operation_count* out of range
ss_aut_rc_error	*ss_aut_re_no_class*	Class does not exist
ss_aut_rc_error	*ss_aut_re_too_many*	Operation limit on class would be exceeded
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed

Return Code	Reason Code	Meaning
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthPermitUser — Permit a User

ssAuthPermitUser

> *retcode*
> *reascode*
> *user_name*
> *user_name_length*
> *object_name*
> *object_name_length*
> *use_arrays*
> *operation_count*
> *operation_array*
> *operation_qualifiers*
> *update_results*

Purpose

Installs, modifies, or deletes a rule in the rule base.

Operands

ssAuthPermitUser
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthPermitUser.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthPermitUser.

user_name
> (input,CHAR,*user_name_length*) is a character string holding the name of the user.

user_name_length
> (input,INT,4) is a signed four-byte binary input variable holding the length of *user_name*.

object_name
> (input,CHAR,*object_name_length*) is a character string holding the name of the object.

object_name_length
> (input,INT,4) is a signed four-byte binary input variable holding the length of *object_name*.

use_arrays
> (input,INT,4) is a signed four-byte binary input variable holding a flag indicating how the operation arrays should be applied to the rule.

operation_count
> (input,INT,4) is a signed four-byte binary input variable holding the length of the *operation_array*, *operation_qualifiers* and *update_results* arrays.

operation_array
> (input,CHAR,4*operation_count*) is an array of character strings holding the operations being edited.

operation_qualifiers
> (input,INT,4*operation_count*) is an array of signed four-byte binary input variables holding the interpretation rules for the corresponding elements of *operation_array*.

update_results
(output,INT,4*operation_count) is an array of signed four-byte binary output variables to hold the results of applying the changes requested in the corresponding elements of the *operation_array* and *operation_qualifier* arrays.

Usage Notes

1. These values are recognized in *use_arrays*:

ss_aut_add_all
First add all operations defined on the object to the user's rule for the object, then use the operation arrays to further update the user's rule

ss_aut_delete_all
First completely delete the current rule, then use the operation arrays to construct a new rule

ss_aut_use_arrays
Just update the current rule, using the operation arrays

2. These items are recognized in *operation_qualifiers*:

ss_aut_add_operation
Add the corresponding operation in *operation_array*

ss_aut_remove_operation
Remove the corresponding operation in *operation_array*

3. These items are filled into *update_results*:

ss_aut_op_not_defined
Operation is not defined on class to which object belongs

ss_aut_op_permitted
Operation is now permitted

ss_aut_op_not_permitted
Operation is now not permitted

ss_aut_no_change
Requested update did not change user's rule for object

4. To completely remove a rule, use *ss_aut_delete_all* and *operation_count*=0.

5. To grant "blanket" access to an object, use *ss_aut_add_all* and *operation_count*=0.

6. To grant all authorities except ones you explicitly wish to exclude, use *ss_aut_add_all* followed by an operation array naming the authorities you wish to exclude, each entry being qualified by *ss_aut_remove_operation*.

7. To "edit" an existing rule, use *ss_aut_use_arrays* and operation arrays containing the changes you wish to apply.

8. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthPermitUser completed successfully
ss_aut_rc_warning	*ss_aut_re_bad_op*	One or more of the elements of *operation_array* is not defined on this object's class
ss_aut_rc_error	*ss_aut_re_bad_user_length*	*user_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_obj_length*	*object_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_use*	*use_arrays* contains an unrecognized value

Return Code	Reason Code	Meaning
ss_aut_rc_error	*ss_aut_re_bad_count*	*operation_count* out of range
ss_aut_rc_error	*ss_aut_re_bad_qual*	One or more of the elements of *operation_qualifiers* is unrecognized
ss_aut_rc_error	*ss_aut_re_out_of_storage*	Not enough storage available
ss_aut_rc_error	*ss_aut_re_no_object*	Object does not exist
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthQueryObject — Query an Object

```
ssAuthQueryObject

    retcode
    reascode
    object_name
    object_name_length
    class_id
    userids_expected
    userid_buffer_pointers
    userid_buffer_sizes
    userid_lengths
    userids_returned
```

Purpose

Queries an object, returning the class to which it belongs and a list of the user IDs for which a rule exists for the object.

Operands

ssAuthQueryObject
  is the name of the function being invoked.

retcode
  (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthQueryObject.

reascode
  (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthQueryObject.

object_name
  (input,CHAR,*object_name_length*) is a character string holding the name of the object.

object_name_length
  (input,INT,4) is a signed four-byte binary input variable holding the length of *object_name*.

class_id
  (output,CHAR,8) is a character string to hold the class to which the object belongs.

userids_expected
  (input,INT,4) is a signed four-byte binary input variable holding the number of elements in the *userid_buffer_pointers*, *userid_buffer_sizes*, and *userid_lengths* arrays.

userid_buffer_pointers
  (input,POINTER,4*userids_expected*) is an array of pointers to buffers to hold the returned user IDs.

userid_buffer_sizes
  (input,INT,4*userids_expected*) is an array of signed four-byte binary input variables holding the sizes of the buffers pointed to by the elements of *userid_buffer_pointers*.

userid_lengths
  (output,INT,4*userids_expected*) is an array of signed four-byte binary output variables to hold the lengths of the returned user IDs.

userids_returned
  (output,INT,4) is a signed four-byte binary output variable to hold the actual number of user IDs for which a rule exists for the object.

Usage Notes

1. If the actual number of user IDs for which a rule exists is greater than *userids_expected*, then the actual number of user IDs is returned in *userids_returned*, as many user IDs as will fit are filled into the output arrays, and a warning return and reason code are produced.

2. If a user ID does not fit into the buffer described by the pair of elements from the *userid_buffer_pointers* and *userid_buffer_sizes* arrays, then the actual length of the user ID is returned in the corresponding element of the *userid_lengths* arrays, as much of the user ID as will fit is returned in the buffer, and a warning return and reason code are produced.

3. If both of the above-mentioned warning conditions are encountered, the reason code will indicate that more user IDs were available than would fit in the output arrays (in other words, the truncated user ID condition will not be visible through reason code).

4. To determine the specific access rights afforded to one of the returned user IDs, use `ssAuthQueryRule`.

5. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	`ssAuthQueryObject` completed successfully
ss_aut_rc_error	*ss_aut_re_bad_obj_length*	*object_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_count*	*userids_expected* out of range
ss_aut_rc_error	*ss_aut_re_no_object*	Object does not exist
ss_aut_rc_warning	*ss_aut_re_too_many*	Some user IDs did not fit into the output arrays
ss_aut_rc_warning	*ss_aut_re_trunc*	One or more returned user IDs was truncated
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthQueryRule — Query a Rule

---

ssAuthQueryRule

    retcode
    reascode
    user_name
    user_name_length
    object_name
    object_name_length
    operations_expected
    operation_array
    operations_returned

Purpose

Queries the operations a user can perform against an object.

Operands

ssAuthQueryRule
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    ssAuthQueryRule.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    ssAuthQueryRule.

user_name
    (input,CHAR,*user_name_length*) is a character string holding the name of the user.

user_name_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of *user_name*.

object_name
    (input,CHAR,*object_name_length*) is a character string holding the name of the object.

object_name_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of *object_name*.

operations_expected
    (input,INT,4) is a signed four-byte binary input variable holding the size of *operation_array*.

operation_array
    (output,CHAR,4**operations_expected*) is an array of character strings to hold the operations the user is
    permitted to perform.

operations_returned
    (output,INT,4) is a signed four-byte binary output variable to hold the number of operations filled into
    operation_array.

Usage Notes

1. If the actual number of operations permitted is greater than *operations_expected*, then the actual
   number of operations permitted is returned in *operations_returned*, as many operations as will fit are
   filled into *operation_array*, and a warning return and reason code are produced.

2. If the named user is not permitted any operations against the named object, then a successful return and reason code are generated and *operations_returned* is set to zero.

3. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthQueryRule completed successfully
ss_aut_rc_error	*ss_aut_re_bad_user_length*	*user_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_obj_length*	*object_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_count*	*operations_expected* out of range
ss_aut_rc_error	*ss_aut_re_no_object*	Object does not exist
ss_aut_rc_warning	*ss_aut_re_too_many*	Some operations did not fit into *operation_array*
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthReload — Reload Authorization Data

```
ssAuthReload

    retcode
    reascode
```

Purpose

Resets the internal authorization engine.

Operands

ssAuthReload
     is the name of the function being invoked.

retcode
     (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthReload.

reascode
     (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthReload.

Usage Notes

This function is intended for use when an I/O error of some kind shuts off the authorization API (causes *ss_aut_re_prev_io_error* to be returned). It performs these functions:

- Closes all authorization data files, ignoring close errors.

    Note: For the SFS, the work unit was rolled back at the time the error was detected. For other repositories, the log file and update algorithms provide appropriate recovery mechanisms.

- Returns its CMS work unit ID, if applicable.
- Flushes all caches.
- Gets a new CMS work unit ID, if applicable.
- Reopens the data files.
- If applicable, attempts to recover the authorization database (processes log file and realigns the two copies).
- Reloads the authorization index into storage.

If all these operations were successful, the authorization API is again available for use.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthReload completed successfully
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvw_fail*	Condition variable wait failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_gwu_fail*	DMSGETWU (Get Work Unit ID) failed

Return Code	Reason Code	Meaning
ss_aut_rc_error	*ss_aut_re_open_fail*	Unable to open authorization files
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_write_fail*	Unable to write authorization files
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssAuthTestOperations — Test Operations

```
ssAuthTestOperations

    retcode
    reascode
    user_name
    user_name_length
    object_name
    object_name_length
    operation_count
    desired_operations
    test_results
```

Purpose

Tests a given user's rights to perform a set of actions against a given object.

Operands

ssAuthTestOperations
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from ssAuthTestOperations.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssAuthTestOperations.

user_name
(input,CHAR,*user_name_length*) is a character string holding the name of the user.

user_name_length
(input,INT,4) is a signed four-byte binary input variable holding the length of *us/r_name*.

object_name
(input,CHAR,*object_name_length*) is a character string holding the name of the object.

object_name_length
(input,INT,4) is a signed four-byte binary input variable holding the length of *object_name*.

operation_count
(input,INT,4) is a signed four-byte binary input variable holding the length of the *desired_operations* and *test_results* arrays.

desired_operations
(input,CHAR,4*operation_count*) is an array of character strings holding the operations to be tested.

test_results
(output,INT,4*operation_count*) is an array of signed four-byte binary output variables to hold the results of the tests.

Usage Notes

1. On successful completion, each element of *test_results* will contain one of these values:

   ss_aut_op_permitted
   Operation is permitted

ss_aut_op_not_permitted
>   Operation is not permitted

ss_aut_op_not_defined
>   Operation is not defined

2. For more information on the naming conventions and other limits for the authorization API, see "Naming Conventions and Other Limits" on page 36.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_aut_rc_success	*ss_aut_re_success*	ssAuthTestOperations completed successfully
ss_aut_rc_error	*ss_aut_re_bad_user_length*	*user_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_obj_length*	*object_name_length* out of range
ss_aut_rc_error	*ss_aut_re_bad_count*	*operation_count* out of range
ss_aut_rc_error	*ss_aut_re_no_object*	Object does not exist
ss_aut_rc_error	*ss_aut_re_maq_fail*	Mutex acquisition failed
ss_aut_rc_error	*ss_aut_re_cvs_fail*	Condition variable signal failed
ss_aut_rc_error	*ss_aut_re_mr_fail*	Mutex release failed
ss_aut_rc_error	*ss_aut_re_read_fail*	Unable to read authorization files
ss_aut_rc_error	*ss_aut_re_prev_io_error*	API disabled due to I/O error on previous call
ss_aut_rc_error	*ss_aut_re_prev_sync_error*	API disabled due to synchronization error on previous call

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMAUT MACRO
PL/X	SSPLXAUT COPY

ssCacheCreate — Create Cache

```
ssCacheCreate

    retcode
    reascode
    cache_name
    cache_size
    cache_alet
```

Purpose

Creates a file cache, using a VM Data Space.

Operands

ssCacheCreate
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    `ssCacheCreate`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    `ssCacheCreate`.

cache_name
    (input,CHAR,8) is a character string holding the name of the new file cache.

cache_size
    (input,INT,4) is a signed four-byte binary input variable holding the size of the new file cache.

cache_ALET
    (output,INT,4) is a signed four-byte binary output variable to hold the returned ALET.

Usage Notes

1. The cache name is used directly in a call to `ssMemoryCreateDS` and therefore must not conflict with
   any other subpool names.

2. The cache size is to be given in pages. It must be greater than 0 and less than or equal to 524288. The
   actual size of the created cache is rounded up to the next 16-page boundary.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_fil_rc_success	*ss_fil_re_success*	`ssCacheCreate` completed successfully
ss_fil_rc_error	*ss_fil_re_bad_size*	*cache_size* is out of range
ss_fil_rc_error	*ss_fil_re_cache_exists*	Cache already exists
ss_fil_rc_error	*ss_fil_re_out_of_storage*	Out of storage
ss_fil_rc_error	*ss_fil_re_dscr_fail*	Creation of data space failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCAC MACRO
PL/X	SSPLXCAC COPY

ssCacheDelete — Delete Cache

---

ssCacheDelete

    retcode
    reascode
    cache_name

Purpose

Deletes a file cache.

Operands

ssCacheDelete
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    `ssCacheDelete`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    `ssCacheDelete`.

cache_name
    (input,CHAR,8) is a character string holding the name of the file cache to be deleted.

Usage Notes

1. Once deletion starts, the server kernel will not honor any more calls to `ssCacheFileOpen` for this cache.

2. The deletion does not complete until the last open file in this cache is closed.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_fil_rc_success	*ss_fil_re_success*	`ssCacheDelete` completed successfully
ss_fil_rc_error	*ss_fil_re_cache_not_found*	Cache not found

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCAC MACRO
PL/X	SSPLXCAC COPY

ssCacheFileClose — Close Cached File

```
ssCacheFileClose

    retcode
    reascode
    cache_name
    file_token
```

Purpose

Close a cached file.

Operands

ssCacheFileClose
  is the name of the function being invoked.

retcode
  (output,INT,4) is a signed four-byte binary output variable to hold the return code from
  `ssCacheFileClose`.

reascode
  (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
  `ssCacheFileClose`.

cache_name
  (input,CHAR,8) is a character string holding the name of the cache in which the file being closed is
  located.

file_token
  (input,CHAR,8) is a character string holding the token of the file being closed.

Usage Notes

If the file being closed was previously marked as stale, it is dropped from the cache.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_fil_rc_success	*ss_fil_re_success*	`ssCacheFileClose` completed successfully
ss_fil_rc_error	*ss_fil_re_cache_not_found*	Cache does not exist
ss_fil_rc_error	*ss_fil_re_bad_token*	File token is bad

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCAC MACRO
PL/X	SSPLXCAC COPY

ssCacheFileOpen — Open Cached File

```
ssCacheFileOpen

    retcode
    reascode
    cache_name
    file_name
    file_name_length
    ESM_data
    ESM_data_length
    flag_count
    flag_names
    flag_values
    file_token
    cache_ALET
    file_address
    file_size
    file_stamp
```

Purpose

Makes a file ready for reading from a cache, loading it from minidisk, SFS, or BFS if necessary.

Operands

ssCacheFileOpen
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from ssCacheFileOpen.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssCacheFileOpen.

cache_name
(input,CHAR,8) is a character string holding the name of the cache in which the file is to be placed.

file_name
(input,CHAR,*file_name_length*) is a character string holding the name of the file to be cached.

file_name_length
(input,INT,4) is a signed four-byte binary input variable holding the length of *file_name*.

ESM_data
(input,CHAR,*ESM_data_length*) is a character string holding ESM data to be passed to DMSOPEN.

ESM_data_length
(input,INT,4) is a signed four-byte binary input variable holding the length of *ESM_data*.

flag_count
(input,INT,4) is a signed four-byte binary input variable holding the number of elements in each of the the *flag_names* and *flag_values* arrays.

flag_names
(input,INT,4*flag_count*) is an array of signed four-byte binary input variables holding flag names.

flag_values
    (input,INT,4*flag_count) is an array of signed four-byte binary input variables holding flag values.

file_token
    (output,CHAR,8) is a character string to hold the returned file token.

cache_ALET
    (output,INT,4) is a signed four-byte binary output variable to hold the ALET of the cache data space.

file_address
    (output,POINTER,4) is a signed four-byte binary output variable to hold the address of the file in the data space.

file_size
    (output,INT,4) is a signed four-byte binary output variable to hold the size of the cached file in bytes.

file_stamp
    (output,CHAR,32) is a character string to hold the returned last update date and time of the file.

Usage Notes

1. Parameters *file_name* and *file_name_length* together describe a string which will be passed unchanged to either CSL routine DMSOPEN or CSL routine BPX1OPN as the name of the file to be opened. The CSL routine the server kernel chooses depends on the values you specify in the flag arrays. Be aware that case is significant in file names.

2. The server kernel will pass parameters *ESM_data* and *ESM_data_length* unchanged to DMSOPEN if it ends up calling DMSOPEN to find the file. The server kernel will ignore the ESM data if it ends up calling BPX1OPN.

3. Parameter arrays *flag_names* and *flag_values* together contain integers specifying various controls on how the file is to be cached. These integers and their meanings are described in Table 46 on page 249.

Table 46. Flags for ssCacheFileOpen			
Flag Name	**Function**	**Acceptable Values**	**Default Value**
ss_cac_ofn_bfs	Corresponding value tells the server kernel whether to use BPX1OPN to open the file.	Specify *ss_cac_ofv_yes* for BPX1OPN or *ss_cac_ofv_no* for DMSOPEN.	If you do not mention this flag in your flag arrays, the server kernel will try to guess whether to use DMSOPEN or BPX1OPN based on the composition of the filename string you supply. If the filename you supply contains a blank (X'40'), the server kernel will try DMSOPEN. If it contains no blanks, the server kernel will try BPX1OPN.
ss_cac_ofn_xlate	Corresponding value nominates a translation table previously identified through ssCacheXlTabSet.	Any table ID, or zero to bypass translations.	Zero
ss_cac_ofn_preserve_dolr	Corresponding value specifies whether the file's date of last reference should be preserved (that is, not updated). *Ignored if the server kernel ends up calling BPX1OPN.*	Specify *ss_cac_ofv_yes* or *ss_cac_ofv_no*.	*ss_cac_ofv_no*

Table 46. Flags for ssCacheFileOpen (continued)

Flag Name	Function	Acceptable Values	Default Value
ss_cac_ofn_recmethod_fs	Corresponding value describes how the server kernel should expect the records to be delimited in the file it is reading from disk.	• **X'00xxxxxx'** - The file's records are delimited according to the structure recorded by the CMS file system (F1 for BFS files). • **X'01nnssss'** - The file's records are delimited by an *nn*-byte suffix appearing in the file's data after each record. Set *nn* equal to X'00', X'01', or X'02'. The suffix bytes to be used are *ssss*. If *nn* is X'01' the second suffix byte is ignored. • **X'02nnxxxx'** - The file's records are delimited by an *nn*-byte length prefix appearing in the file's data before each record. The length prefix does not include the length of the prefix itself. Set *nn* equal to X'02' or X'04'.	X'00000000'
ss_cac_ofn_recmethod_cache	Corresponding value describes how the server kernel should delimit records in the cached file.	• **X'01nnssss'** - Put an *nn*-byte suffix on each record. Set *nn* equal to X'00', X'01', or X'02'. The suffix bytes to be used are *ssss*. If *nn* is X'01' the second suffix byte is ignored. • **X'02nnxxxx'** - Prefix each record with a *nn*-byte length field. The length prefix does not include the length of the prefix itself. Set *nn* equal to X'02' or X'04'.	X'01000000'

4. Use the value supplied in output *file_token* in calls to `ssCacheFileRead` and `ssCacheFileClose`.

5. If the server kernel was able to load the file contiguously in data space storage, then it returns the cache's ALET in *cache_ALET* and the address of the file buffer in *file_address*. This lets the server know that it can use AR mode to access the file data directly if it chooses. If the file was not loaded contiguously, *cache_ALET* and *file_address* are returned as zero.

6. The number of bytes cached -- that is, the size of the transformed file, in bytes -- is returned in *file_size*.

7. If the data space is too full to contain the file, the server kernel throws away cached files in LRU fashion, skipping those files that are still open, until enough storage is freed to hold the new file. If the server kernel removes all files eligible for removal but the new file still will not fit, an error is returned.

8. If there are stale versions of the new file still in the cache, and those stale versions are no longer open, they are discarded prior to loading the new file. Stale, still-open versions are marked as stale and thrown out when they are finally closed.

9. A file's date of last reference is never updated on a cache hit, no matter what the caller requested.

10. Cache contents are indexed by file name as passed by the caller. Depending on accessed file modes, default filepools, SFS aliasing, and default filespaces, several different file names might actually refer to the same physical file; the server kernel cannot discern that these names all refer to the same file. Callers need to be aware of this phenomenon and might need to perform some file name resolution prior to calling ssCacheFileOpen in order to keep unnecessary duplicates out of a file cache.

   Similarly, if the server is referring to files using file mode letters and is switching the accessed file mode set through the ACCESS and RELEASE commands, the same name might refer to two different files at two different moments in time. The cache will be unharmed by this as long as those two different files have different update timestamps, but if two such files have the same update timestamp the cache will fail to reload when a reload truly is required. The server author is responsible for avoiding this situation.

11. Files with record formats other than V or F (as returned by DMSEXIST) cannot be cached.

12. Files with names longer than 256 bytes cannot be cached.

13. If you requested suffixing or prefixing for *ss_cac_ofn_recmethod_fs*, the records encountered in the file must all be less than or equal to 65,535 bytes in length.

14. On VM/ESA 2.3.0 and later, *file_stamp* is always returned in ISO format. On earlier VM/ESA releases, if the cached file was loaded from SFS or minidisk the stamp is returned in ISO format, but if the cached file was loaded from BFS the first four bytes of the returned stamp are Posix time and the remainder of the stamp is blank (X'40').

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_fil_rc_success	*ss_fil_re_success*	ssCacheFileOpen completed successfully
ss_fil_rc_error	*ss_fil_re_cache_not_found*	Cache does not exist
ss_fil_rc_error	*ss_fil_re_bad_length*	Bad value in *file_name_length*
ss_fil_rc_error	*ss_fil_re_bad_count*	Bad value in *flag_count*
ss_fil_rc_error	*ss_fil_re_bad_esmdl*	Bad value in *ESM_data_length*
ss_fil_rc_error	*ss_fil_re_bad_fname*	Bad value in *flag_names*
ss_fil_rc_error	*ss_fil_re_bad_fval*	Bad value in *flag_values*
ss_fil_rc_error	*ss_fil_re_exist_fail*	Call to DMSEXIST failed
ss_fil_rc_error	*ss_fil_re_file_not_found*	DMSOPEN could not find file
ss_fil_rc_error	*ss_fil_re_bad_recfm*	Record format is neither F nor V

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCAC MACRO
PL/X	SSPLXCAC COPY

ssCacheFileRead — Read Cached File

---

ssCacheFileRead

    retcode
    reascode
    cache_name
    file_token
    byte_offset
    byte_count
    buffer
    bytes_read

Purpose

Reads data from a cached file.

Operands

ssCacheFileRead
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    ssCacheFileRead.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    ssCacheFileRead.

cache_name
    (input,CHAR,8) is a character string holding the name of the cache in which the file is located.

file_token
    (input,CHAR,8) is a character string holding the token of the file to be read.

byte_offset
    (input,INT,4) is the zero-origin offset to the first byte of the file to be read.

byte_count
    (input,INT,4) is the number of bytes to be read.

buffer
    (output,CHAR,*byte_count*) is a character string to hold the bytes read from the file.

bytes_returned
    (output,INT,4) is a signed four-byte binary output variable to hold the number of bytes read from the
    file.

Usage Notes

1. The server kernel supports multiple simultaneous read operations against a given file.

2. If not enough bytes are available to satisfy the call, as many bytes as are available are returned in the
   output buffer and success is returned.

3. If the supplied offset is less than zero or is past the end of the file, an error is returned.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_fil_rc_success	*ss_fil_re_success*	ssCacheFileRead completed successfully
ss_fil_rc_error	*ss_fil_re_cache_not_found*	Cache does not exist
ss_fil_rc_error	*ss_fil_re_bad_token*	Bad file token
ss_fil_rc_error	*ss_fil_re_bad_offset*	Bad file offset
ss_fil_rc_error	*ss_fil_re_bad_length*	Bad byte count

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCAC MACRO
PL/X	SSPLXCAC COPY

ssCacheQuery — Query Cache

```
ssCacheQuery

    retcode
    reascode
    cache_name
    files_cached
    cache_size
    in_use
    open_count
    hit_count
```

Purpose

Returns basic statistics about a cache's operation.

Operands

ssCacheQuery
　　is the name of the function being invoked.

retcode
　　(output,INT,4) is a signed four-byte binary output variable to hold the return code from
　　`ssCacheQuery`.

reascode
　　(output,INT,4) is a signed four-byte binary output variable to hold the reason code from
　　`ssCacheQuery`.

cache_name
　　(input,CHAR,8) is a character string holding the name of the file cache to be queried.

files_cached
　　(output,INT,4) is a signed four-byte binary output variable to hold the number of files currently
　　resident in the cache.

cache_size
　　(output,INT,4) is a signed four-byte binary output variable to hold the size of the cache.

in_use
　　(output,INT,4) is a signed four-byte binary output variable to hold the amount of cache space currently
　　in use.

open_count
　　(output,INT,4) is a signed four-byte binary output variable to hold the number of file opens processed
　　through this cache.

hit_count
　　(output,INT,4) is a signed four-byte binary output variable to hold the number of times a file open was
　　satisfied without having to call CMS to read the file from disk.

Usage Notes

Parameters *cache_size* and *in_use* are returned in bytes.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_fil_rc_success	*ss_fil_re_success*	ssCacheQuery completed successfully
ss_fil_rc_error	*ss_fil_re_cache_not_found*	Cache not found

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCAC MACRO
PL/X	SSPLXCAC COPY

ssCacheXlTabSet — Set Translation Table

```
ssCacheXlTabSet

    retcode
    reascode
    table_id
    table
```

Purpose

Sets translation table for use when reading files.

Operands

ssCacheXlTabSet
　　is the name of the function being invoked.

retcode
　　(output,INT,4) is a signed four-byte binary output variable to hold the return code from
　　`ssCacheXlTabSet`.

reascode
　　(output,INT,4) is a signed four-byte binary output variable to hold the reason code from
　　`ssCacheXlTabSet`.

table_id
　　(input,INT,4) is a signed four-byte binary input variable holding the identifier of the new translation
　　table.

table
　　(input,CHAR,256) is a character string holding the translation table itself.

Usage Notes

1. Parameter *table_id* can be any four-byte integer except zero.
2. If *table_id* was previously in use, the previous table is replaced and a warning is returned.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_fil_rc_success	*ss_fil_re_success*	`ssCacheXlTabSet` completed successfully
ss_fil_rc_warning	*ss_fil_re_table_replaced*	Table was replaced
ss_fil_rc_error	*ss_fil_re_bad_table_id*	Table ID cannot be zero
ss_fil_rc_error	*ss_fil_re_out_of_storage*	Out of storage

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCAC MACRO

Language	Language Binding File
PL/X	SSPLXCAC COPY

ssClientDataGet — Get Client Data

```
ssClientDataGet

    retcode
    reascode
    caller_type
    C-block_address
    get_method
    buffer_alet
    data_buffer
    amount_wanted
    amount_given
    amount_remaining
```

Purpose

Obtains or discards data from client data buffers.

Operands

ssClientDataGet
  is the name of the function being invoked.

retcode
  (output,INT,4) is a signed four-byte binary output variable to hold the return code from
  `ssClientDataGet`.

reascode
  (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
  `ssClientDataGet`.

caller_type
  (input,INT,4) is a signed four-byte binary input variable holding an indicator of the kind of caller
  (instance or line driver).

C-block_address
  (input,POINTER,4) is a signed four-byte binary input variable holding the address of the C-block for
  the client in question.

get_method
  (input,INT,4) is a signed four-byte binary input variable holding an indicator of the kind of retrieval
  operation to be performed.

buffer_alet
  (input,INT,4) is a signed four-byte binary input variable holding the ALET to be used when accessing
  data_buffer.

data_buffer
  (input,CHAR,*amount_wanted*) is a character string into which the retrieved data is to be placed.

amount_wanted
  (input,INT,4) is a signed four-byte binary input variable holding the number of bytes of data to be
  retrieved or discarded.

amount_given
  (output,INT,4) is a signed four-byte binary output variable to hold the number of bytes actually
  returned or discarded.

amount_remaining
>   (output,INT,4) is a signed four-byte binary output variable to hold the number of bytes remaining in the client's buffers after the caller's operation completed.

Usage Notes

1. The *caller_type* should be set to one of these values:

   ss_cli_iam_instance
   >   The caller is an instance thread.

   ss_cli_iam_linedriver
   >   The caller is a line driver.

2. The *get_method* should be set to one of these values:

   ss_cli_method_peek
   >   Fill the caller's buffer but do not dequeue and discard it just yet from the reusable server kernel's internal buffers.

   ss_cli_method_read
   >   Fill the caller's buffer and dequeue and discard it from the reusable server kernel's internal buffers.

   ss_cli_method_discard
   >   Dequeue and discard the data from the reusable server kernel's internal buffers but do not fill it into the caller's buffer.

3. Setting *amount_wanted* to -1 means "perform this operation on all of the data currently buffered".

4. If the caller asks for more data than is currently buffered, all of the currently available data is returned, *amount_given* is filled in appropriately, and no error is returned.

5. If the line driver you are using is record-oriented, then the data stream you read from the client will be organized into records, each record prefixed by a four-byte length. For more information on the description of record-oriented line drivers, see .

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_cli_rc_success	*ss_cli_re_success*	`ssClientDataGet` completed successfully
ss_cli_rc_error	*ss_cli_re_bad_iam*	*caller_type* contains unrecognized value
ss_cli_rc_error	*ss_cli_re_bad_method*	*get_method* contains unrecognized value
ss_cli_rc_error	*ss_cli_re_out_of_range*	*amount_wanted* contains illegal value

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCLI MACRO
PL/X	SSPLXCLI COPY

ssClientDataInit — Initialize Client Data Buffers

```
ssClientDataInit

    retcode
    reascode
    C-block_address
    subpool_name
```

Purpose

Initializes client data buffer structures.

Operands

ssClientDataInit
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    `ssClientDataInit`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    `ssClientDataInit`.

C-block_address
    (input,POINTER,4) is a signed four-byte binary input variable holding the address of the C-block for
    the client in question.

subpool_name
    (input,CHARACTER,8) is a character string holding the name of the subpool from which these client
    buffers should be allocated.

Usage Notes

1. This routine is meant for use by a line driver that is preparing to handle a new client. As part of
   initializing the C-block that describes the new client, the line driver should call `ssClientDataInit` to
   ensure that the structures relating to buffering the client's data are initialized.

2. Subpool *subpool_name* must **not** be a subpool that refers to a VM Data Space.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_cli_rc_success	*ss_cli_re_success*	`ssClientDataInit` completed successfully

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCLI MACRO
PL/X	SSPLXCLI COPY

ssClientDataPut — Put Client Data

```
ssClientDataPut

    retcode
    reascode
    caller_type
    C-block_address
    buffer_alet
    data_buffer
    amount_of_data
    new_amount_buffered
```

Purpose

Writes data to client data buffers.

Operands

ssClientDataPut
  is the name of the function being invoked.

retcode
  (output,INT,4) is a signed four-byte binary output variable to hold the return code from
  `ssClientDataPut`.

reascode
  (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
  `ssClientDataPut`.

caller_type
  (input,INT,4) is a signed four-byte binary input variable holding an indicator of the kind of caller
  (instance or line driver).

C-block_address
  (input,POINTER,4) is a signed four-byte binary input variable holding the address of the C-block for
  the client in question.

buffer_alet
  (input,INT,4) is a signed four-byte binary input variable holding the ALET to be used when accessing
  data_buffer.

data_buffer
  (input,CHAR,*amount_of_data*) is a character string containing the data to be written.

amount_of_data
  (input,INT,4) is a signed four-byte binary input variable holding the length of *data_buffer*.

new_amount_buffered
  (output,INT,4) is a signed four-byte binary output variable to hold the new amount of data in the client
  buffer.

Usage Notes

1. *caller_type* should be set to one of these values:

   ss_cli_iam_instance
     The caller is an instance thread.

_ss_cli_iam_linedriver_
The caller is a line driver.

2. `ssClientDataPut` maintains the *bytes in* and *bytes out* fields of the C-block. A line driver should not attempt to maintain these itself.

3. `ssClientDataPut` exerts flow control on its caller. When the caller's operation results in either more than 16 MB being queued for the client or more than 128 distinct buffers being queued for the client, `ssClientDataPut` waits until the corresponding line driver empties the buffers before returning to the caller. The buffer will be emptied only if the server has sent the appropriate IPC message to its line driver; `ssClientDataPut` does not send any IPC messages on behalf of its caller.

4. If the line driver you are using is record-oriented, then the data stream you build for the client must be organized into records, each record prefixed by a four-byte length. For more information on the description of record-oriented line drivers, see Table 8 on page 12.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_cli_rc_success	*ss_cli_re_success*	`ssClientDataPut` completed successfully
ss_cli_rc_error	*ss_cli_re_bad_iam*	*caller_type* contains unrecognized value
ss_cli_rc_error	*ss_cli_re_out_of_range*	*amount_of_data* contains illegal value
ss_cli_rc_error	*ss_cli_re_out_of_storage*	Not enough free storage to buffer this data

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCLI MACRO
PL/X	SSPLXCLI COPY

ssClientDataTerm — Terminate Client Data Buffers

```
ssClientDataTerm

    retcode
    reascode
    C-block_address
```

Purpose

Terminates client data buffer structures.

Operands

ssClientDataTerm
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from
`ssClientDataTerm`.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from
`ssClientDataTerm`.

C-block_address
(input,POINTER,4) is a signed four-byte binary input variable holding the address of the C-block for
the client in question.

Usage Notes

This routine is meant for use by a line driver that is ending its handling of a client. As part of its
termination processing, the line driver should call `ssClientDataTerm` so that the reusable server kernel
can clean up its handling of buffered client data.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_cli_rc_success	*ss_cli_re_success*	`ssClientDataTerm` completed successfully

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMCLI MACRO
PL/X	SSPLXCLI COPY

ssEnrollCommit — Commit Enrollment Set

```
ssEnrollCommit

    retcode
    reascode
    set_name
```

Purpose

Commits changes to an open enrollment set.

Operands

ssEnrollCommit
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssEnrollCommit.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssEnrollCommit.

set_name
    (input,CHAR,8) is a character string holding the name of the enrollment set to be committed.

Usage Notes

1. This entry point commits the SFS file holding the named enrollment set. The enrollment set remains loaded and available for other transactions.

2. If the commit fails, the appropriate action is to call ssEnrollDrop to drop the set, using drop type *ss_enr_drop_rollback*.

3. An attempt to commit a transient enrollment set will return a warning. No other action is taken.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	ssEnrollCommit completed successfully
ss_enr_rc_error	*ss_enr_re_db_not_found*	Named enrollment set not found
ss_enr_rc_warning	*ss_enr_re_not_disk*	Named enrollment set is transient
ss_enr_rc_error	*ss_enr_re_comm_fail*	Call to DMSCOMM failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO

Language	Language Binding File
PL/X	SSPLXENR COPY

ssEnrollDrop — Drop Enrollment Set

```
ssEnrollDrop

    retcode
    reascode
    set_name
    drop_type
```

Purpose

Drops (closes, unloads) an enrollment set.

Operands

ssEnrollDrop
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssEnrollDrop`.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssEnrollDrop`.

set_name
> (input,CHAR,8) is a character string holding the name of the enrollment set to be dropped.

drop_type
> (input,INT,4) is a signed four-byte binary input variable holding a value indicative of the kind of drop to be performed:
>
> **ss_enr_drop_commit**
> > Commit changes
>
> **ss_enr_drop_rollback**
> > Roll back changes

Usage Notes

1. This entry point closes the SFS file holding the named enrollment set, either rolling back or committing the changes, according to the value of parameter *drop_type*. It also deletes the data space and performs other cleanup operations.

2. If *ss_enr_drop_commit* is requested and the commit fails, an error will be returned and no other action will be taken. The appropriate recovery action is to attempt a rollback drop.

3. An attempt to commit a transient enrollment set will return a warning and the drop will proceed.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	ssEnrollDrop completed successfully
ss_enr_rc_error	*ss_enr_re_bad_drop_type*	Unrecognized drop type
ss_enr_rc_error	*ss_enr_re_db_not_found*	Named enrollment set not found

Return Code	Reason Code	Meaning
ss_enr_rc_warning	*ss_enr_re_not_disk*	Named enrollment set is transient
ss_enr_rc_error	*ss_enr_re_close_fail*	Call to DMSCLOSE failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO
PL/X	SSPLXENR COPY

ssEnrollList — List Enrollment Sets

```
ssEnrollList

   retcode
   reascode
   C-block_pointer
```

Purpose

Produces a summary list of the loaded enrollment sets.

Operands

ssEnrollList
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from
ssEnrollList.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from
ssEnrollList.

C-block_pointer
(input,POINTER,4) is a signed four-byte binary input variable holding the address of the C-block
representing the client to whom the summary list should be sent.

Usage Notes

1. The reusable server kernel writes the summary list to the client represented by *C-block_pointer*, using
   routine ssClientDataPut.

2. If the programmer wishes to capture the output of ssEnrollList for his own purposes, he can
   allocate storage to represent a C-block, initialize the C-block using routine ssClientDataInit,
   and then call routine ssEnrollList. When ssEnrollList returns, the programmer can call
   ssClientDataGet to retrieve the response. After the response is decoded, he should deallocate
   the C-block. Note that the response is record-oriented.

3. The form of the output is:

```
Name     Pages  Entries    InUse D K
----     -----  -------    ----- - -
test     256    1          1     0 d
```

The columns are:

Name
The name of the enrollment set

Pages
The size of the data space, in pages

Entries
The number of records in the enrollment set

InUse
The number of pages of data space storage being used to hold records

D

"Dirty" bit - if 1, set needs to be committed

K

Kind of set

d

On-disk (permanent)

m

In-memory (transient)

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	ssEnrollList completed successfully

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO
PL/X	SSPLXENR COPY

ssEnrollLoad — Load Enrollment Set

```
ssEnrollLoad

    retcode
    reascode
    set_name
    set_kind
    dataspace_size
    file_name
    file_name_length
```

Purpose

Loads an enrollment set from the Shared File System, or initializes an empty transient enrollment set.

Operands

ssEnrollLoad
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from
ssEnrollLoad.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from
ssEnrollLoad.

set_name
(input,CHAR,8) is a character string holding the name of the enrollment set to be loaded.

set_kind
(input,INT,4) is a signed four-byte binary input variable holding a value that indicates whether the
enrollment set is permanent or transient, as follows:

ss_enr_kind_memory
transient set

ss_enr_kind_disk
permanent set

dataspace_size
(input,INT,4) is a signed four-byte binary input variable holding the size of the dataspace.

file_name
(input,CHAR,*file_name_length*) is a character string holding the name of the SFS file containing the
enrollment set.

file_name_length
(input,INT,4) is a signed four-byte binary input variable holding the length of *file_name*.

Usage Notes

1. The name supplied in parameter *set_name* is used unchanged as a subpool name in a call to
   ssMemoryCreateDS. The server author must ensure that this name does not collide with any subpool
   names he might be using for other purposes.

2. The caller can use parameter *dataspace_size* to influence the size of the created data space. Express the size in pages. The reusable server kernel rounds the suggested size up to the next 16-page boundary before using it further. To refrain from influencing the data space size, specify a size of zero.

3. When it creates the data space, the reusable server kernel uses the larger of the following two parameters as the size of the space:

   • The number of records in the SFS file multiplied by the LRECL of the SFS file, multiplied by 1.5

   • The size requested by the caller in the *dataspace_size* parameter

   If the larger of these two sizes is less than 1 MB, then the reusable server kernel uses 1 MB (256 pages) instead.

4. Parameter *file_name* accepts any syntax acceptable to CSL routine DMSOPEN. This includes NAMEDEFs.

5. The file nominated by *file_name* must reside in the Shared File System. If the file does not (or would not) reside in the Shared File System, an error is returned and the enrollment set is not loaded.

6. The virtual machine in which the server program is running must have write authority to the file nominated by *file_name*.

7. If the file nominated by *file_name* does not exist, it is created and a warning is returned.

8. The file nominated by *file_name* is opened on its own work unit.

9. If a transient enrollment set is being loaded, no CMS file I/O takes place and no work unit is gotten. The data space is created, initialized as empty, and made ready to hold records.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	ssEnrollLoad completed successfully
ss_enr_rc_error	*ss_enr_re_bad_kind*	Parameter *set_kind* contains an unrecognized value
ss_enr_rc_error	*ss_enr_re_bad_length*	Parameter *file_name_length* contains an unrecognized value
ss_enr_rc_error	*ss_enr_re_no_storage*	Insufficient storage is available
ss_enr_rc_error	*ss_enr_re_db_exists*	Set *set_name* already exists
ss_enr_rc_error	*ss_enr_re_dscr_fail*	Attempt to create data space failed
ss_enr_rc_error	*ss_enr_re_gwu_fail*	Attempt to get work unit failed
ss_enr_rc_error	*ss_enr_re_open_fail*	Attempt to open file failed
ss_enr_rc_error	*ss_enr_re_not_sfs*	File is not SFS-resident
ss_enr_rc_error	*ss_enr_re_not_v*	File is not V-format
ss_enr_rc_error	*ss_enr_re_point_fail*	Attempt to move file pointers failed
ss_enr_rc_error	*ss_enr_re_read_fail*	Attempt to read SFS file failed
ss_enr_rc_warning	*ss_enr_re_new_file*	SFS file not found - new permanent enrollment set created

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO
PL/X	SSPLXENR COPY

ssEnrollRecordGet — Get Enrollment Record

```
ssEnrollRecordGet

    retcode
    reascode
    set_name
    key
    buffer
    buffer_size
    data_length
```

Purpose

Retrieves a record from an enrollment set.

Operands

ssEnrollRecordGet
 is the name of the function being invoked.

retcode
 (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssEnrollRecordGet`.

reascode
 (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssEnrollRecordGet`.

set_name
 (input,CHAR,8) is a character string holding the name of the enrollment set to be interrogated.

key
 (input,CHAR,64) is a character string holding the key of the record to be retrieved.

buffer
 (output,CHAR,*buffer_size*) is a character string buffer to hold the data of the retrieved record.

buffer_size
 (input,INT,4) is a signed four-byte binary input variable holding the size of *buffer*.

data_length
 (output,INT,4) is a signed four-byte binary output variable to hold the amount of data stored under key *key*.

Usage Notes

1. Every byte of the key is significant. If your application's keys are, say, text strings, be sure to pad your keys on the right to fill out the entire key field.

2. Case is significant in keys.

3. If the amount of data stored under key *key* will not fit in *buffer*, as much as will fit is returned, output *data_length* is set to the actual size of the data, and a warning is returned. This lets the caller retry the operation with a buffer large enough to hold all of the data.

4. If the record does not exist in set *set_name*, an error is returned.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	ssEnrollRecordGet completed successfully
ss_enr_rc_error	*ss_enr_re_db_not_found*	Set *set_name* does not exist
ss_enr_rc_error	*ss_enr_re_rec_not_found*	No record matches key *key*
ss_enr_rc_warning	*ss_enr_re_truncated*	Record was found but truncated because *buffer* could not contain all of it

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO
PL/X	SSPLXENR COPY

ssEnrollRecordInsert — Insert Enrollment Record

```
ssEnrollRecordInsert

    retcode
    reascode
    set_name
    key
    buffer
    data_length
    insert_type
```

Purpose

Inserts or replaces a record in an enrollment set.

Operands

ssEnrollRecordInsert
  is the name of the function being invoked.

retcode
  (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssEnrollRecordInsert`.

reascode
  (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssEnrollRecordInsert`.

set_name
  (input,CHAR,8) is a character string holding the name of the enrollment set to be modified.

key
  (input,CHAR,64) is a character string holding the key of the record to be inserted or replaced.

buffer
  (output,CHAR,*data_length*) is a character string buffer holding the data to be associated with *key*.

buffer_size
  (input,INT,4) is a signed four-byte binary input variable holding the size of *buffer*.

data_length
  (output,INT,4) is a signed four-byte binary output variable to hold the amount of data stored under key *key*.

insert_type
  (input,INT,4) is a signed four-byte binary input variable to hold the kind of insertion being done:

  ss_enr_insert_new
    New record

  ss_enr_insert_replace
    Replacement record

Usage Notes

1. Every byte of the key is significant. If your application's keys are, say, text strings, be sure to pad your keys on the right to fill out the entire key field.

2. Case is significant in keys.

3. The differences between *ss_enr_insert_new* and *ss_enr_insert_replace* are:

   - For *_new*, the reusable server kernel will fail the API call if the enrollment set already holds a record bearing key *key*. Thus the programmer can use *_new* to guard against inadvertent replacements.
   - For *_replace*, if the record bearing key *key* already exists, it is replaced and a warning is returned.

4. The change is not permanent until it is committed.

5. For permanent enrollment sets, the data cannot be more than 65,500 bytes long.

6. For transient enrollment sets, the data cannot be more than 16 MB long.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	ssEnrollRecordInsert completed successfully
ss_enr_rc_error	*ss_enr_re_bad_method*	Parameter *insert_type* contains an unrecognized value
ss_enr_rc_error	*ss_enr_re_bad_length*	Parameter *data_length* contains an invalid value
ss_enr_rc_error	*ss_enr_re_db_not_found*	Set *set_name* does not exist
ss_enr_rc_error	*ss_enr_re_no_storage*	Insufficient storage to satisfy request
ss_enr_rc_error	*ss_enr_re_write_storage*	Write to SFS file failed
ss_enr_rc_warning	*ss_enr_re_rec_exists*	Record exists and was replaced
ss_enr_rc_error	*ss_enr_re_rec_exists*	Record exists and was not replaced

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO
PL/X	SSPLXENR COPY

ssEnrollRecordList — List Records In Enrollment Set

```
ssEnrollRecordList

    retcode
    reascode
    set_name
    C-block_pointer
```

Purpose

Produces a summary list of the records in an enrollment set.

Operands

ssEnrollRecordList
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    `ssEnrollRecordList`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    `ssEnrollRecordList`.

set_name
    (input,CHAR,8) is a character string holding the name of the enrollment set.

C-block_pointer
    (input,POINTER,4) is a signed four-byte binary input variable holding the address of the C-block
    representing the client to whom the summary list should be sent.

Usage Notes

1. The reusable server kernel writes the summary list to the client represented by *C-block_pointer*, using
   routine `ssClientDataPut`.

2. If the programmer wishes to capture the output of `ssEnrollRecordList` for his own purposes, he
   can allocate storage to represent a C-block, initialize the C-block using routine `ssClientDataInit`,
   and then call routine `ssEnrollRecordList`. When `ssEnrollRecordList` returns, the programmer
   can call `ssClientDataGet` to retrieve the response. After the response is decoded, he should
   deallocate the C-block. Note that the response is record-oriented.

3. The output of `ssEnrollRecordList` is simply one enrollment record per output record. Each output
   record contains only the key of the corresponding enrollment record.

4. To retrieve the data associated with a given key, use `ssEnrollRecordGet`.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	`ssEnrollRecordList` completed successfully
ss_enr_rc_error	*ss_enr_re_db_not_found*	Set *set_name* is not loaded

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO
PL/X	SSPLXENR COPY

ssEnrollRecordRemove — Remove Enrollment Record

```
ssEnrollRecordRemove

    retcode
    reascode
    set_name
    key
```

Purpose

Removes a record from an enrollment set.

Operands

ssEnrollRecordRemove
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssEnrollRecordRemove`.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssEnrollRecordRemove`.

set_name
(input,CHAR,8) is a character string holding the name of the enrollment set to be modified.

key
(input,CHAR,64) is a character string holding the key of the record to be removed.

Usage Notes

1. Every byte of the key is significant. If your application's keys are, say, text strings, be sure to pad your keys on the right to fill out the entire key field.
2. Case is significant in keys.
3. If the record bearing key *key* is not found, an error is returned.
4. The change is not permanent until it is committed.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_enr_rc_success	*ss_enr_re_success*	ssEnrollRecordRemove completed successfully
ss_enr_rc_error	*ss_enr_re_db_not_found*	Set *set_name* does not exist
ss_enr_rc_error	*ss_enr_re_rec_not_found*	Record bearing key *key* does not exist
ss_enr_rc_error	*ss_enr_re_write_storage*	Write to SFS file failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMENR MACRO
PL/X	SSPLXENR COPY

ssMemoryAllocate — Allocate Memory

```
ssMemoryAllocate

    return_code
    reason_code
    min_bytes_needed
    max_bytes_needed
    subpool_name
    align_type
    memory_pointer
    bytes_obtained
```

Purpose

Allocates a block of primary storage (memory).

Operands

ssMemoryAllocate
is the name of the function being invoked.

return_code
(output,INT,4) is a signed four-byte binary output variable to hold the return code from ssMemoryAllocate.

reason_code
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssMemoryAllocate.

min_bytes_needed
(input,INT,4) is a signed four-byte binary input variable holding the minimum number of bytes needed.

max_bytes_needed
(input,INT,4) is a signed four-byte binary input variable holding the maximum number of bytes needed.

subpool_name
(input,CHAR,8) is a character string holding the name of the subpool from which the storage should be allocated.

align_type
(input,INT,4) is a signed four-byte binary input variable holding the type of alignment the new buffer will require.

memory_pointer
(output,INT,4) is a signed four-byte binary output variable to hold the returned memory address.

bytes_obtained
(output,INT,4) is a signed four-byte binary output variable to hold the returned number of bytes actually allocated.

Usage Notes

1. To issue a request for a block of storage of variable size, set *min_bytes_needed* equal to the minimum amount of storage needed and set *max_bytes_needed* equal to the maximum amount of storage desired.

2. To issue a request for a block of storage of fixed size, set *min_bytes_needed=max_bytes_needed*.

3. Parameter *subpool_name* is used unchanged in calls to CMSSTOR and therefore must adhere to CMSSTOR's rules for subpool names.

4. Parameter *align_type* must have one of these values:

   ss_mem_align_norm
   Align allocated storage on doubleword boundary

   ss_mem_align_page
   Align allocated storage on page boundary

5. The reusable server kernel allocates and releases memory in multiples of doublewords. The amount of storage requested by the caller will be rounded up to the next doubleword boundary before the allocation request is processed.

6. If the requested storage could not be obtained, *memory_pointer* and *bytes_obtained* are set to zero and appropriate return and reason codes are returned.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_mem_rc_success	*ss_mem_re_success*	ssMemoryAllocate completed successfully
ss_mem_rc_error	*ss_mem_re_bad_align*	*align_type* is not recognized
ss_mem_rc_error	*ss_mem_re_bad_amount*	Error in amount specification
ss_mem_rc_error	*ss_mem_re_out_of_storage*	Storage could not be obtained
ss_mem_rc_error	*ss_mem_re_subpool_deleted*	Subpool deleted while call was in progress

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMMEM MACRO
PL/X	SSPLXMEM COPY

ssMemoryCreateDS — Create Data Space

```
ssMemoryCreateDS

    return_code
    reason_code
    subpool_name
    number_of_pages
    storage_key
    option_count
    option_array
    asit
    alet
```

Purpose

Creates a data space and prepares the reusable server kernel to manage the storage therein.

Operands

ssMemoryCreateDS
is the name of the function being invoked.

return_code
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssMemoryCreateDS`.

reason_code
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssMemoryCreateDS`.

subpool_name
(input,CHAR,8) is a character string holding the subpool name to be assigned to the new data space.

number_of_pages
(input,INT,4) is a signed four-byte binary input variable specifying the size to be passed to DMSSPCC.

storage_key
(input,INT,4) is a signed four-byte binary input variable specifying the storage key to be passed to DMSSPCC.

option_count
(input,INT,4) is a signed four-byte binary input variable specifying the option count to be passed to DMSSPCC.

option_array
(input,INT,4*option_count) is an array of signed four-byte binary input variables specifying the option array to be passed to DMSSPCC.

asit
(output,CHAR,8) is an output character buffer to hold the returned ASIT.

alet
(output,INT,4) is an signed four-byte binary output variable to hold the returned ALET.

Usage Notes

1. Review the usage notes for CSL routines DMSSPCC and DMSSPLA before using `ssMemoryCreateDS`. For more information, see *z/VM: CMS Callable Services Reference*.

2. The value of *subpool_name* is used in constructing the name of the data space and therefore must adhere to the character set composition rules for data space names. For more information, see the description of CSL routine DMSSPCC in the book *z/VM: CMS Callable Services Reference*.

3. The reusable server kernel uses storage in the primary address space to keep track of free and used pieces of storage in the data space. The primary address space storage used for this purpose is taken from CMS through CMSSTOR OBTAIN under subpool name *subpool_name*.

4. Parameters *number_of_pages* and *storage_key* are passed directly to DMSSPCC.

5. If *option_count* is zero, ssMemoryCreateDS uses DMSSPCC's defaults, except that it asks for the data space to be created SHARE. The virtual machine's XCONFIG ADDRSPACE directory entry must be set up accordingly.

6. ssMemoryCreateDS asks DMSSPLA to create the ALET using the WRITE and SYNCH options. The reusable server kernel does not keep track of the generated ALET; the application is free to use DMSSPLR and DMSSPLA to manipulate ALETs.

7. After calling ssMemoryCreateDS successfully, allocate and release storage in the data space using routines ssMemoryAllocate and ssMemoryRelease.

8. To delete the data space, use ssMemoryDelete.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_mem_rc_success	*ss_mem_re_success*	ssMemoryCreateDS completed successfully
ss_mem_rc_error	*ss_mem_re_bad_amount*	*number_of_pages* is invalid
ss_mem_rc_error	*ss_mem_re_bad_key*	*storage_key* is invalid
ss_mem_rc_error	*ss_mem_re_spcc_fail*	DMSSPCC failed
ss_mem_rc_error	*ss_mem_re_spla_fail*	DMSSPLA failed
ss_mem_rc_error	*ss_mem_re_out_of_storage*	Storage could not be obtained
ss_mem_rc_error	*ss_mem_re_subpool_exists*	Subpool already exists

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMMEM MACRO
PL/X	SSPLXMEM COPY

ssMemoryDelete — Delete Subpool

```
ssMemoryDelete

    return_code
    reason_code
    subpool_name
```

Purpose

Deletes a memory subpool, and the corresponding data space if there is one.

Operands

ssMemoryDelete
    is the name of the function being invoked.

return_code
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssMemoryDelete`.

reason_code
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssMemoryDelete`.

subpool_name
    (input,CHAR,8) is a character string holding the name of the subpool to be deleted.

Usage Notes

1. The reusable server kernel deletes its record of the subpool and issues a corresponding SUBPOOL DELETE call to CMS.
2. If the subpool is a data space, the corresponding data space is also deleted.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_mem_rc_success	*ss_mem_re_success*	`ssMemoryDelete` completed successfully
ss_mem_rc_error	*ss_mem_re_no_subpool*	Unrecognized subpool name
ss_mem_rc_error	*ss_mem_re_spd_fail*	SUBPOOL DELETE call failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMMEM MACRO
PL/X	SSPLXMEM COPY

ssMemoryRelease — Release Memory

```
ssMemoryRelease

    return_code
    reason_code
    bytes_released
    subpool_name
    memory_pointer
```

Purpose

Releases a block of primary storage (memory).

Operands

ssMemoryRelease
is the name of the function being invoked.

return_code
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssMemoryRelease`.

reason_code
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssMemoryRelease`.

bytes_released
(input,INT,4) is a signed four-byte binary input variable holding the number of bytes being released.

subpool_name
(input,CHAR,8) is a character string holding the name of the subpool from which the storage was allocated.

memory_pointer
(input,INT,4) is a signed four-byte binary input variable holding the address of the storage being released.

Usage Notes

1. The buffer being released must reside on a doubleword boundary.

2. If it does not represent an integral number of doublewords, parameter *bytes_released* is rounded up to the next doubleword boundary before being used.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_mem_rc_success	*ss_mem_re_success*	ssMemoryRelease completed successfully
ss_mem_rc_error	*ss_mem_re_bad_align*	Buffer is not aligned on doubleword boundary
ss_mem_rc_error	*ss_mem_re_bad_amount*	Error in amount specification
ss_mem_rc_error	*ss_mem_re_no_subpool*	Unrecognized subpool name

ssMemoryRelease

Return Code	Reason Code	Meaning
ss_mem_rc_error	*ss_mem_re_not_alloc*	Some or all of buffer is already free
ss_mem_rc_error	*ss_mem_re_subpool_deleted*	Subpool deleted while call in progress
ss_mem_rc_error	*ss_mem_re_out_of_storage*	Not enough storage available

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMMEM MACRO
PL/X	SSPLXMEM COPY

ssServerRun — Run the Server

```
ssServerRun

    retcode
    reascode
```

Purpose

Runs the server program.

Operands

ssServerRun
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from
ssServerRun.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from
ssServerRun.

Usage Notes

Call this routine only from RSKMAIN and only after you have called ssServiceBind sufficiently to set up
your server.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_srv_rc_success	*ss_srv_re_success*	ssServerRun completed successfully
ss_srv_rc_error	anything else	Nonzero return code from PROFILE RSK.

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSRV MACRO
PL/X	SSPLXSRV COPY

ssServerStop — Stop the Server

```
ssServerStop

    retcode
    reascode
```

Purpose

Stops the server program.

Operands

ssServerStop
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from
> `ssServerStop`.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
> `ssServerStop`.

Usage Notes

Calling this function will cause the `WAITSERV` command in `PROFILE  RSK` to complete.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_srv_rc_success	*ss_srv_re_success*	`ssServerStop` completed successfully

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSRV MACRO
PL/X	SSPLXSRV COPY

ssServiceBind — Bind A Service

```
ssServiceBind

    retcode
    reascode
    service_name
    service_name_length
    init_addr
    service_addr
    term_addr
```

Purpose

Informs the reusable server kernel of the existence of a new service.

Operands

ssServiceBind
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    `ssServiceBind`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    `ssServiceBind`.

service_name
    (input,CHAR,*service_name_length*) is the name of the new service.

service_name_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of the service name.

init_addr
    (input,INT,4) is a signed four-byte binary input variable holding the address of the service's
    initialization entry point.

service_addr
    (input,INT,4) is a signed four-byte binary input variable holding the address of the service's service
    entry point.

term_addr
    (input,INT,4) is a signed four-byte binary input variable holding the address of the service's
    termination entry point.

service_type
    (input,INT,4) is a signed four-byte binary input variable holding the kind of service being bound.

Usage Notes

1. Case is not significant in service names.

2. The parameter list array passed to the initialization entry point (pointed to by R1) is organized as
   shown in Table 3 on page 6.

3. To signal successful initialization, the initialization entry point should return with the return and reason
   code words set to zero. A nonzero return code will cause the start of the service to fail.

4. The parameter list array passed to the service entry point (pointed to by R1) is organized as shown in Table 4 on page 7.
5. The parameter list array passed to the termination entry point (pointed to by R1) is organized as shown in Table 5 on page 7.
6. The values that can be supplied for *service_type* are:

   ss_srv_srvtype_normal
   > Plain old service.

   ss_srv_srvtype_ld
   > Plain old line driver.

   ss_srv_srvtype_ldss
   > Self-sourced line driver.

7. To activate the service, use one of the line drivers' START commands.
8. ssServiceBind will produce correct results only when it is called by RSKMAIN prior to ssServerRun. ssServiceBind should never be called under any other conditions. Unpredicable results could occur.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_srv_rc_success	*ss_srv_re_success*	ssServiceBind completed successfully
ss_srv_rc_error	*ss_srv_re_out_of_range*	*service_name_length*<0 or >8
ss_srv_rc_error	*ss_srv_re_bad_type*	*service_type* contains unrecognized value.
ss_srv_rc_error	*ss_srv_re_exists*	Service already exists
ss_srv_rc_error	*ss_srv_re_out_of_storage*	Out of storage

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSRV MACRO
PL/X	SSPLXSRV COPY

ssServiceFind — Find A Service

---

ssServiceFind

> *retcode*
> *reascode*
> *service_name*
> *service_name_length*
> *S-block_address*

Purpose

Obtains descriptive information about a service.

Operands

ssServiceFind
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssServiceFind`.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssServiceFind`.

service_name
> (input,CHAR,*service_name_length*) is the name of the new service.

service_name_length
> (input,INT,4) is a signed four-byte binary input variable holding the length of the service name.

S-block_address
> (output,POINTER,4) is a signed four-byte binary output variable to hold the address of the found service's S-block.

Usage Notes

1. Case is not significant in service names.

2. The returned S-block is organized according to Table 2 on page 6.

3. If the service could not be found, a return and reason code are generated and *sblock_address* is returned as 0.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_srv_rc_success	*ss_srv_re_success*	`ssServiceFind` completed successfully
ss_srv_rc_error	*ss_srv_re_out_of_range*	*service_name_length*<0 or >8
ss_srv_rc_error	*ss_srv_re_not_found*	The named service could not be found.

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSRV MACRO
PL/X	SSPLXSRV COPY

ssSgpCreate — Create a Storage Group

```
ssSgpCreate

    retcode
    reascode
    storage_group_number
    minidisk_count
    minidisk_array
    attribute_count
    attribute_array
```

Purpose

Identifies a set of minidisks to be managed as a storage group.

Operands

ssSgpCreate
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from ssSgpCreate.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssSgpCreate.

storage_group_number
(input,INT,4) is a signed four-byte binary input variable holding the number of the new storage group.

minidisk_count
(input,INT,4) is a signed four-byte binary input variable holding the number of minidisks in the new storage group.

minidisk_array
(input,INT,4*minidisk_count) is an array of signed four-byte binary input variables holding the device addresses of the minidisks to be included in the new storage group.

attribute_count
(input,INT,4) is a signed four-byte binary input variable holding the number of attributes in the *attribute_array* array.

attribute_array
(input,INT,4*attribute_count) is an array of signed four-byte binary input variables holding the attributes to be associated with the new storage group.

Usage Notes

1. Parameter *storage_group_number* must be in the range 0 to 1023, inclusive.

2. Each minidisk to be included in the storage group must have already been formatted at 4 KB by the FORMAT command and reserved by the RESERVE command. The reusable server kernel requires that its minidisks exhibit this format.

3. There is a limit of 13,000 minidisks per storage group, and the sum of the sizes of the data areas on the minidisks must not exceed X'FFFFFFFF' 4 KB blocks.

4. The storage group's existence is recorded in the storage group definition file and persists across instances of the server program. For more information on the description of the storage group definition file, see Chapter 12, "Initialization and Profiles," on page 63.

5. No attributes are currently recognized in the *attribute_array* (in other words, if *attribute_count* is nonzero, *ss_sgp_re_bad_attrib* is returned).

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	`ssSgpCreate` completed successfully
ss_sgp_rc_error	*ss_sgp_re_out_of_range*	*storage_group_number*, *minidisk_count* or *attribute_count* is out of range
ss_sgp_rc_error	*ss_sgp_re_bad_attrib*	*attribute_array* contains an unrecognized attribute
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex creation or acquisition failed
ss_sgp_rc_error	*ss_sgp_re_exists*	Storage group already exists
ss_sgp_rc_error	*ss_sgp_re_out_of_storage*	Out of storage
ss_sgp_rc_error	*ss_sgp_re_cv_fail*	Condition variable creation failed
ss_sgp_rc_warning	*ss_sgp_re_rewrite_fail*	Rewrite of storage group definitions failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpDelete — Delete a Storage Group

```
ssSgpDelete

    retcode
    reascode
    storage_group_number
```

Purpose

Removes a set of minidisks from the control of the reusable server kernel.

Operands

ssSgpDelete
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssSgpDelete`.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssSgpDelete`.

storage_group_number
(input,INT,4) is a signed four-byte binary input variable holding the number of the storage group to be deleted.

Usage Notes

1. To be deleted, the storage group must not be started.

2. The storage group definition file is updated to reflect the fact that the storage group no longer exists.

3. No I/O is done to the storage group as part of deletion; the minidisks remain as they were. To recreate the storage group, just issue an appropriate call to `ssSgpCreate`.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	`ssSgpDelete` completed successfully
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex acquisition failed
ss_sgp_rc_error	*ss_sgp_re_not_found*	Storage group not found
ss_sgp_rc_error	*ss_sgp_re_online*	Storage group is online
ss_sgp_rc_error	*ss_sgp_re_maint*	Maintenance in progress
ss_sgp_rc_warning	*ss_sgp_re_rewrite_fail*	Rewrite of storage group definitions failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpFind — Find a Storage Group

```
ssSgpFind

    retcode
    reascode
    storage_group_name
    storage_group_number
    io_mode
    total_blocks
```

Purpose

Returns information about the storage group whose name is supplied.

Operands

ssSgpFind
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssSgpFind`.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssSgpFind`.

storage_group_name
(input,CHAR,8) is an input character string holding the name of the storage group to find.

storage_group_number
(output,INT,4) is a signed four-byte binary output variable to hold the number of the found storage group.

io_mode
(output,INT,4) is a signed four-byte binary output variable to hold the I/O mode of the found storage group.

total_blocks
(output,INT,4) is a signed four-byte binary output variable to hold the number of blocks in the storage group.

Usage Notes

1. Because the lookup is by name, only started storage groups can be found.
2. Right-pad the name with spaces.
3. The value returned in *io_mode* is one of:

   ss_sgp_attrib_block_rw
   Started read-write

   ss_sgp_attrib_block_ro
   Started read-only

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	`ssSgpFind` completed successfully

ssSgpFind

Return Code	Reason Code	Meaning
ss_sgp_rc_error	*ss_sgp_re_not_found*	Storage group is not found

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpList — List Storage Groups

```
ssSgpList

    retcode
    reascode
    number_expected
    number_returned
    storage_group_list
```

Purpose

Returns a list of the known storage groups.

Operands

ssSgpList
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from ssSgpList.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssSgpList.

number_expected
(input,INT,4) is a signed four-byte binary input variable holding the number of storage groups whose identifiers can fit into the *storage_group_list* array.

number_returned
(output,INT,4) is a signed four-byte binary output variable to hold the number of storage group identifiers placed into the *storage_group_list* array.

storage_group_list
(output,INT,4*number_expected) is an array of signed four-byte binary output variables to hold the identifiers of the existing storage groups.

Usage Notes

1. If the actual number of existing storage groups is greater than *number_expected*, then the actual number of storage groups is filled into *number_returned*, the identifiers of the first *number_expected* storage groups are returned in *storage_group_list*, and a warning is given.

2. To determine information about a particular storage group, use ssSgpQuery.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	ssSgpList completed successfully
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex acquisition failed
ss_sgp_rc_warning	*ss_sgp_re_too_many*	More storage groups than *number_expected*

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpQuery — Query a Storage Group

```
ssSgpQuery

    retcode
    reascode
    storage_group_number
    io_mode
    total_blocks
    status_word
    attributes_expected
    attributes_returned
    attribute_array
    minidisks_expected
    minidisks_returned
    minidisk_address_array
    minidisk_blocks_array
```

Purpose

Returns information about a specific storage group.

Operands

ssSgpQuery
  is the name of the function being invoked.

retcode
  (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssSgpQuery`.

reascode
  (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssSgpQuery`.

storage_group_number
  (input,INT,4) is a signed four-byte binary input variable holding the number of the storage group about which information is desired.

io_mode
  (output,INT,4) is a signed four-byte binary output variable to hold the storage group I/O mode.

total_blocks
  (output,INT,4) is a signed four-byte binary output variable to hold the total number of 4 KB blocks in the storage group.

status_word
  (output,INT,4) is a signed four-byte binary output variable to hold the storage group status word.

attributes_expected
  (input,INT,4) is a signed four-byte binary input variable holding the number of attribute identifiers that will fit in the *attribute_array* array.

attributes_returned
  (input,INT,4) is a signed four-byte binary output variable to hold the number of entries filled into the *attribute_array* array.

attribute_array
  (output,INT,4*attribute_count*) is an array of signed four-byte binary output variables to hold the returned storage group attribute indicators.

minidisks_expected
(input,INT,4) is a signed four-byte binary input variable holding the number of minidisks for which descriptive information will fit in the *minidisk_address_array*, *minidisk_total_array*, and *minidisk_free_array* arrays.

minidisks_returned
(output,INT,4) is a signed four-byte binary output variable to hold the number of minidisks for which descriptive information was deposited in the *minidisk_address_array*, *minidisk_total_array*, and *minidisk_free_array* arrays.

minidisk_address_array
(output,INT,4*minidisks_expected*) is an array of signed four-byte binary output variables to hold the returned minidisk addresses.

minidisk_total_array
(output,INT,4*minidisks_expected*) is an array of signed four-byte binary output variables to hold the returned sizes of each of the minidisks in the storage group.

minidisk_free_array
(output,INT,4*minidisks_expected*) is an array of signed four-byte binary output variables to hold the returned free block counts for each of the minidisks in the storage group.

Usage Notes

1. The possible values returned for *io_mode* are:

    ss_sgp_attrib_offline
    Not started (not online)

    ss_sgp_attrib_block_ro
    Started for read-only block I/O

    ss_sgp_attrib_block_rw
    Started for read-write block I/O

2. The size information (total blocks, blocks per minidisk) and status word returned by this function are meaningful only if the storage group is started.

3. The integer returned in *status_word* is to be interpreted bit-by-bit according to the following key. In this key, the bits are numbered from 0 to 31, most significant to least significant. If the named bit is set, the condition is true. The bits that are not mentioned are meaningless.

Bit	Description
0	Stop in progress
1	VM Data Spaces in use
2	DIAG X'250' in use

4. No attributes are currently returned in *attribute_array*.

5. If the actual number of minidisks is greater than *minidisks_expected,* then the actual number of minidisks is returned in parameter *minidisks_returned*, the descriptive information for the first *minidisks_expected* minidisks is filled into the arrays, and a warning is given.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	`ssSgpQuery` completed successfully
ss_sgp_rc_error	*ss_sgp_re_out_of_range*	Bad value for *attributes_expected* or *minidisks_expected*
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex acquisition failed
ss_sgp_rc_error	*ss_sgp_re_not_found*	Storage group not found

Return Code	Reason Code	Meaning
ss_sgp_rc_warning	*ss_sgp_re_too_many*	More attributes than *attributes_expected* or more minidisks than *minidisks_expected*

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpRead — Read a Storage Group

---

ssSgpRead

> *retcode*
> *reascode*
> *storage_group_number*
> *starting_block*
> *block_count*
> *buffer_alet*
> *buffer*

Purpose

Reads one or more blocks from a storage group.

Operands

ssSgpRead
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from ssSgpRead.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from ssSgpRead.

storage_group_number
> (input,INT,4) is a signed four-byte binary input variable holding the number of the storage group from which blocks should be read.

starting_block
> (input,INT,4) is a signed four-byte binary input variable holding the starting block number of the block extent to be read.

block_count
> (input,INT,4) is a signed four-byte binary input variable holding the number of blocks to be read.

buffer_alet
> (input,INT,4) is a signed four-byte binary input variable holding the ALET to be used when referring to *buffer*.

buffer
> (output,CHAR,4096*block_count*) is a character string to hold the data read from the storage group.

Usage Notes

1. The first block of the storage group is block 0.

2. This entry point can be used only if the storage group is online.

3. This entry point does not serialize access to storage groups. If your application performs storage group I/O on multiple threads, it is possible that the I/O might happen in parallel, *especially in MP situations*. It is the application developer's responsibility to implement any serialization paradigms required.

4. When VM Data Spaces are used, the transfer from the storage group's data space to the target space is done with PSW key 0.

5. When a CP DIAGNOSE is used, CP is instructed to use key 0 in the channel programs it builds.

6. If DIAG X'A4' is being used for storage group I/O, *buffer_alet* must be 0.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	ssSgpRead completed successfully
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex acquisition failed
ss_sgp_rc_error	*ss_sgp_re_not_found*	Storage group not found
ss_sgp_rc_error	*ss_sgp_re_out_of_range*	Extent is not within storage group size
ss_sgp_rc_error	*ss_sgp_re_io_fail*	Requested read failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpStart — Start a Storage Group

```
ssSgpStart

    retcode
    reascode
    storage_group_number
    storage_group_name
    attribute_count
    attribute_array
```

Purpose

Makes a storage group ready for use.

Operands

ssSgpStart
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssSgpStart`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssSgpStart`.

storage_group_number
    (input,INT,4) is a signed four-byte binary input variable holding the number of the storage group to be brought online.

storage_group_name
    (input,CHAR,8) is a character string holding the name to be assigned to the storage group while it is online.

attribute_count
    (input,INT,4) is a signed four-byte binary input variable holding the number of attributes present in the *attribute_array* array.

attribute_array
    (input,INT,4*attribute_count) is an array of signed four-byte binary input variables holding the attributes to be used in bringing the storage group online.

Usage Notes

1. Each minidisk to be included in the storage group must have already been formatted at 4 KB by the FORMAT command and reserved by the RESERVE command. The reusable server kernel requires that its minidisks exhibit this format.

2. There is a limit of 13,000 minidisks per storage group, and the sum of the sizes of the data areas on the minidisks must not exceed 16 TB (X'FFFFFFFF' 4 KB blocks).

3. To be eligible for starting, the storage group must be completely stopped.

4. These attributes are recognized in the *attribute_array* (defaults are labeled as such):

   ss_sgp_attrib_ds
       Use VM Data Spaces MAPMDISK facility (default)

   ss_sgp_attrib_no_ds
       Do not use VM Data Spaces MAPMDISK facility

> **ss_sgp_attrib_block_rw**
> Online read-write for block I/O (default)

> **ss_sgp_attrib_block_ro**
> Online read-only for block I/O

5. To use *ss_sgp_attrib_ds* successfully, the real hardware and the server virtual machine's CP directory entry must be set up appropriately. This includes:

   - The z/VM system must be running on an ESA/390(™) processor.
   - In the CP directory, MACHINE XC must be specified.
   - In the CP directory, XCONFIG ADDRSPACE must allow enough data spaces to span the storage groups. Each 2 GB or fraction thereof in a storage group requires one data space.
   - In the CP directory, XCONFIG ADDRSPACE must allow an aggregate data space size at least as large as the sum of the sizes of the storage groups to be brought online with this attribute.

6. If *ss_sgp_attrib_ds* is specified and the reusable server kernel could not activate VM Data Spaces support for it, then the reusable server kernel:

   a. Sets a warning return code indicating why VM Data Spaces failed, and
   b. Attempts to bring the storage group online as if *ss_sgp_attrib_no_ds* had been specified.

7. If *ss_sgp_attrib_no_ds* is specified, then the reusable server kernel makes use of DIAGNOSE X'250' or DIAGNOSE X'A4' for I/O to the storage group, as follows:

   a. The reusable server kernel attempts to initialize the DIAGNOSE X'250' environment for each minidisk in the storage group, using the diagnose in asynchronous mode and with minidisk caching (MDC) enabled.
   b. If DIAGNOSE X'250' initialization is successful for all minidisks in the storage group, then DIAGNOSE X'250' is used for I/O to the storage group.
   c. If DIAGNOSE X'250' initialization fails for at least one minidisk in the storage group, then DIAGNOSE X'A4' is used for I/O to the storage group and a warning return code and reason code are returned.

8. Reason codes related to VM Data Spaces are produced with a warning return code. These reason codes indicate that the use of VM Data Spaces failed and that DIAGNOSE X'250' is being used instead.

9. Reason codes related to DIAGNOSE X'250' are produced with a warning return code. These reason codes indicate that the use of DIAGNOSE X'250' failed and that DIAGNOSE X'A4' is being used instead.

10. If reason code *ss_sgp_re_read_only* is produced and it really is desired to bring the storage group online read-write, follow these steps:

Step	Task
1	Determine which minidisk(s) are linked read-only.
2	Detach the read-only minidisks and link them read-write.
3	Try again to start the storage group.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	`ssSgpStart` completed successfully
ss_sgp_rc_error	*ss_sgp_re_bad_attrib*	Unrecognized item in attribute array
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex acquisition failed
ss_sgp_rc_error	*ss_sgp_re_not_found*	Storage group not found

ssSgpStart

Return Code	Reason Code	Meaning
ss_sgp_rc_error	*ss_sgp_re_name_in_use*	Storage group name already in use
ss_sgp_rc_warning	*ss_sgp_re_online*	Storage group is already online
ss_sgp_rc_error	*ss_sgp_re_vdq_fail*	Minidisk format incorrect or query of format failed
ss_sgp_rc_error	*ss_sgp_re_read_only*	At least one minidisk is linked read-only
ss_sgp_rc_warning	*ss_sgp_re_ds_fail*	Data space creation failed
ss_sgp_rc_warning	*ss_sgp_re_pool_fail*	MAPMDISK minidisk pool definition failed
ss_sgp_rc_warning	*ss_sgp_re_map_fail*	MAPMDISK minidisk pool mapping failed
ss_sgp_rc_warning	*ss_sgp_re_diag_250_fail*	Use of DIAGNOSE X'250' failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpStop — Stop a Storage Group

```
ssSgpStop

    retcode
    reascode
    storage_group_number
    attribute_count
    attribute_array
```

Purpose

Makes a storage group unready.

Operands

ssSgpStop
> is the name of the function being invoked.

retcode
> (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssSgpStop`.

reascode
> (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssSgpStop`.

storage_group_number
> (input,INT,4) is a signed four-byte binary input variable holding the number of the storage group to be taken offline.

attribute_count
> (input,INT,4) is a signed four-byte binary input variable holding the number of attributes present in the *attribute_array* array.

attribute_array
> (input,INT,4*attribute_count) is an array of signed four-byte binary input variables holding the attributes to be used in taking the storage group offline.

Usage Notes

1. To stop all defined storage groups, set *storage_group_number* to -1.
2. Once the stop of the storage group begins, no more block I/O may be started, and the stop completes only after all block I/O to the storage group is completed.
3. No elements are currently recognized in *attribute_array*.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	ssSgpStop completed successfully
ss_sgp_rc_error	*ss_sgp_re_out_of_range*	Bad value for *attribute_count*
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex acquisition failed
ss_sgp_rc_error	*ss_sgp_re_not_found*	Storage group not found
ss_sgp_rc_warning	*ss_sgp_re_offline*	Already stopped or stop in progress

ssSgpStop

Return Code	Reason Code	Meaning
ss_sgp_rc_error	*ss_sgp_re_cv_fail*	Condition variable wait failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssSgpWrite — Write a Storage Group

```
ssSgpWrite

    retcode
    reascode
    storage_group_number
    starting_block
    block_count
    buffer_alet
    buffer
```

Purpose

Writes one or more blocks to a storage group.

Operands

ssSgpWrite
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssSgpWrite`.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssSgpWrite`.

storage_group_number
(input,INT,4) is a signed four-byte binary input variable holding the number of the storage group to which blocks should be written.

starting_block
(input,INT,4) is a signed four-byte binary input variable holding the starting block number of the block extent to be written.

block_count
(input,INT,4) is a signed four-byte binary input variable holding the number of blocks to be written.

buffer_alet
(input,INT,4) is a signed four-byte binary input variable holding the ALET to be used when referring to *buffer*.

buffer
(input,CHAR,4096*block_count) is a character string holding the data to be written to the storage group.

Usage Notes

1. The first block of the storage group is block 0.

2. This entry point can be used only if the storage group is online with attribute *ss_sgp_attrib_block_rw*.

3. This entry point does not serialize access to storage groups. If your application performs storage group I/O on multiple threads, it is possible that the I/O might happen in parallel, *especially in MP situations*. It is the application developer's responsibility to implement any serialization paradigms required.

4. When VM Data Spaces are used, the transfer from the source space to the storage group's data space is done with PSW key 0.

5. When a CP DIAGNOSE is used, CP is instructed to use key 0 in the channel programs it builds.

6. If DIAG X'A4' is being used for storage group I/O, *buffer_alet* must be 0.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_sgp_rc_success	*ss_sgp_re_success*	`ssSgpWrite` completed successfully
ss_sgp_rc_error	*ss_sgp_re_mx_fail*	Mutex acquisition failed
ss_sgp_rc_error	*ss_sgp_re_not_found*	Storage group not found
ss_sgp_rc_error	*ss_sgp_re_out_of_range*	Extent is not within storage group size
ss_sgp_rc_error	*ss_sgp_re_wrong_mode*	Storage group is not started for read-write block I/O
ss_sgp_rc_error	*ss_sgp_re_io_fail*	Requested write failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMSGP MACRO
PL/X	SSPLXSGP COPY

ssTrieCreate — Create a Trie

```
ssTrieCreate

    retcode
    reascode
    triename
    triesize
    trieasit
    triealet
```

Purpose

Creates a trie.

Operands

ssTrieCreate
:   is the name of the function being invoked.

retcode
:   (output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssTrieCreate`.

reascode
:   (output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssTrieCreate`.

triename
:   (input,CHAR,8) is a character string holding the name of the new trie.

triesize
:   (input,INT,4) is a signed four-byte binary input variable holding the size of the new trie's data space, in pages.

trieasit
:   (output,CHAR,8) is a character string to hold the ASIT of the data space for the new trie.

triealet
:   (output,INT,4) is a signed four-byte binary output variable to hold the ALET associated with the new trie's data space.

Usage Notes

1. The name supplied in parameter *triename* is used unchanged as a subpool name in a call to `ssMemoryCreateDS`. The server author must ensure that this name does not collide with any subpool names he might be using for other purposes.

2. The caller should specify parameter *triesize* in pages. The reusable server kernel passes *triesize* directly to `ssMemoryCreateDS`.

3. The reusable server kernel creates the new trie in a data space and returns the data space's ASIT and ALET to the caller.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_tri_rc_success	*ss_tri_re_success*	`ssTrieCreate` completed successfully

ssTrieCreate

Return Code	Reason Code	Meaning
ss_tri_rc_error	*ss_tri_re_bad_size*	*triesize* <0 or >524288
ss_tri_rc_error	*ss_tri_re_trie_exists*	Trie *triename* already exists
ss_tri_rc_error	*ss_tri_re_out_of_storage*	Out of storage
ss_tri_rc_error	*ss_tri_re_dscr_fail*	Call to `ssMemoryCreateDS` failed

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMTRI MACRO
PL/X	SSPLXTRI COPY

ssTrieDelete — Delete a Trie

```
ssTrieDelete

    retcode
    reascode
    triename
```

Purpose

Deletes a trie.

Operands

ssTrieDelete
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssTrieDelete`.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssTrieDelete`.

triename
(input,CHAR,8) is a character string holding the name of the trie to be deleted.

Usage Notes

1. This call results in the data space being deleted via call to `ssMemoryDelete`.
2. If your application has shared the trie's ASIT with other virtual machines, your application is responsible for telling those other virtual machines about the upcoming deletion prior to calling `ssTrieDelete`.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_tri_rc_success	*ss_tri_re_success*	`ssTrieDelete` completed successfully
ss_tri_rc_error	*ss_tri_re_trie_not_found*	Trie *triename* was not found
ss_tri_rc_error	*ss_tri_re_trie_busy*	Unable to acquire lock necessary to delete trie

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMTRI MACRO
PL/X	SSPLXTRI COPY

ssTrieRecordInsert — Insert Record Into Trie

---

ssTrieRecordInsert

    retcode
    reascode
    triename
    triealet
    recnum
    index_buffer
    index_length

Purpose

Inserts the record number into the trie, using the specified key.

Operands

ssTrieRecordInsert
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    `ssTrieRecordInsert`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    `ssTrieRecordInsert`.

triename
    (input,CHAR,8) is a character string holding the name of the trie into which the record is to be
    inserted.

triealet
    (input,INT,4) is a signed four-byte binary input variable holding the ALET of the data space in which
    the trie resides.

recnum
    (input,INT,4) is a signed four-byte binary input variable holding the record number to be inserted into
    the trie.

index_buffer
    (input,CHAR,*index_length*) is a character string holding the index of the record being inserted.

index_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of *index_buffer*.

Usage Notes

1. If your virtual machine created the trie, you may use either the trie name or the trie ALET value to
   identify the trie. If *triealet* is nonzero the reusable server kernel will use your ALET directly. To refer to
   your trie by name, set *triealet* to zero and use input *triename* to specify the name of your trie.

2. If your virtual machine did not create the trie (that is, if the creator passed you the trie ASIT and you
   generated the ALET yourself), you must use parameter *triealet* to pass the reusable server kernel the
   ALET you generated for the trie. In this case, what you pass via *triename* is irrelevant.

3. The index string must not be longer than 256 bytes.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_tri_rc_success	*ss_tri_re_success*	ssTrieRecordInsert completed successfully
ss_tri_rc_error	*ss_tri_re_bad_index_len*	Index string has improper length
ss_tri_rc_error	*ss_tri_re_trie_not_found*	Trie *triename* was not found
ss_tri_rc_error	*ss_tri_re_trie_busy*	Unable to acquire lock necessary to update trie
ss_tri_rc_error	*ss_tri_re_out_of_ds_storage*	The data space is full

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMTRI MACRO
PL/X	SSPLXTRI COPY

ssTrieRecordList — List Matching Records

```
ssTrieRecordList

    retcode
    reascode
    triename
    triealet
    index_buffer
    index_length
    recnum_array
    recnum_array_capacity
    records_found
```

Purpose

Generates a list of all the record numbers whose keys match the specified prefix.

Operands

ssTrieRecordList
is the name of the function being invoked.

retcode
(output,INT,4) is a signed four-byte binary output variable to hold the return code from `ssTrieRecordList`.

reascode
(output,INT,4) is a signed four-byte binary output variable to hold the reason code from `ssTrieRecordList`.

triename
(input,CHAR,8) is a character string holding the name of the trie to be interrogated.

triealet
(input,INT,4) is a signed four-byte binary input variable holding the ALET of the data space in which the trie resides.

index_buffer
(input,CHAR,*index_length*) is a character string holding the key prefix to be used in the lookup.

index_length
(input,INT,4) is a signed four-byte binary input variable holding the length of *index_buffer*.

recnum_array
(output,INT,4**recnum_array_capacity*) is an array of signed four-byte binary output variables to hold the record numbers whose keys match the supplied prefix.

recnum_array_capacity
(input,INT,4) is a signed four-byte binary input variable holding the size of *recnum_array*.

records_found
(output,INT,4) is a signed four-byte binary output variable to hold the number of record numbers found.

Usage Notes

1. If your virtual machine created the trie, you may use either the trie name or the trie ALET value to identify the trie. If *triealet* is nonzero the reusable server kernel will use your ALET directly. To refer to your trie by name, set *triealet* to zero and use input *triename* to specify the name of your trie.

2. If your virtual machine did not create the trie (that is, if the creator passed you the trie ASIT and you generated the ALET yourself), you must use parameter *triealet* to pass the reusable server kernel the ALET you generated for the trie. In this case, what you pass via *triename* is irrelevant.

3. The index string must not be longer than 256 bytes.

4. The reusable server kernel examines the trie and determines the set of record numbers whose keys' prefixes match the prefix you specified in *index_buffer*. It then writes the record numbers to the *recnum_array* array.

5. If there are more matching records than *recnum_array* can hold, the reusable server kernel fills *recnum_array* to capacity, writes the actual number of matching records to *records_found*, and returns success. You must always examine *records_found* to determine whether your array was large enough.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_tri_rc_success	*ss_tri_re_success*	`ssTrieRecordList` completed successfully
ss_tri_rc_error	*ss_tri_re_bad_index_len*	Index string has improper length
ss_tri_rc_error	*ss_tri_re_bad_capacity_len*	*recnum_array_capacity* must be ≥ 0
ss_tri_rc_error	*ss_tri_re_trie_not_found*	Trie *triename* was not found
ss_tri_rc_error	*ss_tri_re_trie_busy*	Unable to acquire lock necessary to update trie

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMTRI MACRO
PL/X	SSPLXTRI COPY

ssUseridMap — Produce Mapped User ID

```
ssUseridMap

    retcode
    reascode
    linedriver
    linedriver_length
    input_node
    input_node_length
    input_userid
    input_userid_length
    output_userid
    output_userid_length
```

Purpose

Maps line-driver-specific information through the user ID mapping file.

Operands

ssUseridMap
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    ssUseridMap.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    ssUseridMap.

linedriver
    (input,CHAR,*linedriver_length*) is a character string holding the name of the line driver.

linedriver_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of *linedriver*.

input_node
    (input,CHAR,*input_node_length*) is a character string holding the input node for the mapping function.

input_node_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of *input_node*.

input_userid
    (input,CHAR,*input_userid_length*) is a character string holding the input user ID for the mapping
    function.

input_userid_length
    (input,INT,4) is a signed four-byte binary input variable holding the length of *input_userid*.

output_userid
    (output,CHAR,64) is a character string to hold the output of the mapping function.

output_userid_length
    (output,INT,4) is a signed four-byte binary output variable to hold the length of the retrieved user ID.

Usage Notes

1. The reusable server kernel maps the triplet (*linedriver,input_node,input_userid*) through the user ID mapping file and returns the resultant user identifier.

2. For more information about the organization and use of the user ID mapping file, see "User ID Mapping Facility" on page 69.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_uid_rc_success	*ss_uid_re_success*	`ssUseridMap` completed successfully
ss_uid_rc_error	*ss_uid_re_not_found*	No matching entry in user ID mapping file

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMUID MACRO
PL/X	SSPLXUID COPY

ssWorkerAllocate — Allocate Connection to Worker Machine

---

ssWorkerAllocate

>   *retcode*
>   *reascode*
>   *instance_C-block*
>   *class_name*
>   *option_count*
>   *option_names*
>   *option_values*
>   *worker_C-block*
>   *connection_ID*

Purpose

Allocates a connection to a worker machine, autologging a worker if necessary.

Operands

ssWorkerAllocate
    is the name of the function being invoked.

retcode
    (output,INT,4) is a signed four-byte binary output variable to hold the return code from
    `ssWorkerAllocate`.

reascode
    (output,INT,4) is a signed four-byte binary output variable to hold the reason code from
    `ssWorkerAllocate`.

instance_C-block
    (input,POINTER,4) is a pointer holding the address of the C-block previously created for the calling
    instance by its own line driver.

class_name
    (input,CHAR,8) is a character string holding the name of the class from which a worker machine
    should be selected.

option_count
    (input,INT,4) is a signed four-byte binary input variable holding the number of elements in the
    option_names and *option_values* arrays.

option_names
    (input,INT,4*option_count*) is an array of signed four-byte binary input variables holding option names.

option_values
    (input,INT,4*option_count*) is an array of signed four-byte binary input variables holding option values.

worker_C-block
    (output,POINTER,4) is a pointer to hold the address of the returned worker C-block, constructed by
    the server kernel to represent the connection between the instance and the selected worker.

connection_ID
    (output,INT,4) is a signed four-byte binary output variable to hold the returned connection ID.

Usage Notes

1. Input *instance_C-block* is the address of the C-block assigned to the instance by its line driver. This value was passed to the instance in its own parameter list when the instance was started.

2. If the service instance prefers not to receive worker API notifications on its line driver's queue, the service instance can set parameter *instance_C-block* to 0 (zero) instead. Supplying a value of 0 will cause the server kernel to build an instance C-block in which only the following fields are valid:

   vc_qh
   > line driver queue handle

   vc_ikey
   > instance key

   The service instance can use these values as inputs in subsequent calls to QueueReceiveImmed or QueueReceiveBlock, to receive messages indicating worker activity. The C-block is not useful for any other purpose. The server kernel returns the address of the built C-block in parameter *instance_C-block*.

3. The worker class *class_name* should correspond to a class defined through the WORKER ADD command. If the class has not yet been created through WORKER ADD, an error is returned.

4. Case is significant in class names.

5. The *option_names* array can contain any of these values:

   ss_wrk_ofn_prefer_empty
   > The corresponding entry in the *option_values* array controls how the server kernel will search for an available worker, as follows:
   >
   > **ss_wrk_ofv_yes**
   >> The server kernel will search for empty or not-yet-logged-on worker machines first and direct the connection to one of those. If no such worker is found the server kernel will determine the least burdened worker and direct the connection to it.
   >
   > **ss_wrk_ofv_no**
   >> The server kernel will search the already-logged-on workers, determine the least burdened one, and direct the connection to it. If no workers are logged on yet, or if all logged-on workers are full, the server kernel will autolog another worker and direct the connection to it.

   ss_wrk_ofn_retry_count
   > The corresponding value in the *option_values* array is the number of worker machines the server kernel should try before it gives up. Specifying a count of zero means that the server kernel should try until it runs out of worker machine candidates.

   ss_wrk_ofn_alt_userid
   > The corresponding value in the *option_values* array is a pointer to an 8-byte character string which is the alternate user ID to use.

   ss_wrk_ofn_alt_seclabel
   > The corresponding value in the *option_values* array is a pointer to an 8-byte character string which is the alternate seclabel to use.

6. The server kernel maintains status information about the workers in each class and uses that status information when considering whether to try to connect to a worker. The status information, an integer, indicates either that the worker machine appears healthy or tells the reason why the last attempt to connect to the worker machine failed. For more information, see "WORKER MACHINES" on page 208.

7. To be able to set a worker's alternate user ID and seclabel, the controlling virtual machine must have permission to issue Diagnose X'D4'. See *z/VM: CP Programming Services* for more information. If you attempt to use the reusable server kernel's alternate user ID machinery and your virtual machine does not have the privilege necessary to issue Diagnose X'D4', your virtual machine will take a program check. It is your responsibility to recover from this.

8. If you specify an alternate seclabel, you must also specify an alternate user ID. The reusable server kernel does not check this.

9. Output *worker_C-block* will contain the address of the C-block that describes the connection from the instance to the worker. The instance should consult this C-block for:

- The queue handle it should use when sending IPC messages to the server kernel about this worker connection

- The line driver key it should use when sending IPC messages to the server kernel about this worker connection

10. The returned connection ID will appear in IPC messages arriving on the instance's line driver queue. These messages, keyed with the instance's key, are indicative of activity on the worker connection.

Messages and Return Codes

Return Code	Reason Code	Meaning
ss_wrk_rc_success	*ss_wrk_re_success*	`ssWorkerAllocate` completed successfully
ss_wrk_rc_error	*ss_wrk_re_out_of_storage*	Insufficient storage to connect to worker
ss_wrk_rc_error	*ss_wrk_re_bad_count*	Input *option_count* contains a negative value
ss_wrk_rc_error	*ss_wrk_re_bad_flag_name*	Input *option_names* contains an unrecognized name
ss_wrk_rc_error	*ss_wrk_re_bad_flag_value*	Input *option_values* contains an unrecognized value
ss_wrk_rc_error	*ss_wrk_re_no_class*	The class you requested does not exist
ss_wrk_rc_error	*ss_wrk_re_no_subordinates*	No worker machine could be found
ss_wrk_rc_error	*ss_wrk_re_algtries_exceeded*	The last worker machine tried was autologged several times but the IUCV connection never succeeded
ss_wrk_rc_error	*ss_wrk_re_autolog_fail*	The server kernel was unable to autolog the last virtual machine it tried
ss_wrk_rc_error	*ss_wrk_re_timer_fail*	The server kernel tried to use the CMS Timer API to set a timer but the Timer API failed
ss_wrk_rc_error	*ss_wrk_re_iucvcon_fail*	The server kernel encountered an unrecoverable IUCV CONNECT error on the last worker virtual machine it tried
ss_wrk_rc_error	*ss_wrk_re_force_fail*	The server kernel tried to CP FORCE a worker (to reset it) but was unable to issue the FORCE command
ss_wrk_rc_error	*ss_wrk_re_force_timeout*	The server kernel FORCEd a worker (to reset it) but did not see the worker become logged off - possible hung user
ss_wrk_rc_error	*ss_wrk_re_oper_delete*	While the server kernel was trying to bring up the worker connection, the operator issued `WORKER DELETE` or `WORKER DELCLASS`, thus nullifying the connection attempt

Programming Language Bindings

Language	Language Binding File
Assembler	SSASMWRK MACRO
PL/X	SSPLXWRK COPY

Chapter 16. RSK Sockets

The RSK socket library is a PL/X application programming interface for socket programming. The library is a *very thin layer* over the IUCV socket interface and can be used only within an RSK program. [25] While the RSK socket library does not provide a correspondent for every IUCV socket function, it provides many of the basic operations necessary to communicate with other socket programs. The RSK socket library also provides some RSK-specific functions.

The RSK socket library is aware of multitasking CMS and integrates well with it. For example, when a socket operation blocks, only the calling thread blocks. Further, the library offers extensions to traditional socket semantics, making available asynchronous versions of often-used socket calls (such as *write()*). When the caller performs an asynchronous socket operation, the completion notice arrives as a message on a CMS queue.

Prerequisite Knowledge

This chapter assumes you have a working knowledge of the Reusable Server Kernel. You will also need to be experienced in socket programming, such as from having used IUCV sockets, C sockets, or Rexx/Sockets. To use the asynchronous features of the RSK socket library, you will need to understand CMS interprocess communication (IPC) as implemented by multitasking CMS's "queue" functions (e.g., `QueueReceiveBlock`). Finally, you will need to know how to program in PL/X.

To use the RSK socket documentation effectively, you will need a copy of the "IUCV Sockets" section of *z/VM: TCP/IP Programmer's Reference*. That material gives complete usage information for the IUCV socket API. The best way to use this RSK socket library documentation is to refer to the RSK socket documentation and the IUCV socket documentation side-by-side.

Available Functions

The following IUCV socket functions have correspondents in the RSK socket interface:

Table 47. Socket Functions Available in RSK Library

IUCV socket function name	RSK entry point name
accept()	PS_accept()
bind()	PS_bind()
close()	PS_close()
connect()	PS_connect()
gethostid()	PS_gethostid()
getpeername()	PS_getpeername()
getsockname()	PS_getsockname()
getsockopt()	PS_getsockopt()

---

[25] That is, the callers of the RSK socket library entry points must adhere to the RSK linkage and automatic storage conventions. See Chapter 11, "Run-Time Environment," on page 59 for more information.

Table 47. Socket Functions Available in RSK Library (continued)

IUCV socket function name	RSK entry point name
ioctl()	PS_ioctl()
listen()	PS_listen()
read()	PS_read()
recvfrom()	PS_recvfrom()
select()	PS_select()
sendto()	PS_sendto()
setsockopt()	PS_setsockopt()
shutdown()	PS_shutdown()
socket()	PS_socket()
write()	PS_write()

The following additional functions are specific to the RSK socket library:

Table 48. Additional RSK-Specific Functions in Library

Function	RSK entry point name
Library initialization	PS_libinit()
Library termination	PS_libterm()
Application initialization	PS_applinit()
Application termination	PS_applterm()
Asynchronous *read()*	PS_async_read()
Asynchronous *recvfrom()*	PS_async_recv()
Asynchronous *sendto()*	PS_async_sendto()
Asynchronous *write()*	PS_async_write()
Cancel asynchronous operation	PS_cancel()

Programming with RSK Sockets

Programming with the RSK socket library involves the following steps:

1. In each of your PL/X compilation units that will use the RSK socket library, you must *include the RSK socket library language binding macro*. To do so, put the following statement into each compilation unit:

```
%include syslib(plxsock);
```

   PLXSOCK COPY is in DMSRP MACLIB, which is part of the z/VM PL/X Restricted Source Feature, which you can order as a feature of z/VM.

2. At run-time, your first step must be to initialize the RSK socket library. This prepares the library to receive socket calls. To initialize the library, you must either call *PS_libinit()* yourself or arrange for the RSK to call it. See "PS_libinit" on page 349 for more information.

3. To perform socket operations, you must create a *socket set*. [26] We call each RSK socket set an *application* and hence the entry point you use for this is *PS_applinit()*.

   You supply *PS_applinit()* with the name (VM user ID) of the TCP/IP stack machine, a unique name for your new set of sockets, and the number of sockets you want in the set.

   PS_applinit() establishes the IUCV connection to the TCP/IP stack machine and prepares the socket set for your use.

4. You perform operations on the sockets in your set. You use the RSK socket library entry points to do so. For example, to allocate a new socket, you call *PS_socket()*, or to write data to a socket, you call *PS_write()*.

5. When you are done with your set of sockets, you dispose of it by calling *PS_applterm()*, identifying the socket set by the unique name you chose for it at its creation.

6. Prior to your server ending, either you should call *PS_libterm()* or you should arrange for the RSK to call it. See "PS_libterm" on page 350 for more information.

Restrictions and Limitations

Be aware of the following restrictions and limitations when you use the RSK socket library:

- The RSK socket library uses storage subpool name DMSSBPS0. You should refrain from using this subpool name.
- The RSK socket library creates an HNDIUCV exit named DMSPLXSK. You should refrain from using this HNDIUCV exit name.
- The RSK socket library creates CMS semaphores whose names are of the form DMSPLXSKxxxx, where xxxx is a hexadecimal number. You should refrain from using semaphore names of these forms.
- Each socket set may contain 50 to 2000 sockets, inclusive.
- The RSK itself uses socket set names of the form Uxxxxxxx and Txxxxxxx, where xxxxxxx is a hexadecimal number. You should refrain from using socket set names of these forms.
- You may create more than one named socket set concurrently. The absolute limit on the number of socket sets the library can manage is set by call to *PS_libinit()*. [27] This limit counts both socket sets you create yourself and RSK UDP or TCP subtasks you have running in your server. Each such subtask uses one socket set.
- You may overlap operations on a socket set, but you should not overlap operations on a single socket. For example, if you use *PS_async_write()* to write data to a socket, you should not start another write to that socket until the current write to that socket finishes.
- When you call a synchronous socket operation (such as *PS_write()*), the calling thread blocks until the operation completes. Other CMS threads might run while the calling thread waits for the operation to complete. While the synchronous operation is in progress, other threads are permitted to perform operations on other sockets in that socket set and on other socket sets.

---

[26] In IUCV sockets, this step corresponds to establishing a connection to the TCP/IP stack machine and sending the initial message. In Rexx/Sockets, this step corresponds to invoking `Socket('Initialize')`.

[27] When the RSK calls *PS_libinit()*, it sets the limit to 100.

Data Structures

Certain data structures are important in socket programming. For example, the 16-byte structure containing the address of a new client (known to C programmers as *sockaddr_in*) is used throughout the API. Here are some hypothetical PL/X representations of those data structures. These representations are referred to in the routines' descriptions below, *but they are not provided in PLXSOCK COPY* and are here just for illustrative purposes.

Address Structure

```
/* sockaddr_in */
declare
 1  sockaddr_in        based boundary(word),
    5  si_family       fixed(15),           /* address family */
    5  si_port         fixed(16),           /* port number    */
    5  si_address      fixed(32),           /* IP address     */
    5  si_zero         char(8);             /* must be zero    */
```

Timeout Structure

```
/* timeout structure for select() */
declare
 1  timeval            based boundary(word),
    5  tv_sec          fixed(31),           /* seconds        */
    5  tv_usec         fixed(31);           /* microseconds   */
```

Notes on PLXSOCK COPY

The language binding file PLXSOCK COPY contains constant definitions, structure definitions, and function prototypes. Some notes on each:

Constants

Certain (but certainly not all) constants relevant to socket programming appear in PLXSOCK COPY. When the library requires you to supply a constant (such as AF_INET), check the binding to see if a symbolic name is available. If there is no symbolic name, you will have to make up your own.

Structures

PLXSOCK COPY contains definitions for certain structures commonly used in socket programming. Feel free to use these structures if you find them helpful.

Function Prototypes

PLXSOCK COPY contains function prototypes for each RSK socket library entry point.

Return Codes and ERRNO Values

By and large, the return code values and *errno* values returned by the RSK socket library correspond exactly to the values returned by the IUCV socket API. The following exceptions apply:

- Some entry points unique to the RSK socket library (such as *PS_applinit()*) supply a return and reason code. The descriptions below list the return and reason codes that might be produced.

- The RSK socket library defines additional *errno* values not found in the IUCV socket API. These *errno* values come from the additional complexity in the RSK socket library. Their symbolic names and meanings are:

Name
   Meaning

EIBMIUCVERR
   Some kind of IUCV error occurred

EIBMLIBERR
   The RSK socket library is not initialized

EIBMNOAPPL
   The socket set you named does not exist

EIBMNOSOCKAVAIL
   No sockets available in socket set

EIBMBADKEYLEN
   Notify key length is invalid

EIBMNOSTORAGE
   No storage available

EIBMBADBUFLEN
   A supplied buffer length is invalid

EIBMBADPARM
   Timeout buffer length is invalid

EIBMSHUTDOWN
   The TCP/IP stack is shutting down

Any of the RSK socket library routines having *errno* as an output might produce some of these *errno* values.

RSK Socket Calls

This section provides the PL/X language syntax, parameters, and other appropriate information for each socket call the RSK supports.

The parameter lists and syntax for each routine are illustrated with PL/X snippets. *These snippets are not verbatim examples you can compile and run.* They just show the data type of each parameter list entry, whether the item is input (**I**) or output (**O**), and how to code the CALL statement to invoke the function.

Usage notes here are confined to explaining particulars of the RSK socket API. As a result, the information here is intentionally terse. Again, refer to "IUCV Sockets" in *z/VM: TCP/IP Programmer's Reference*.

PS_accept

Purpose

Performs socket *accept()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname      char(8),
 lsocket       fixed(31),
 addrbufptr    pointer(31),
 addrbufsize   fixed(31),
 addrlen       fixed(31),
 socket        fixed(31),
 errno         fixed(31);

/* how to call */
call PS_accept
(
```

```
applname,              /* I:  application name      */
lsocket,               /* I:  listen socket         */
addrbufptr,            /* I:  address buffer pointer */
addrbufsize,           /* I:  address buffer size   */
addrlen,               /* O:  address length        */
socket,                /* O:  new socket number     */
errno                  /* O:  ERRNO                 */
);
```

Parameters

Parameter
   Definition

applname
   Name of socket set

lsocket
   Socket you listened on

addrbufptr
   Pointer to buffer into which API should place a completed sockaddr_in structure

addrbufsize
   Size of said buffer

addrlen
   Returned length of sockaddr_in structure

socket
   Socket number for new connection

errno
   Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_applinit

Purpose

Creates a socket set.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 rc          fixed(31),
 re          fixed(31),
 tcpname     char(8),
 applname    char(8),
 numwanted   fixed(31),
 numgotten   fixed(31);

/* how to call */
call PS_applinit
(
 rc,                    /* O:  return code           */
 re,                    /* O:  reason code           */
 tcpname,               /* I:  name of TCP/IP stack  */
```

```
  applname,              /* I:  appl name to use     */
  numwanted,             /* I:  num of sockets wanted */
  numgotten              /* O:  num of sockets gotten */
);
```

Parameters

Parameter
   Definition

rc
   Return code

re
   Reason code

tcpname
   User ID of TCP/IP stack machine

applname
   Name for new socket set

numwanted
   Number of sockets wanted (50 to 2000)

numgotten
   Number of sockets gotten

Reason Codes

Reason Code
   Meaning

sok_re_bad_ns
   numwanted is out of range

sok_re_dup_appl
   applname already in use

sok_re_ic_fail
   IUCV CONNECT to stack failed

sok_re_bad_inttype
   Stack responded improperly to CONNECT

sok_re_is_fail
   IUCV SEND to stack failed

sok_re_diff_ns
   numgotten ¬= numwanted

sok_re_no_library
   Socket library not initialized

sok_re_no_apps
   Library unable to handle additional socket sets

Usage Notes

1. If you get a warning return code and you get reason code sok_re_diff_ns, you may proceed to use the socket set, recognizing you did not get as many sockets as you requested.

2. If you get an error return code and you get reason code sok_re_diff_ns, the socket set was not created because the TCP/IP stack tried to give you more sockets than you requested.

PS_applterm

Purpose

Terminates a socket set.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 rc           fixed(31),
 re           fixed(31),
 applname     char(8);

/* how to call */
call PS_applterm
(
 rc,                    /* O:  return code           */
 re,                    /* O:  reason code           */
 applname               /* I:  set to terminate      */
);
```

Parameters

Parameter
> **Definition**

rc
> Return code

re
> Reason code

applname
> Name of socket set to terminate

Reason Codes

Reason Code
> **Meaning**

sok_re_no_appl
> Application not found

sok_re_no_library
> Socket library not initialized

Usage Notes

None.

PS_async_read

Purpose

Starts a read of a socket. The library sends an IPC message when the read completes.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname     char(8),
 socket       fixed(31),
```

```
 bufpointer    pointer(31),
 bufsize       fixed(31),
 nqhandle      fixed(31),
 nkpointer     pointer(31),
 nklength      fixed(31),
 xid           fixed(31),
 rc            fixed(31),
 errno         fixed(31);

/* how to call */
call PS_async_read
(
 applname,                /* I:  application name     */
 socket,                  /* I:  socket to read       */
 bufpointer,              /* I:  pointer to read buffer */
 bufsize,                 /* I:  size of read buffer   */
 nqhandle,                /* I:  handle of notify queue */
 nkpointer,               /* I:  pointer to notify key  */
 nklength,                /* I:  length of notify key   */
 xid,                     /* O:  transaction ID        */
 rc,                      /* O:  return code           */
 errno                    /* O:  ERRNO                 */
);
```

Parameters

Parameter
   Definition

applname
   Name of socket set

socket
   Socket to read

bufpointer
   Pointer to buffer to be filled

bufsize
   Amount of data wanted

nqhandle
   Handle of notify queue

nkpointer
   Pointer to key for notify message

nklength
   Length of notify message

xid
   Transaction ID

rc
   Return code

errno
   Returned ERRNO

Reason Codes

Not applicable.

Usage Notes

1. The handle for the notify queue must be a service ID. In other words, the queue in which the notification is to be placed must be a service queue. You must have already arranged for this by calling QueueIdentifyService.

2. The notification message you see in the service queue will be the concatenation of your notify key and the following extra data:

Offset.Length
   Usage

0.4
   Return code

4.4
   Errno

8.16
   Unused

3. The message will be sent with your notify key as its key.

4. If you need to cancel the operation before it completes, use the returned transaction ID in a call to *PS_cancel()*.

PS_async_recv

Purpose

Starts a receive of a datagram. The library sends an IPC message when the receive completes.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname     char(8),
 socket       fixed(31),
 bufpointer   pointer(31),
 bufsize      fixed(31),
 flagword     fixed(31),
 nqhandle     fixed(31),
 nkpointer    pointer(31),
 nklength     fixed(31),
 xid          fixed(31),
 rc           fixed(31),
 errno        fixed(31);

/* how to call */
call PS_async_recv
(
 applname,                /* I:  application name       */
 socket,                  /* I:  socket to receive on   */
 bufpointer,              /* I:  pointer to recv buffer */
 bufsize,                 /* I:  size of recv buffer    */
 flagword,                /* I:  flag word              */
 nqhandle,                /* I:  handle of notify queue */
 nkpointer,               /* I:  pointer to notify key  */
 nklength,                /* I:  length of notify key   */
 xid,                     /* O:  transaction ID         */
 rc,                      /* O:  return code            */
 errno                    /* O:  ERRNO                  */
);
```

Parameters

Parameter
   Definition

applname
   Name of socket set

socket
   Socket to receive on

bufpointer
   Pointer to buffer to be filled

bufsize
>   Amount of data wanted

flagword
>   Flag word

nqhandle
>   Handle of notify queue

nkpointer
>   Pointer to key for notify message

nklength
>   Length of notify message

xid
>   Transaction ID

rc
>   Return code

errno
>   Returned ERRNO

Reason Codes

Not applicable.

Usage Notes

1. See the IUCV socket library documentation for definition of the flag word.

2. The handle for the notify queue must be a service ID. In other words, the queue in which the notification is to be placed must be a service queue. You must have already arranged for this by calling QueueIdentifyService.

3. The notification message you see in the service queue will be the concatenation of your notify key and the following extra data:

   Offset.Length
   >   Usage

   0.4
   >   Return code

   4.4
   >   Errno

   8.16
   >   sockaddr_in describing message source

4. The message will be sent with your notify key as its key.

5. If you need to cancel the operation before it completes, use the returned transaction ID in a call to *PS_cancel()*.

PS_async_sendto

Purpose

Starts a send of a datagram. The library sends an IPC message when the send completes.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
```

```
declare
 applname      char(8),
 socket        fixed(31),
 bufpointer    pointer(31),
 bufsize       fixed(31),
 flagword      fixed(31),
 addrbufptr    pointer(31),
 addrbufsize   fixed(31),
 nqhandle      fixed(31),
 nkpointer     pointer(31),
 nklength      fixed(31),
 xid           fixed(31),
 rc            fixed(31),
 errno         fixed(31);

/* how to call */
call PS_async_sendto
(
 applname,                /* I:  application name      */
 socket,                  /* I:  socket to send on     */
 bufpointer,              /* I:  pointer to data buffer */
 bufsize,                 /* I:  size of data buffer   */
 flagword,                /* I:  flag word             */
 addrbufptr,              /* I:  pointer to addr buffer */
 addrbufsize,             /* I:  size of addr buffer   */
 nqhandle,                /* I:  handle of notify queue */
 nkpointer,               /* I:  pointer to notify key  */
 nklength,                /* I:  length of notify key   */
 xid,                     /* O:  transaction ID        */
 rc,                      /* O:  return code           */
 errno                    /* O:  ERRNO                 */
);
```

Parameters

Parameter
  Definition

applname
  Name of socket set

socket
  Socket to send on

bufpointer
  Pointer to data buffer

bufsize
  Length of data buffer

flagword
  Flag word

addrbufptr
  Pointer to sockaddr_in structure

addrbufsize
  Length of sockaddr_in structure

nqhandle
  Handle of notify queue

nkpointer
  Pointer to key for notify message

nklength
  Length of notify message

xid
  Transaction ID

rc
  Return code

errno
    Returned ERRNO

Reason Codes

Not applicable.

Usage Notes

1. See the IUCV socket library documentation for definition of the flag word.
2. The handle for the notify queue must be a service ID. In other words, the queue in which the notification is to be placed must be a service queue. You must have already arranged for this by calling QueueIdentifyService.
3. The notification message you see in the service queue will be the concatenation of your notify key and the following extra data:

   Offset.Length
       Usage
   0.4
       Return code
   4.4
       Errno
   8.16
       Unused

4. The message will be sent with your notify key as its key.
5. If you need to cancel the operation before it completes, use the returned transaction ID in a call to *PS_cancel()*.

PS_async_write

Purpose

Starts a write to a socket. The library sends an IPC message when the write completes.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname     char(8),
 socket       fixed(31),
 bufpointer   pointer(31),
 bufsize      fixed(31),
 nqhandle     fixed(31),
 nkpointer    pointer(31),
 nklength     fixed(31),
 xid          fixed(31),
 rc           fixed(31),
 errno        fixed(31);

/* how to call */
call PS_async_write
(
 applname,                /* I:  application name       */
 socket,                  /* I:  socket to write to     */
 bufpointer,              /* I:  pointer to data buffer */
 bufsize,                 /* I:  size of data buffer    */
 nqhandle,                /* I:  handle of notify queue */
 nkpointer,               /* I:  pointer to notify key  */
 nklength,                /* I:  length of notify key   */
 xid,                     /* O:  transaction ID         */
 rc,                      /* O:  return code            */
```

```
  errno                    /* 0:  ERRNO                    */
);
```

Parameters

Parameter
   Definition

applname
   Name of socket set

socket
   Socket to write to

bufpointer
   Pointer to data buffer

bufsize
   Length of data buffer

nqhandle
   Handle of notify queue

nkpointer
   Pointer to key for notify message

nklength
   Length of notify message

xid
   Transaction ID

rc
   Return code

errno
   Returned ERRNO

Reason Codes

Not applicable.

Usage Notes

1. The handle for the notify queue must be a service ID. In other words, the queue in which the notification is to be placed must be a service queue. You must have already arranged for this by calling QueueIdentifyService.

2. The notification message you see in the service queue will be the concatenation of your notify key and the following extra data:

   Offset.Length
      Usage

   0.4
      Return code

   4.4
      Errno

   8.16
      Unused

3. The message will be sent with your notify key as its key.

4. If you need to cancel the operation before it completes, use the returned transaction ID in a call to *PS_cancel()*.

PS_bind

Purpose

Performs *bind()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname     char(8),
 socket       fixed(31),
 addrbufptr   pointer(31),
 addrbufsize  fixed(31),
 rc           fixed(31),
 errno        fixed(31);

/* how to call */
call PS_bind
(
 applname,              /* I:  application name       */
 socket,                /* I:  socket for bind        */
 addrbufptr,            /* I:  address buffer pointer  */
 addrbufsize,           /* I:  address buffer size     */
 rc,                    /* O:  return code            */
 errno                  /* O:  ERRNO                  */
);
```

Parameters

Parameter
  Definition

applname
   Name of socket set

socket
   Socket for bind

addrbufptr
   Pointer to your built sockaddr_in structure

addrbufsize
   Length of your sockaddr_in structure

rc
   Return code

errno
   Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_cancel

Purpose

Cancels an asynchronous RSK socket function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname    char(8),
 xid         fixed(31),
 rc          fixed(31),
 errno       fixed(31);

/* how to call */
call PS_cancel
(
 applname,              /* I:  application name      */
 xid,                   /* I:  transaction to cancel */
 rc,                    /* O:  return code           */
 errno                  /* O:  ERRNO                 */
);
```

Parameters

Parameter
 Definition

applname
   Name of socket set

xid
   Transaction to cancel

rc
   Return code

errno
   Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_close

Purpose

Performs *close()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname    char(8),
 socket      fixed(31),
 rc          fixed(31),
 errno       fixed(31);

/* how to call */
call PS_close
(
 applname,              /* I:  application name      */
 socket,                /* I:  socket to close       */
```

```
 rc,                     /* O:  return code          */
 errno                   /* O:  ERRNO                */
);
```

Parameters

Parameter
  Definition

applname
  Name of socket set

socket
  Socket to close

rc
  Return code

errno
  Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_connect

Purpose

Performs *connect()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname     char(8),
 socket       fixed(31),
 addrbufptr   pointer(31),
 addrbufsize  fixed(31),
 rc           fixed(31),
 errno        fixed(31);

/* how to call */
call PS_connect
(
 applname,              /* I:  application name       */
 socket,                /* I:  socket to use          */
 addrbufptr,            /* I:  pointer to sockaddr_in  */
 addrbufsize,           /* I:  length of sockaddr_in   */
 rc,                    /* O:  return code            */
 errno                  /* O:  ERRNO                  */
);
```

Parameters

Parameter
  Definition

applname
  Name of socket set

socket
Socket to close

addrbufptr
Pointer to sockaddr_in describing target

addrbufsize
Length of sockaddr_in

rc
Return code

errno
Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_gethostid

Purpose

Performs *gethostid()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname     char(8),
 hostid       fixed(31),
 errno        fixed(31);

/* how to call */
call PS_gethostid
(
 applname,              /* I:  application name       */
 hostid,                /* O:  host ID                */
 errno                  /* O:  ERRNO                  */
);
```

Parameters

Parameter
    Definition

applname
Name of socket set

hostid
Returned host ID

errno
Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_getpeername

Purpose

Performs *getpeername()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname     char(8),
 socket       fixed(31),
 addrbufptr   pointer(31),
 addrbufsize  fixed(31),
 rc           fixed(31),
 errno        fixed(31);

/* how to call */
call PS_getpeername
(
 applname,                /* I:  application name       */
 socket,                  /* I:  socket number          */
 addrbufptr,              /* I:  pointer to sockaddr_in  */
 addrbufsize,             /* I:  length of sockaddr_in   */
 rc,                      /* O:  return code            */
 errno                    /* O:  ERRNO                  */
);
```

Parameters

Parameter
Definition

applname
Name of socket set

socket
Socket number

addrbufptr
Pointer to buffer to contain sockaddr_in

addrbufsize
Length of sockaddr_in

rc
Return code

errno
Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_getsockname

Purpose

Performs *getsockname()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname    char(8),
 socket      fixed(31),
 addrbufptr  pointer(31),
 addrbufsize fixed(31),
 rc          fixed(31),
 errno       fixed(31);

/* how to call */
call PS_getsockname
(
 applname,              /* I:  application name       */
 socket,                /* I:  socket number          */
 addrbufptr,            /* I:  pointer to sockaddr_in  */
 addrbufsize,           /* I:  length of sockaddr_in   */
 rc,                    /* O:  return code            */
 errno                  /* O:  ERRNO                  */
);
```

Parameters

Parameter
  Definition

applname
  Name of socket set

socket
  Socket number

addrbufptr
  Pointer to buffer to contain sockaddr_in

addrbufsize
  Length of sockaddr_in

rc
  Return code

errno
  Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_getsockopt

Purpose

Performs *getsockopt()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname          char(8),
 socket            fixed(31),
 level             fixed(31),
 optname           fixed(31),
 optvalptr         pointer(31),
 optvalbufsize     fixed(31),
 rc                fixed(31),
 errno             fixed(31);

/* how to call */
call PS_getsockopt
(
 applname,              /* I:  application name       */
 socket,                /* I:  socket number          */
 level,                 /* I:  level setting          */
 optname,               /* I:  option name            */
 optvalptr,             /* I:  pointer to value buffer */
 optvalbufsize,         /* I:  length of value buffer  */
 rc,                    /* O:  return code            */
 errno                  /* O:  ERRNO                  */
);
```

Parameters

Parameter
   Definition

applname
   Name of socket set

socket
   Socket number

level
   Option level

optname
   Name of option being interrogated

optvalptr
   Pointer to buffer for option value

optvalbufsize
   Size of buffer for option value

rc
   Return code

errno
   Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_ioctl

Purpose

Performs *ioctl()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname           char(8),
 socket             fixed(31),
 command            fixed(31),
 argstrptr          pointer(31),
 argstrlen          fixed(31),
 respbufptr         pointer(31),
 respbufsize        fixed(31),
 rc                 fixed(31),
 errno              fixed(31);

/* how to call */
call PS_ioctl
(
 applname,                   /* I:  application name        */
 socket,                     /* I:  socket number           */
 command,                    /* I:  ioctl command           */
 argstrptr,                  /* I:  pointer to arg string   */
 argstrlen,                  /* I:  length of arg string    */
 respbufptr,                 /* I:  pointer to resp buffer  */
 respbufsize,                /* I:  size of resp buffer     */
 rc,                         /* O:  return code             */
 errno                       /* O:  ERRNO                   */
);
```

Parameters

Parameter
  Definition

applname
  Name of socket set

socket
  Socket number

command
  ioctl() command

argstrptr
  Pointer to argument string

argstrlen
  Length of argument string

respbufptr
  Pointer to response buffer

respbufsize
  Size of response buffer

rc
  Return code

errno
  Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_libinit

Purpose

Initializes the RSK socket library.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 rc               fixed(31),
 re               fixed(31),
 numofapps        fixed(31);

/* how to call */
call PS_libinit
(
 rc,                      /* O:  return code          */
 re,                      /* O:  reason code          */
 numofapps                /* I:  number of applications */
);
```

Parameters

Parameter
   Definition

rc
   Return code

re
   Reason code

numofapps
   Number of concurrent applications

Reason Codes

Reason Code
   Meaning

sok_re_success
   Function worked correctly

sok_re_already
   Socket library already initialized

sok_re_bad_appl_count
   numofapps is out of range

sok_re_out_of_storage
   Insufficient storage

sok_re_hs_fail
   HNDIUCV SET failed

sok_re_sc_fail
   SemCreate failed

Usage Notes

1. You need to coordinate your use of *PS_libinit()* with the RSK's TCP and UDP line drivers.

    The objective in such coordination is to make sure that if the RSK decides to call *PS_libinit()*, its call will work. (Most service levels of the RSK cannot tolerate failure of a call to *PS_libinit()*.)

    If you plan **never ever** to use any of the IP functions in the RSK, you will definitely need to call *PS_libinit()* exactly once to initialize the RSK socket library, so you should go ahead and issue the call before you issue any other RSK socket calls.

    However, if your server starts the TCP or UDP line drivers (for example, SUBCOM START UDP appears in your PROFILE RSK), then you should refrain from calling *PS_libinit()* because the RSK will do so as part of initializing those line drivers.

    If the latter is your situation, you can assume that the RSK has initialized the socket library as soon as control returns from the first START of the TCP or UDP line driver (e.g., SUBCOM START TCP in PROFILE RSK).

PS_libterm

Purpose

Terminates the RSK socket library.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 rc             fixed(31),
 re             fixed(31),

/* how to call */
call PS_libterm
(
 rc,                    /* 0:  return code           */
 re                     /* 0:  reason code           */
);
```

Parameters

Parameter
    Definition

rc
    Return code

re
    Reason code

Reason Codes

Reason Code
    Meaning

sok_re_success
    Function worked correctly

Usage Notes

1. You need to coordinate your use of *PS_libterm()* with the RSK's TCP and UDP line drivers.

The objective in such coordination is to make sure that you do not terminate the socket library prior to the RSK's being ready for it to be terminated.

If you plan **never ever** to use any of the IP functions in the RSK, you will definitely need to call *PS_libinit()* exactly once to terminate the RSK socket library, so you should go ahead and issue the call after you are all done issuing other RSK socket calls.

However, if your server starts the TCP or UDP line drivers (for example, SUBCOM START UDP appears in your PROFILE RSK), then you should refrain from calling *PS_libterm()* because the RSK will do so as part of terminating those line drivers.

The RSK will terminate the TCP and UDP line drivers only after all of your instance threads have terminated.

PS_listen

Purpose

Performs *listen()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname         char(8),
 socket           fixed(31),
 queuesize        fixed(31),
 rc               fixed(31),
 errno            fixed(31);

/* how to call */
call PS_listen
(
 applname,                /* I:  application name       */
 socket,                  /* I:  socket number          */
 queuesize,               /* I:  backlog queue size      */
 rc,                      /* O:  return code            */
 errno                    /* O:  ERRNO                  */
);
```

Parameters

Parameter
    Definition

applname
    Name of socket set

socket
    Socket number

queuesize
    Backlog queue size

rc
    Return code

errno
    Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_read

Purpose

Performs *read()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname          char(8),
 socket            fixed(31),
 bufpointer        pointer(31),
 bufsize           fixed(31),
 rc                fixed(31),
 errno             fixed(31);

/* how to call */
call PS_read
(
 applname,               /* I:  application name        */
 socket,                 /* I:  socket number           */
 bufpointer,             /* I:  pointer to read buffer  */
 bufsize,                /* I:  size of read buffer     */
 rc,                     /* O:  return code             */
 errno                   /* O:  ERRNO                   */
);
```

Parameters

Parameter
    Definition

applname
    Name of socket set

socket
    Socket number

bufpointer
    Pointer to read buffer

bufsize
    Size of read buffer

rc
    Return code

errno
    Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_recvfrom

Purpose

Performs *recvfrom()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname          char(8),
 socket            fixed(31),
 bufpointer        pointer(31),
 bufsize           fixed(31),
 flagword          fixed(31),
 addrbufptr        pointer(31),
 addrbufsize       fixed(31),
 rc                fixed(31),
 errno             fixed(31);

/* how to call */
call PS_recvfrom
(
 applname,                /* I:  application name       */
 socket,                  /* I:  socket number          */
 bufpointer,              /* I:  pointer to recv buffer */
 bufsize,                 /* I:  size of recv buffer    */
 flagword,                /* I:  flag word              */
 addrbufptr,              /* I:  pointer to sockaddr_in */
 addrbufsize,             /* I:  size of sockaddr_in    */
 rc,                      /* O:  return code            */
 errno                    /* O:  ERRNO                  */
);
```

Parameters

Parameter
  Definition

applname
  Name of socket set

socket
  Socket number

bufpointer
  Pointer to recv buffer

bufsize
  Size of recv buffer

flagword
  Flag word

addrbufptr
  Pointer to buffer to receive sockaddr_in

addrbufsize
  Size of buffer to receive sockaddr_in

rc
  Return code

errno
  Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

1. See the IUCV socket library documentation for definition of the flag word.

PS_select

Purpose

Performs *select()* function. Completion notification arrives as an IPC message in a CMS queue.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname          char(8),
 numinuse          fixed(31),
 rdptr             pointer(31),
 wrptr             pointer(31),
 exptr             pointer(31),
 toptr             pointer(31),
 nqhandle          fixed(31),
 nkpointer         pointer(31),
 nklength          fixed(31),
 xid               fixed(31),
 rc                fixed(31),
 errno             fixed(31);

/* how to call */
call PS_select
(
 applname,              /* I:  application name             */
 numinuse,              /* I:  sockets in use               */
 rdptr,                 /* I:  pointer to read descriptor    */
 wrptr,                 /* I:  pointer to write descriptor   */
 exptr,                 /* I:  pointer to exception descriptor */
 toptr,                 /* I:  pointer to timeval structure  */
 nqhandle,              /* I:  handle of notify queue        */
 nkpointer,             /* I:  pointer to notify key         */
 nklength,              /* I:  length of notify key          */
 xid,                   /* O:  transaction ID                */
 rc,                    /* O:  return code                   */
 errno                  /* O:  ERRNO                         */
);
```

Parameters

Parameter
> **Definition**

applname
> Name of socket set

numinuse
> Number of sockets named in descriptors

rdptr
> Pointer to read-interrogation descriptor

wrptr
> Pointer to write-interrogation descriptor

exptr
> Pointer to exception-interrogation descriptor

toptr
> Pointer to timeval structure

nqhandle
Handle of notify queue

nkpointer
Pointer to notify key

nklength
Length of notify key

xid
Returned transaction ID

rc
Return code

errno
Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

1. The handle for the notify queue must be a service ID. In other words, the queue in which the notification is to be placed must be a service queue. You must have already arranged for this by calling QueueIdentifyService.

2. The size of each descriptor in bytes, *fdsize*, is given by the formula 4 * ((numinuse+31)/32).

3. The notification message you see in the service queue will be the concatenation of your notify key and the following extra data:

   Offset.Length
   Usage

   0.4
   Return code

   4.4
   Errno

   8.8
   Unused

   16.*fdsize*
   Read-readiness descriptor

   16+*fdsize*.*fdsize*
   Write-readiness descriptor

   16+2*fdsize.*fdsize***
   Exception-readiness descriptor

4. The message will be sent with your notify key as its key.

5. If you need to cancel the operation before it completes, use the returned transaction ID in a call to *PS_cancel()*.

PS_sendto

Purpose

Performs *sendto()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname        char(8),
 socket          fixed(31),
 bufpointer      pointer(31),
 bufsize         fixed(31),
 flagword        fixed(31),
 addrbufptr      pointer(31),
 addrbufsize     fixed(31),
 rc              fixed(31),
 errno           fixed(31);

/* how to call */
call PS_sendto
(
 applname,              /* I:  application name      */
 socket,                /* I:  socket number         */
 bufpointer,            /* I:  pointer to send buffer */
 bufsize,               /* I:  size of send buffer   */
 flagword,              /* I:  flag word             */
 addrbufptr,            /* I:  pointer to sockaddr_in */
 addrbufsize,           /* I:  size of sockaddr_in   */
 rc,                    /* O:  return code           */
 errno                  /* O:  ERRNO                 */
);
```

Parameters

Parameter
    Definition

applname
    Name of socket set

socket
    Socket number

bufpointer
    Pointer to send buffer

bufsize
    Size of send buffer

flagword
    Flag word

addrbufptr
    Pointer to sockaddr_in describing recipient

addrbufsize
    Size of buffer to receive sockaddr_in

rc
    Return code

errno
    Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

1. See the IUCV socket library documentation for definition of the flag word.

PS_setsockopt

Purpose

Performs *setsockopt()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname        char(8),
 socket          fixed(31),
 level           fixed(31),
 optname         fixed(31),
 optvalptr       pointer(31),
 optvalbufsize   fixed(31),
 rc              fixed(31),
 errno           fixed(31);

/* how to call */
call PS_setsockopt
(
 applname,              /* I:  application name       */
 socket,                /* I:  socket number          */
 level,                 /* I:  level setting          */
 optname,               /* I:  option name            */
 optvalptr,             /* I:  pointer to value buffer */
 optvalbufsize,         /* I:  length of value buffer */
 rc,                    /* O:  return code            */
 errno                  /* O:  ERRNO                  */
);
```

Parameters

Parameter
   Definition

applname
   Name of socket set

socket
   Socket number

level
   Option level

optname
   Name of option being set

optvalptr
   Pointer to option value

optvalbufsize
   Size of option value

rc
   Return code

errno
   Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_shutdown

Purpose

Performs *shutdown()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname         char(8),
 socket           fixed(31),
 method           fixed(31),
 rc               fixed(31),
 errno            fixed(31);

/* how to call */
call PS_shutdown
(
 applname,              /* I:  application name       */
 socket,                /* I:  socket number          */
 method,                /* I:  shutdown method        */
 rc,                    /* O:  return code            */
 errno                  /* O:  ERRNO                  */
);
```

Parameters

Parameter
    Definition

applname
    Name of socket set

socket
    Socket number

method
    Shutdown method

rc
    Return code

errno
    Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

PS_socket

Purpose

Performs *socket()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
 applname         char(8),
 domain           fixed(31),
 type             fixed(31),
 protocol         fixed(31),
 socket           fixed(31),
 errno            fixed(31);

/* how to call */
call PS_socket
(
 applname,                /* I:  application name      */
 domain,                  /* I:  domain                */
 type,                    /* I:  type                  */
 protocol,                /* I:  protocol              */
 socket,                  /* O:  socket number         */
 errno                    /* O:  ERRNO                 */
);
```

Parameters

Parameter
   Definition

applname
   Name of socket set

domain
   Socket domain

type
   Socket type

protocol
   Protocol to use

socket
   Socket number

errno
   Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

1. Only domain AF_INET is supported.

PS_write

Purpose

Performs *write()* function.

PL/X Illustration

```
%include syslib(plxsock);

/* parameter data types */
declare
```

```
  applname           char(8),
  socket             fixed(31),
  bufpointer         pointer(31),
  bufsize            fixed(31),
  rc                 fixed(31),
  errno              fixed(31);

/* how to call */
call PS_write
(
 applname,                 /* I:  application name      */
 socket,                   /* I:  socket number         */
 bufpointer,               /* I:  pointer to write buffer */
 bufsize,                  /* I:  size of write buffer  */
 rc,                       /* O:  return code           */
 errno                     /* O:  ERRNO                 */
);
```

Parameters

Parameter
  Definition

applname
  Name of socket set

socket
  Socket number

bufpointer
  Pointer to write buffer

bufsize
  Size of write buffer

rc
  Return code

errno
  Returned ERRNO value

Reason Codes

Not applicable.

Usage Notes

None.

Appendix A. Sample PROFILE RSK

```
/*  */

/*********************************************************/
/*                                                       */
/* Sample Reusable Server Kernel profile file            */
/*                                                       */
/*********************************************************/

parse arg stuff
say 'Args were' stuff

/***************************/
/* first, config the server */
/***************************/

/***************************/
/* set names of data files  */
/***************************/

/* configure key data files */
'CONFIG SGP_FILE       MYSERV RSKSGP  A'
'CONFIG UMAP_FILE      MYSERV RSKUMAP A'

/* config auth data */
'CONFIG AUT_LOCATION   MINIDISK'
'CONFIG AUT_LOG        MYSERV  RSKAUL  B'
'CONFIG AUT_DATA_1     MYSERV1 RSKAUD  B'
'CONFIG AUT_INDEX_1    MYSERV1 RSKAUX  B'
'CONFIG AUT_DATA_2     MYSERV2 RSKAUD  B'
'CONFIG AUT_INDEX_2    MYSERV2 RSKAUX  B'

/***************************/
/* set other config vars    */
/***************************/

/* configure RSCS userid */
address command 'IDENTIFY ( LIFO'
parse pull . . . . rscsid .
'CONFIG RSCS_USERID' rscsid

/* configure monitor data */
'CONFIG MON_PRODUCT_ID MYSERVER'
'CONFIG MON_KERNEL_ROWS 50'

/* configure authorization database */
'CONFIG AUT_CACHE 100'
'CONFIG AUT_FREE 100'

/* configure AUTHCHECK family */
'CONFIG AUTHCHECK_AUTH ON'
'CONFIG AUTHCHECK_CACHE ON'
'CONFIG AUTHCHECK_CMS ON'
'CONFIG AUTHCHECK_CONFIG ON'
'CONFIG AUTHCHECK_CP ON'
'CONFIG AUTHCHECK_ENROLL ON'
'CONFIG AUTHCHECK_LD ON'
'CONFIG AUTHCHECK_SERVER ON'
'CONFIG AUTHCHECK_SGP ON'
'CONFIG AUTHCHECK_USERID ON'
'CONFIG AUTHCHECK_WORKER ON'

/* configure memory API */
'CONFIG MEM_MAXFREE 100'

/* set NOMAP actions */
'CONFIG NOMAP_TCP   OFF'
'CONFIG NOMAP_UDP   OFF'
'CONFIG NOMAP_MSG   OFF'
'CONFIG NOMAP_APPC  OFF'
'CONFIG NOMAP_IUCV  OFF'
'CONFIG NOMAP_SPOOL OFF'

/* configure MSG driver */
'CONFIG MSG_NOHDR OFF'
```

```
/* configure SPOOL driver */
'CONFIG SPL_INPUT_FT RSKRQST'
'CONFIG SPL_OUTPUT_FT RSKRESP'

/* configure implicit routing */
'CONFIG VM_CONSOLE ON'
'CONFIG VM_MSG ON'
'CONFIG VM_SPOOL ON'
'CONFIG VM_SUBCOM ON'

/****************************/
/* and start it            */
/****************************/

'RUNSERV'
if (rc<>0) then
 return 100

/*********************************************/
/* attach certain services to subcom driver */
/*********************************************/

'SUBCOM START WORKER'
'SUBCOM START USERID'
'SUBCOM START SERVER'
'SUBCOM START AUTH'
'SUBCOM START ENROLL'
'SUBCOM START SGP'
'SUBCOM START CMS'
'SUBCOM START CP'

'SUBCOM START TCP'
'SUBCOM START IUCV'
'SUBCOM START APPC'
'SUBCOM START SPOOL'
'SUBCOM START MSG'
'SUBCOM START CONSOLE'

/*********************************************/
/* attach certain services to console too   */
/*********************************************/

'CONSOLE START CACHE'
'CONSOLE START CONFIG'
'CONSOLE START USERID'
'CONSOLE START WORKER'
'CONSOLE START SERVER'
'CONSOLE START AUTH'
'CONSOLE START SGP'
'CONSOLE START CMS'
'CONSOLE START CP'
'CONSOLE START ENROLL'

'CONSOLE START TCP'
'CONSOLE START IUCV'
'CONSOLE START APPC'
'CONSOLE START SPOOL'
'CONSOLE START MSG'
'CONSOLE START SUBCOM'

/*********************************************/
/* and attach some to the MSG driver        */
/*********************************************/

'MSG START CACHE'
'MSG START CONFIG'
'MSG START USERID'
'MSG START SERVER'
'MSG START AUTH'
'MSG START SGP'
'MSG START CMS'
'MSG START CP'
'MSG START ENROLL'
'MSG START WORKER'

'MSG START TCP'
'MSG START SPOOL'
'MSG START MSG'
'MSG START SUBCOM'

/*********************************************/
```

```
/* start author-supplied services         */
/*********************************************/

/* for example... */
'TCP START MYSERV 500 10 0.0.0.0 TCPIP1'
'TCP START MYSERV 500 10 0.0.0.0 TCPIP2'
'TCP START MYSERV 500 10 0.0.0.0 TCPIP3'

'SUBCOM  START MYOP'
'CONSOLE START MYOP'
'MSG     START MYOP'

/*********************************************/
/* wait for server to end                  */
/*********************************************/

'WAITSERV'

/*********************************************/
/* perform server-specific termination here */
/*********************************************/

/*********************************************/
/* ... and return to caller                 */
/*********************************************/

return 0
```

Appendix B. Sample User ID Mapping File

```
*************************************************************************
*                                                                       *
*  Sample Reusable Server Kernel userid mapping file                    *
*                                                                       *
*  This file contains the mapping table that translates                 *
*  a two-token userid identifier to a single-token userid.              *
*                                                                       *
*                                                                       *
*  USAGE NOTES:                                                          *
*                                                                       *
*  1. File can be V-format or F-format, it doesn't matter.              *
*     LRECL doesn't matter, either.                                     *
*                                                                       *
*  2. Blank lines and lines starting with "*" are ignored.              *
*                                                                       *
*  3. If a ";" appears in the line, the ";" and everything              *
*     after the ";" are ignored.                                        *
*                                                                       *
*  4. Each clause must fit completely in one file record.               *
*                                                                       *
*  5. Case IS significant in this file.                                 *
*                                                                       *
*  6. The keyword in each clause must be in UPPER CASE.                  *
*                                                                       *
*  7. Unrecognized clauses are skipped without mention.                 *
*                                                                       *
*  8. The server kernel requires a userid mapping file to               *
*     be present.                                                       *
*                                                                       *
*                                                                       *
*  CLAUSE DEFINITION:                                                    *
*                                                                       *
*  Each clause is a record as follows:                                  *
*                                                                       *
*   MAP input_conn input_nodeid input_userid output_userid ; comment    *
*                                                                       *
*   where:                                                              *
*                                                                       *
*     MAP               is a literal identifying a mapping record       *
*     input_conn        is the input connectivity technology name       *
*     input_nodeid      is the input node ID                            *
*     input_userid      is the input user ID                            *
*     output_userid     is the output of translation                    *
*     comment           is an optional comment                          *
*                                                                       *
*   input_conn is one of:                                               *
*                                                                       *
*     TCP               describes a TCP/IP mapping                       *
*     UDP               describes a UDP/IP mapping                       *
*     IUCV              describes an IUCV mapping                        *
*     APPC              describes an APPC/VM mapping                      *
*     SPOOL             describes a SPOOL mapping                        *
*     MSG               describes a MSG mapping                          *
*     *                 applies to all technologies                      *
*                                                                       *
*  Notes:                                                               *
*                                                                       *
*  1.  The input fields are expressed in the same notation as queue     *
*      and event keys in CMS Application Multitasking, namely:          *
*                                                                       *
*      a. Case is significant,                                          *
*      b. "*" is a wildcard of 0 or more characters,                     *
*      c. "%" is a wildcard of exactly one character,                   *
*      d. "'" is an escape character.                                   *
*                                                                       *
*      For example, "GDLVM%" matches GDLVM1, GDLVM2, etc. but not       *
*      GDLVM50, and "GDL*" matches GDLVM1, GDLVM50, GDLAIX, etc.         *
*      WARNING: if you want "*", "%", or "'" to be a literal in         *
*      the field, precede it by the escape character '.                 *
*                                                                       *
*  2.  The output_userid field can be any literal or "=" to mean        *
*      "use the value of input_userid".                                 *
*                                                                       *
*  3.  The input fields can each be up to 64 bytes long.                *
```

```
*                                                                    *
*   4.  The output_userid field can be up to 64 bytes long.          *
*                                                                    *
*  Examples:                                                         *
*                                                                    *
*  MAP APPC      '*USERID:*       BKW   BKW                          *
*  MAP IUCV      GDLVM7           BKW   BKW                          *
*  MAP TCP       9.130.57.10      *     BKW                          *
*  MAP UDP       9.130.57.10      *     BKW                          *
*  MAP SPOOL     GDLVM7           BKW   BKW                          *
*  MAP MSG       GDLVMWEB         BKW   BKW                          *
*                                                                    *
*  In these examples, all of the following clients appear to be      *
*  userid BKW:                                                       *
*                                                                    *
*  - an IUCV-connected client coming from a virtual machine          *
*    whose userid is BKW                                             *
*                                                                    *
*  - an APPC/VM-connected client whose LU starts with "*USERID"      *
*    and whose security userid is BKW                                *
*                                                                    *
*  - a TCP/IP-connected client residing on machine 9.130.57.10       *
*                                                                    *
*  - a UDP/IP-connected client residing on machine 9.130.57.10       *
*                                                                    *
*  - a spool-connected client sending from BKW at GDLVM7             *
*                                                                    *
*  - a MSG-connected client sending from BKW at GDLVMWEB             *
*                                                                    *
*                                                                    *
*  SEARCH TECHNIQUE:                                                 *
*                                                                    *
*  The file is searched top to bottom, the first matching clause     *
*  being the one that takes effect.                                  *
*                                                                    *
**********************************************************************
```

Appendix C. Authorization Data File Formats

This appendix describes the internals of the files used to hold authorization data managed by the reusable server kernel. The information is provided so that vendors and toolsmiths might have a way to write management tools for these data files.

Overview

First, it's important to note that an *authorization data set* consists of a *data file* together with its corresponding *index file*. The data file contains records that define object classes, objects, users, and rules. The index file contains hash tables that let the reusable server kernel quickly locate specific objects' and specific users' information in the corresponding data file.

If the authorization data is being kept on minidisk, the reusable server kernel will keep twin copies of the authorization data set and will also keep a third kind of file, a *log file*, that lets it ensure consistency between an index file and its corresponding data file.[28] The reusable server kernel uses the log file to keep track of whether related changes are successfully applied to both an index file and its corresponding data file. The log file lets the reusable server kernel recover an authorization data set from its twin if a system failure should introduce some kind of integrity problem.

The authorization data files make heavy use of linked lists within the files themselves to relate records to one another. For example, all of the authorization rules applying to a given user are linked to one another, so that they may all be removed together by `ssAuthDeleteUser`. In all such linked lists, the linking is accomplished by file record number.

The Data File

The data file's role is to contain specific definitions of objects, users, classes, and rules. The data file is an F 300 file. Each record (or *row*) of a data file contains:

- A definition of an object class and a doubly-linked-list listhead that anchors all of the rows defining objects in this class, OR
- A definition of an object and a doubly-linked-list listhead that anchors all of the rows defining rules applying to this object, OR
- A definition of a user and a doubly-linked-list listhead that anchors all of the rules mentioning this user, OR
- A definition of a specific rule, that is, a correlation between an object, a user, and some subset of the actions defined on the class to which the object belongs, OR
- A stamp indicating that the row is free (unused) so that it might be allocated for another purpose at some time in the future.

One can see, then, that the relationship between object classes, objects, users, and actions is recorded by maintaining linkages among the records in the data file.

The following tables give the specific formats of each of the kinds of records found in the data file.

Table 49. Free Row

Offset	Length	Usage
0	4	X'00000000'
4	8	Unused

---

[28] The log file is unnecessary for SFS situations because the reusable server kernel just dedicates a work unit to the authorization data set.

Table 49. Free Row (continued)

Offset	Length	Usage
12	4	Row number of next free row

Table 50. Class Row

Offset	Length	Usage
0	4	X'00000001'
4	4	Row number of next class row
8	4	Row number of previous class row
12	4	Row number of first object in class
16	4	Row number of last object in class
20	4	Class identifier
24	8	Class name
32	4	Number of operations defined on class
36	128	Operation names (four bytes each)

Table 51. Object Row

Offset	Length	Usage
0	4	X'00000002'
4	4	Row number of first rule for object
8	4	Row number of last rule for object
12	4	Row number of next object in class
16	4	Row number of previous object in class
20	4	Row number of next object in object hash
24	4	Row number of previous object in object hash
28	4	Object ID
32	4	Class ID of class to which object belongs
36	4	Row number of said class's row
40	4	Length of object name
44	256	Object name

Table 52. User Row

Offset	Length	Usage
0	4	X'00000003'
4	4	Length of user ID
8	4	Unused
12	4	Row number of first rule for user
16	4	Row number of last rule for user

Table 52. User Row (continued)		
Offset	**Length**	**Usage**
20	4	Row number of next user in user hash
24	4	Row number of previous user in user hash
28	64	User ID

Table 53. Rule Row		
Offset	**Length**	**Usage**
0	4	X'00000004'
4	4	Row number of next rule for object
8	4	Row number of previous rule for object
12	4	Row number of next rule for user
16	4	Row number of previous rule for user
20	4	Row number of user row
24	4	Row number of object row
28	4	Length of user ID
32	64	User ID
96	4	Object ID
100	4	Operation count
104	128	Permitted operations (four bytes each)

The Index File

The index file, an F 4096 file, contains these three things:

- An *anchor row* that gives certain critical information about the authorization data set
- An *object hash* that lets the reusable server kernel find a given object's row quickly
- A *user hash* that lets the reusable server kernel find a given user's row quickly

The anchor row -- record 1 of the index file -- is described in .

Table 54. Anchor Row		
Offset	**Length**	**Usage**
0	4	Number of rows in data file
4	4	Row number of first class row in data file
8	4	Row number of last class row in data file
12	4	Row number of first free row in data file
16	4	Next class ID to use
20	4	Next object ID to use
24	4	Status bits (all zero when server down)

The object hash and user hash are each the same size. Each hash consists of 4096 buckets, numbered 1 to 4096. Each bucket consists of an eight-byte listhead - a *first row in hash* record number and a *last row in hash* record number. Thus each hash is 8 4096-byte records long. Records 2-9 are the object hash, and records 10-17 are the user hash.

To locate the row for a given object, the reusable server kernel hashes the object name to produce an integer *i* in the range [1,4096]. It then searches object hash bucket *i* for the object row nominating the object of interest. A similar hash-and-search procedure is used to find the row for a given user.

The Log File

When the authorization data sets reside on minidisk, the reusable server kernel maintains an F 256 *log file* that records updates that are in progress against an authorization data set's pair of files. The records in the log file are these:

- The *log stamp* row records which twin is known to be good and which twin has an update in progress. There is only one log stamp row in the log file and it is always record 1.

- A *log update* row lists a set of records in either an index file or a data file. Said list of records is in the process of being updated (rewritten).

The following tables give the organizations of these records.

Table 55. Log Stamp Row		
Offset	**Length**	**Usage**
0	4	Last known good authorization set (1 or 2)
4	4	Set against which an update is in progress
8	4	Number of update records following in log file

Table 56. Log Update Row		
Offset	**Length**	**Usage**
0	4	Data file (1) or index file (2) changes
4	4	Number of records being changed
8	248	Record numbers of records being changed (four bytes each)

The reusable server kernel performs log file updates, index file updates, data file updates, and file closes in a specific order which exploits the safety properties of the minidisk file system. The order of updates to these files is carefully controlled so that the files are always maintained on disk in a state from which the authorization database can be recovered even if there is an I/O failure.

The recovery algorithm is simple. When the reusable server kernel starts, it reads the first record of the log file to determine whether one of the twins was in the process of being updated when the files were last committed to disk. If one of the twins was being updated, the log update records tell which records were being rewritten. The reusable server kernel uses that list to restore the in-progress twin to a consistent state, merely copying the named records from the known-good twin to the in-progress twin. If the failing writes reflected a transaction that had already been performed against the known-good twin, the transaction will be propagated to the in-progress twin; if the failing writes reflected a transaction that had not yet been performed against the known-good twin, the transaction will be backed out. In this manner the in-progress twin is restored to a consistent state.

Appendix D. Enrollment Data File Format

An enrollment file is just a V-format CMS file, one file record per enrolled .

Columns
> **Usage**

1
> A for add, D for delete

2-65
> Record's key

66-end
> Record's data, if column 1 is A

When it loads the file into the data space, the reusable server kernel reads the file one record at a time, performing the operation specified in column 1. As API calls change the database, records are written to the end of the enrollment file, describing the API calls that took place. When the enrollment set is dropped, the file is closed with commit. If commit could not take place, the changes are backed out.

Appendix E. Storage Group File

The file containing storage group definitions is very simple. Each storage group is represented by one record. The first token of the record is the storage group number in decimal. The remaining tokens of the record are the hexadecimal virtual device numbers of the minidisks making up the storage group.

Appendix F. Reserved Names

The reusable server kernel uses several named CMS objects, such as storage subpools, mutexes, and the like. Further, in some cases the reusable server kernel uses named objects managed by its own entry points (for example, services registered through call to `ssServiceBind`).

The names of all CMS-managed objects used by the reusable server kernel start with the prefix BKW (case is not significant). Server authors should avoid this prefix.

Of course, CMS itself names objects with the prefixes DMS and VM, so these prefixes should be avoided as well.

Service Names

Specifically, the following service names are used:

Name
  Object

APPC
  APPC/VM line driver service name

AUTH
  Authorization data manipulation service

CACHE
  File cache manipulation service

CMS
  CMS command execution service name

CONFIG
  Configuration manipulation service

CONSOLE
  Console line driver service name

CP
  CP command execution service name

ENROLL
  Enrollment service name

IUCV
  IUCV line driver service name

MSG
  MSG/SMSG line driver service name

SERVER
  Server management service name

SPOOL
  Spool line driver service name

SUBCOM
  Subcom line driver service name

TCP
  TCP/IP line driver service name

TRIE
  Trie manipulation service

UDP
  TCP/IP line driver service name

USERID
  Userid mapping service name

WORKER
  Userid mapping service name

Data Spaces

The reusable server kernel creates data spaces whose names are of the form BKW@n, where *n* is the storage group number. It also creates data spaces whose names begin with BKW_.

TCP/IP Subtask Names

The TCP/IP line driver uses the IUCV interface to TCP/IP. When it connects to the TCP/IP service machine, it uses subtask names that are uppercase seven-digit hexadecimal numbers prefixed by T (that is, anything from T0000000 to TFFFFFFF).

UDP/IP Subtask Names

The UDP/IP line driver uses the IUCV interface to TCP/IP. When it connects to the TCP/IP service machine, it uses subtask names that are uppercase seven-digit hexadecimal numbers prefixed by U (that is, anything from U0000000 to UFFFFFFF).

Appendix G. More Detail On Reason Codes

gives the correspondence between numeric values of nonzero reason codes and their symbolic names. When an entry point (for example, `ssSgpStart`) gives you a nonzero reason code, use the table to interpret the reason code and devise a recovery strategy.

Table 57. Reason Codes and Recommended Actions

Numeric	Symbolic	Routine	Action
101	ss_uid_re_not_found	all	Add the appropriate mapping information to the user ID mapping file.
301	ss_aut_re_bad_count	all	Supply a valid option count or array length count.
302	ss_aut_re_bad_user_length	all	Supply a user ID length between 1 and 64 inclusive.
303	ss_aut_re_bad_obj_length	all	Supply an object length between 1 and 256 inclusive.
304	ss_aut_re_bad_option	all	Review the options array you supplied. One of the entries contains an unrecognized option code.
305	ss_aut_re_bad_qual	all	Review the qualifiers array you supplied. One of the entries contains a bad qualifier.
307	ss_aut_re_exists	all	The class or object you are trying to create already exists. Supply a different class name or object name.
308	ss_aut_re_no_class	all	The class to which you are referring does not exist. Supply a different class name.
309	ss_aut_re_no_object	all	The object to which you are referring does not exist. Supply a different object name.
310	ss_aut_re_maq_fail	all	A call by the server kernel to CSL routine `MutexAcquire` has failed. Contact IBM support.
311	ss_aut_re_cvw_fail	all	A call by the server kernel to CSL routine `CondVarWait` has failed. Contact IBM support.
312	ss_aut_re_cvs_fail	all	A call by the server kernel to CSL routine `CondVarSignal` has failed. Contact IBM support.
313	ss_aut_re_mr_fail	all	A call by the server kernel to CSL routine `MutexRelease` has failed. Contact IBM support.
314	ss_aut_re_too_many	ssAuthListClasses	There were more classes defined than your output array expected. Use a larger array.

Table 57. Reason Codes and Recommended Actions (continued)			
Numeric	**Symbolic**	**Routine**	**Action**
314	ss_aut_re_too_many	ssAuthListObjects	There were more objects defined than your output array expected. Use a larger array.
314	ss_aut_re_too_many	ssAuthModifyClass	Your call would result in exceeding the limit of 32 operations defined per object class. Use fewer operations.
314	ss_aut_re_too_many	ssAuthQueryObject	There were more user IDs defined than your output array expected. Use a larger array.
314	ss_aut_re_too_many	ssAuthQueryRule	There were more rules defined than your output array expected. Use a larger array.
316	ss_aut_re_no_user	all	The user ID you are attempting to locate does not exist in the authorization data. Try a different user ID.
317	ss_aut_re_prev_io_error	all	A previous I/O error to the authorization data base has taken it offline. Try `ssAuthReload`.
318	ss_aut_re_prev_sync_error	all	A previous error in calling one of CMS's synchronization routines (for example, `CondVarSignal`) has taken the authorization data base offline. Try `ssAuthReload`.
319	ss_aut_re_read_fail	all	An attempt to retrieve one or more records from one of the authorization data files has failed. This could mean either that an I/O error to one of the files has occurred or that there is insufficient storage to hold the records retrieved. Check for both conditions and respond accordingly.
320	ss_aut_re_write_fail	all	An attempt to write one or more records to one of the authorization data files has failed. This means an I/O error to one of the files has occurred. Check the file system and respond accordingly.
321	ss_aut_re_trunc	ssAuthListObjects	One or more returned object names was truncated. Use larger buffers.
321	ss_aut_re_trunc	ssAuthQueryObject	One or more returned user IDs was truncated. Use larger buffers.
322	ss_aut_re_gwu_fail	all	An attempt to get a CMS work unit has failed. Contact IBM support.

Table 57. Reason Codes and Recommended Actions (continued)			
Numeric	**Symbolic**	**Routine**	**Action**
323	ss_aut_re_open_fail	all	An attempt to open one of the authorization data files has failed. Check the AUT_ configuration parameters and the file system and respond accordingly.
601	ss_sgp_re_too_many	ssSgpList	There were more storage groups defined than your output array could hold. Use a larger array.
601	ss_sgp_re_too_many	ssSgpQuery	There were more minidisks defined than your output array could hold. Use a larger array.
602	ss_sgp_re_not_found	all	The storage group to which you are referring does not exist. Check the storage group identifier you are using (name or ID, as appropriate) and retry the operation.
603	ss_sgp_re_out_of_storage	all	There is insufficient storage to hold the control blocks necessary to represent the storage group. Use a larger virtual machine and try again.
604	ss_sgp_re_mx_fail	all	One of the server kernel's calls to the CSL mutex routines has failed. Contact IBM support.
607	ss_sgp_re_exists	all	The storage group you are attempting to create already exists. Use a different storage group number or delete the storage group first.
608	ss_sgp_re_vdq_fail	all	The server kernel's attempt to determine the attributes of one or more of the minidisks defined in your storage group has failed. You might have an incorrect device number or perhaps the minidisk is not linked. It is also possible that the minidisk is not formatted at 4 KB or that it has not been reserved. Check all of these conditions and try again.
609	ss_sgp_re_online	ssSgpDelete	You cannot delete this storage group because it is online right now. Take it offline (use `ssSgpStop`) and then retry the operation.
609	ss_sgp_re_online	ssSgpStart	The storage group is already started. Stop it first.
610	ss_sgp_re_offline	ssSgpStop	The storage group is already offline.
610	ss_sgp_re_offline	ssSgpWrite	The storage group is offline and therefore writes cannot happen. Bring the storage group online first.

Table 57. Reason Codes and Recommended Actions (continued)

Numeric	Symbolic	Routine	Action
612	ss_sgp_re_cv_fail	all	One of the server kernel's calls to the CSL condition variable routines has failed. Contact IBM support.
615	ss_sgp_re_ds_fail	all	The server kernel's attempt to create data spaces to map a storage group's minidisks has failed. Check your virtual machine's XCONFIG ADDRSPACE CP directory statement to ensure that you have not exceeded either the number of dataspaces limit or the aggregate storage size limit. Adjust the directory statement as appropriate. If you cannot adjust the directory statement, consider starting the storage group using DIAG X'0250' I/O instead.
616	ss_sgp_re_pool_fail	all	The server kernel's attempt to define the minidisk pool (MAPMDISK IDENTIFY) might have failed. If this happened, there should be a return and reason code on the virtual machine console. Research the return and reason code and act appropriately. This error can also be caused by insufficient storage. If this appears to be the cause, try increasing your virtual machine size.
617	ss_sgp_re_map_fail	all	The server kernel's attempt to map data space pages to minidisk blocks failed. There should be a MAPMDISK DEFINE return code on the virtual machine console. Contact IBM support.
618	ss_sgp_re_bad_attrib	all	The attribute array you supplied contains an unrecognized value. Repair the attribute array and try again.
619	ss_sgp_re_rewrite_fail	all	The server kernel's attempt to rewrite the file pointed to by configuration parameter SGP_FILE failed. Check to make sure the configuration value is correct and check to make sure the server virtual machine has the permissions necessary to write to the file.
620	ss_sgp_re_read_only	all	You asked to start the storage group read-write but one or more of the minidisks in the storage group is linked read-only. Change the link and try again, or start the storage group read-only.

Numeric	Symbolic	Routine	Action
622	ss_sgp_re_out_of_range	all	Some scalar parameter you supplied, such as a storage group number or the count of elements in an array, is out of range. Check your inputs and try again.
623	ss_sgp_re_wrong_mode	all	You attempted to write to the storage group but the storage group is started read-only. Stop the storage group and restart it in read-write mode or refrain from writing to the storage group.
624	ss_sgp_re_io_fail	all	If you started the storage group using DIAG X'00A4', you cannot specify an nonzero ALET value. If this is your situation, use zero for the value of your ALET. It is possible your virtual machine is out of storage. Try using a larger virtual storage size. Finally, it is possible that the real I/O failed. Check with your system programmer about whether the devices on which your minidisks reside have incurred some kind of failure. Be sure to tell the system programmer how you had started the storage group -- DIAG X'0250', DIAG X'00A4', or VM Data Spaces.
625	ss_sgp_re_diag_250_fail	all	You asked to use DIAG X'0250' as the I/O method for your storage group but the server kernel was unable to initialize the DIAG X'0250' environment. A return code of other than 0 or 4 was returned by DIAG X'0250' Initialize. Check the appropriate CP documentation and recover as necessary.
626	ss_sgp_re_too_big	all	The storage group you are attempting to start is too large - there are more than X'FFFFFFFF' 4 KB blocks in it. Use a smaller storage group.
628	ss_sgp_re_bad_name	all	You are attempting to start the storage group with an all-blank name. Supply a non-blank name for the name of the storage group. IBM recommends a printable EBCDIC name for storage groups.
629	ss_sgp_re_name_in_use	all	The storage group name you are trying to assign is already in use. Try a different storage group name.

Table 57. Reason Codes and Recommended Actions (continued)

Table 57. Reason Codes and Recommended Actions (continued)

Numeric	Symbolic	Routine	Action
701	ss_srv_re_bad_type	all	The service type you are supplying is unrecognized. Check your parameter list and try again.
702	ss_srv_re_not_found	all	The service you are trying to locate has not been bound. Check your RSKMAIN to be sure you called `ssServiceBind` and make sure you supplied the correct name in your call to `ssServiceFind`.
703	ss_srv_re_out_of_range	all	The service name length you supplied is out of range. Change the value to be within limits and try the API call again.
706	ss_srv_re_out_of_storage	all	There is not enough storage to hold the control blocks necessary to keep a record of the service. Increase your virtual storage size and try the server again.
709	ss_srv_re_exists	all	The service you are trying to bind already exists. Check your program to see whether you are calling `ssServiceBind` more than once, and check to see that you are supplying a unique service name each time. Check also to see whether you are trying to use one of the names IBM uses.
801	ss_mem_re_out_of_storage	all	There is not enough memory in the virtual machine or data space to satisfy your storage request. Use a larger virtual machine or a larger data space or be more economical in your use of storage.
802	ss_mem_re_bad_amount	ssMemoryAllocate	The storage size you supplied is out of range. Adjust the size and try again.
802	ss_mem_re_bad_amount	ssMemoryCreateDS	The size of the data space you are attempting to create is out of range. Adjust the data space size and try again.
802	ss_mem_re_bad_amount	ssMemoryRelease	The storage size you supplied is out of range. Adjust the size and try again.
803	ss_mem_re_bad_align	all	The alignment request you made in your call to `ssMemoryAllocate` is unrecognized. Specify one of the supported alignment types and try the API call again.
804	ss_mem_re_no_subpool	all	The subpool you named does not exist. Check the subpool name and try your API call again.

Numeric	Symbolic	Routine	Action
805	ss_mem_re_not_alloc	all	The storage you are attempting to release does not seem to be allocated. Check the storage pointer you are supplying and try the API call again.
807	ss_mem_re_spd_fail	all	The server kernel's call to SUBPOOL DELETE failed. Contact IBM support.
808	ss_mem_re_bad_key	all	The storage key you provide must be in the range [0,15]. Correct the error and try the API call again.
809	ss_mem_re_subpool_exists	all	The server kernel is already managing a subpool of this name. Change the subpool name to one that will be unique and try your API call again.
810	ss_mem_re_spcc_fail	all	The server kernel attempted to create a VM Data Space for you but could not do so. The virtual machine console should be displaying the return and reason code from CSL routine DMSSPCC. Interpret the return and reason code, correct the situation, and try again. The most likely reason for failure is that you have exceeded some limit imposed by the virtual machine's XCONFIG ADDRSPACE CP directory statement.
811	ss_mem_re_spla_fail	all	The server kernel attempted to establish addressability to a VM Data Space for you but could not do so. The virtual machine console should be displaying the return and reason code from CSL routine DMSSPLA. Interpret the return and reason code, correct the situation, and try again. The most likely reason for failure is that you have exceeded the limit imposed by the virtual machine's XCONFIG ACCESSLIST CP directory statement.
901	ss_cli_re_out_of_range	all	The amount of data you are attempting to put or get is out of range. Check your parameter list and try your API call again.
902	ss_cli_re_out_of_storage	all	There is insufficient storage to process your request to put data. Increase your virtual machine size and try your call again.
903	ss_cli_re_bad_iam	all	The caller type you specified is not one of the recognized caller types. Review your parameter list and try again.

Table 57. Reason Codes and Recommended Actions (continued)

Table 57. Reason Codes and Recommended Actions (continued)

Numeric	Symbolic	Routine	Action
904	ss_cli_re_bad_method	all	The byte retrieval method you specified is not one of the recognized retrieval methods. Review your parameter list and try again.
905	ss_cli_re_semc_fail	all	The server kernel performed a call to CSL routine SemCreate and the call failed. Contact IBM support.
1001	ss_enr_re_db_not_found	all	The enrollment data base you specified in your call does not exist. Check your parameter list and try your call again.
1002	ss_enr_re_rec_not_found	all	The enrollment record you requested does not exist. You might have specified the wrong record key, or you might be looking in the wrong enrollment data base. Check your parameter list and try again.
1003	ss_enr_re_truncated	all	The enrollment data you retrieved was truncated because your output buffer was not large enough. Change your program to specify a larger output buffer and try your call again.
1005	ss_enr_re_rec_exists	all	The record you tried to insert already exists. The enrollment record you specified on your call was replaced if you used method *ss_enr_insert_replace,* otherwise it was not replaced. Depending on your intentions, you may need to change your API call and try your call again.
1006	ss_enr_re_bad_length	ssEnrollLoad	The file name length you specified contains an invalid value. Change your parameter list and try your call again.
1006	ss_enr_re_bad_length	ssEnrollRecordGet	You specified an unacceptable length for the buffer in which the server kernel is to place the retrieved enrollment data. Change your parameter list and try your call again.
1006	ss_enr_re_bad_length	ssEnrollRecordInsert	You specified an unacceptable length for the data portion of the enrollment record you are attempting to insert. Change your parameter list and try your call again.
1007	ss_enr_re_bad_droptype	all	The parameter list you specified contains an unrecognized value for the drop type. Change your parameter list and try your API call again.

Numeric	Symbolic	Routine	Action
			Table 57. Reason Codes and Recommended Actions (continued)
1008	ss_enr_re_no_storage	ssEnrollLoad	There is not enough storage available to load the enrollment set. The data space containing the records is full. Unload the data base and reload it using a larger data space size.
1008	ss_enr_re_no_storage	ssEnrollRecordInsert	There is not enough storage available to insert the record. The data space containing the records is full. Unload the data base and reload it using a larger data space size.
1009	ss_enr_re_close_fail	all	The file backing the VM Data Space could not be closed. The changes made to the enrollment data base were backed out. Check into your SFS server to see whether it went down or the communication connection to it was severed (for example, VTAM® outage).
1010	ss_enr_re_write_fail	all	The server kernel's attempt to write to the enrollment file failed. Because the file is opened at load time and kept open, this write failure probably means some error has happened in the SFS server. Check with your system administrator.
1011	ss_enr_re_bad_method	all	The insertion method you specified in your parameter list was unrecognized. Check your parameter list and try your call again.
1012	ss_enr_re_open_fail	all	The server kernel's attempt to open the enrollment file failed. The name you specified might be incorrect, or the server might not have the permissions necessary to open the enrollment file for write, or the SFS server might not be operating. Check these things and try your call again.
1013	ss_enr_re_gwu_fail	all	The server kernel was not able to get a work unit on which to open the enrollment file. The return and reason code from DMSGETWU should have appeared on the virtual machine console. Investigate the return and reason code and take appropriate action.

Table 57. Reason Codes and Recommended Actions (continued)

Table 57. Reason Codes and Recommended Actions (continued)			
Numeric	**Symbolic**	**Routine**	**Action**
1014	ss_enr_re_point_fail	all	The server kernel was not able to move the file pointers for the enrollment file. The return and reason code from DMSPOINT should have appeared on the virtual machine console. Investigate the return and reason code and take appropriate action.
1015	ss_enr_re_exist_fail	all	The server kernel attempted to retrieve the attributes of the enrollment file but was not able to do so. The return and reason code from DMSEXIST should have appeared on the virtual machine console. Investigate the return and reason code and take appropriate action.
1016	ss_enr_re_not_sfs	all	The server kernel determined that the enrollment file does not reside in the Shared File System. Move the file to an SFS directory and try your call again.
1017	ss_enr_re_not_v	all	The server kernel determined that the enrollment file does not use V records. Change the file to V-format (use XEDIT, perhaps, or write a pipeline) and try your call again.
1018	ss_enr_re_dscr_fail	all	The server kernel was not able to create the data space needed to hold the enrollment records. It is possible that some limit associated with XCONFIG ADDRSPACE was violated; check these limits and retry. It's also possible that the enrollment set name you used is already in use as a subpool for some other purpose. If this is the case, choose a different enrollment set name.
1019	ss_enr_re_read_fail	all	The server kernel was unable to read the enrollment file. Because the server kernel's call to DMSOPEN worked, this probably indicates an SFS error of some kind. Check the health of the SFS server and try your call again.
1020	ss_enr_re_db_exists	all	The enrollment set you are attempting to load already exists. Choose a different name and try your call again. If you meant to reload the enrollment set, drop the set first and then load it again.

Table 57. Reason Codes and Recommended Actions (continued)			
Numeric	**Symbolic**	**Routine**	**Action**
1021	ss_enr_re_comm_fail	all	The server kernel's attempt to commit the changes to the enrollment set has failed. The most likely cause is that the enrollment set has grown so large that the filespace limit has been exceeded - your SFS administrator might have to issue `MODIFY USER` before your commit will work. The return and reason code from `DMSCOMM` are displayed on the virtual machine console. Investigate the return and reason code and take appropriate corrective action.
1022	ss_enr_re_not_disk	all	You tried to commit changes to a transient enrollment set. Because a transient enrollment set has no backing file in the Shared File System, you cannot commit its changes. Use a permanent enrollment set instead of a transient one.
1023	ss_enr_re_bad_kind	all	The *set_kind* parameter you specified contains an unrecognized value. Change your parameter list and try your call again.
1024	ss_enr_re_new_file	all	The file you nominated doesn't exist, so the server kernel created it and initialized the enrollment set as empty. If you did not expect this result, check the file name you supplied and try your call again.
1025	ss_enr_re_no_sets	all	There are no enrollment sets loaded. If you didn't expect this, check your program to see whether you forgot to load your enrollment set or whether you dropped the enrollment set unknowingly.
1026	ss_enr_re_set_empty	all	The enrollment set you interrogated contains no records. If you didn't expect this, check to make sure you loaded the correct SFS file.
1501	ss_cac_re_out_of_storage	all	There is insufficient storage available to process your cache request. Increase your virtual machine's storage size.
1502	ss_cac_re_table_replaced	all	You submitted a translation table *n* when there was already a table known by that number. If you did not expect this result, check your parameter list and the other `ssCacheXlTabSet` calls your server has performed.

Numeric	Symbolic	Routine	Action
1503	ss_cac_re_cache_not_found	all	The cache you are attempting to use does not exist. Check to be sure the cache was created.
1504	ss_cac_re_dscr_fail	all	The server kernel attempted to create a VM Data Space to hold the cached files but was not able to create it. The most likely cause here is that you have exceeded some limit set by XCONFIG ADDRSPACE. Check your CP directory entry, issue CP QUERY SPACES, compare the two, and make a configuration change if necessary.
1505	ss_cac_re_cache_exists	all	The cache you are trying to create already exists. Delete the cache before recreating it, or change your parameter list to specify a different cache name.
1506	ss_cac_re_bad_size	all	The cache size you specified is out of range. Check your parameter list against the documentation to see whether your cache size is in range. The cache size is specified in pages.
1511	ss_cac_re_bad_token	all	The file token you supplied is not recognized. Check your parameter list to be sure that the token you are providing is one that was given to you by ssCacheFileOpen.
1512	ss_cac_re_bad_length	ssCacheFileOpen	The file name length you supplied is unacceptable. Check to be sure the length is in range. Correct your parameter list and try your call again.
1512	ss_cac_re_bad_length	ssCacheFileRead	The byte count you supplied is out of range. Check your parameter list and try your call again.
1513	ss_cac_re_bad_count	all	The *flag_count* value you supplied is out of range. Correct your parameter list and try your call again.
1514	ss_cac_re_bad_esmdl	all	The ESM data length you supplied is unacceptable. Check your parameter list and make the appropriate correction.
1515	ss_cac_re_bad_fname	all	One of the flag names you specified in your flag name array is unrecognized. Check your flag name array and try your call again.
1516	ss_cac_re_bad_fval	all	One of the flag values you specified in your flag value array is unrecognized. Check your flag value array and try your call again.

Table 57. Reason Codes and Recommended Actions (continued)

Numeric	Symbolic	Routine	Action
Table 57. Reason Codes and Recommended Actions (continued)			
1517	ss_cac_re_exist_fail	all	The server kernel's call to DMSEXIST failed. The return and reason code from DMSEXIST can be found on the virtual machine console. Investigate the return and reason code and try your call again.
1518	ss_cac_re_file_not_found	all	The server kernel was not able to find the file you are trying to cache. Check the file name to be sure it is what you intended, and then try your call again.
1519	ss_cac_re_delete_in_progress	all	The server kernel was not able to cache the file you specified because the cache you specified is in the process of being deleted. Use a different cache to cache the file.
1520	ss_cac_re_bad_offset	all	The byte offset you specified is negative or goes beyond the last byte of the file. Correct your parameter list.
1521	ss_cac_re_bad_table_id	all	The table ID you specified was zero. Zero is a reserved table identifier. Specify any non-zero table identifier.
1522	ss_cac_re_table_not_found	all	The translation table you requested in your call to ssCacheFileOpen does not exist. Check your parameter list to see if you used the table ID you intended, or check to see that you did not omit a call to ssCacheXlTabSet.
1523	ss_cac_re_open_fail	all	The server kernel was not able to open the file you wanted to cache. The return and reason code from DMSOPEN are displayed on the virtual machine console. Investigate the return and reason code and take appropriate action.
1524	ss_cac_re_bad_recfm	all	The file you wanted to cache has a record format other than F or V. The server kernel cannot cache it. Change the file's record format and try the call again.
1526	ss_cac_re_out_of_storage_ds	all	There is not enough free storage in the data space to cache your file. Create a larger file cache and try your operation again.
1527	ss_cac_re_read_fail	all	The server kernel was able to open the file being cached but could not read it. The return code and reason code from DMSREAD appear on the virtual machine console. Investigate the return and reason code and try the call again.

Numeric	Symbolic	Routine	Action
1528	ss_cac_re_bad_data_stream	all	The server kernel was looking for record delimiters in the data of a CMS file (SFS, minidisk, or BFS) but did not find them. The probable cause is that there is a run of more than 65,535 bytes without a delimiter - in other words, some record in the file is too long. Change the file and try again.
1601	ss_wrk_re_out_of_storage	all	The server kernel was unable to allocate storage to hold information related to your connection to a worker. Increase your virtual storage size.
1602	ss_wrk_re_bad_count	all	You supplied a less-than-zero option count. Fix your API call and try again.
1603	ss_wrk_re_bad_flag_name	all	One of the flag names you supplied in your parameter list is incorrect. Inspect the parameter list you built and try again.
1604	ss_wrk_re_bad_flag_value	all	One of the flag values you supplied in your parameter list is incorrect. Inspect the parameter list you built and try again.
1605	ss_wrk_re_no_class	all	The worker class you specified in your call is not defined. Inspect your parameter list and try your call again, or inspect PROFILE RSK to see whether you misspelled or omitted the WORKER commands necessary to create your worker machine class.
1606	ss_wrk_re_no_subordinates	all	The server kernel tried to allocate a connection for you to a worker machine but could not do so. Either all of the workers are full or the non-full ones didn't answer (autologging failed, IUCV connections failed, or some other indeterminate failure happened).
1607	ss_wrk_re_algtries_exceeded	all	The server kernel tried repeatedly to autolog a worker machine but the worker did not answer IUCV connection requests. Check your workers' configurations and try the server again.
1608	ss_wrk_re_autolog_fail	all	The server kernel tried to autolog a worker machine but the XAUTOLOG command failed. The server virtual machine probably has insufficient CP privilege to use the XAUTOLOG command. Check the configuration and try again.

Table 57. Reason Codes and Recommended Actions (continued)

Table 57. Reason Codes and Recommended Actions *(continued)*			
Numeric	**Symbolic**	**Routine**	**Action**
1609	ss_wrk_re_timer_fail	all	The server kernel tried to use CMS's Timer API to set a timer but was not able to do so. Contact IBM support.
1610	ss_wrk_re_iucvcon_fail	all	The server kernel tried to IUCV CONNECT to a worker machine but encountered some kind of permanent error, such as the worker not having IUCV ALLOW in its CP directory entry. Check your worker machine configurations and try again.
1611	ss_wrk_re_force_fail	all	The server kernel tried to issue the CP FORCE command to force a worker machine but was unable to do so. The most likely cause is that the server virtual machine has insufficient CP privilege to use the FORCE command. Check the server virtual machine's CP directory entry and try again.
1612	ss_wrk_re_force_timeout	all	The server kernel issued the CP FORCE command to force off a worker and began waiting for the worker machine to be logged off, but after a timeout period the CP QUERY command showed that the worker was still logged on. The most likely cause is that the worker machine is a hung user.
1613	ss_wrk_re_oper_delete	all	Your program attempted to allocate a connection to a worker machine, but while the connection was being established an operator used the WORKER DELETE or WORKER DELCLASS command to delete the worker machine. Your connection attempt failed.
1701	ss_tri_re_bad_size	all	The trie size you specified is out of range. Check your parameter list against the documentation to see whether your size is in range. The trie size is specified in pages.
1702	ss_tri_re_trie_exists	all	You are trying to create a trie but it already exists. Choose a different trie name or delete the previous instance of the trie.
1703	ss_tri_re_out_of_storage	all	There is not enough primary storage (memory) to create your trie. Run your server in a larger virtual machine.

Table 57. Reason Codes and Recommended Actions (continued)			
Numeric	**Symbolic**	**Routine**	**Action**
1704	ss_tri_re_dscr_fail	all	Creation of the trie's data space failed. You probably have created too many data spaces or the total size of your data spaces would be too large. Check your server and its XCONFIG ADDRSPACE CP directory entry and make any needed corrections.
1705	ss_tri_re_trie_not_found	all	The trie you are attempting to manipulate does not exist. Check the name your are using and try again.
1706	ss_tri_re_trie_busy	all	The server kernel was unable to acquire your trie's lock in a reasonable period of time. Perhaps the trie is shared among many virtual machines and the lock holder has abended or logged off unexpectedly. Re-IPL your set of servers.
1707	ss_tri_re_bad_index_len	all	The index you supplied has an incorrect length. Correct the index length and try the API call again.
1708	ss_tri_re_bad_capacity	all	The array capacity you supplied is incorrect. Correct the value and try the API call again.
1709	ss_tri_re_out_of_ds_storage	all	There is no room left in the trie's data space. No more indices can be added. Create the trie with a larger size.

Appendix H. Messages

Here is a summary of messages and recommended recovery actions.

Generally Applicable Messages

BKW0000I **Operation completed OK.**

Explanation

The command you issued completed normally.

System action

The system performed the action you requested.

System programmer response

Nothing.

BKW0001E **Not authorized.**

Explanation

You are not authorized to issue the command you attempted.

System action

The system declined to execute the command you supplied, responding with this error message instead.

System programmer response

The system programmer can use the AUTH command set to grant you permission to perform the requested operation.

BKW0002E **Enter a command.**

Explanation

You entered a null command.

System action

The system did nothing.

System programmer response

Enter a non-null command.

BKW0003E **Syntax error.**

Explanation

There is a syntax error in the command you issued.

System action

The system did nothing.

System programmer response

Refer to the syntax diagram for the command you issued, repair its syntax, and reissue the command.

BKW0004E **Unrecognized command.**

Explanation

The command you entered is not recognized.

System action

The system did nothing.

System programmer response

Refer to the command documentation and submit a recognized command.

BKW0005E **Out of storage.**

Explanation

Not enough virtual storage was available to perform the operation you requested.

System action

The system backed out any partial results and returned to the state it had just prior to your issuing the failing command.

System programmer response

Define a larger virtual machine.

BKW0007E **RC=&1 RE=&2 from routine &3**

Explanation

The displayed routine produced the given return and reason code.

System action

The system did not complete the operation you requested.

System programmer response

Locate the documentation for the displayed routine and research the return and reason code. Take appropriate corrective action.

BKW0010E	DMSQEFL returns CP_product &1 CP_level &1

Explanation

CSL routine DMSQEFL returned the displayed CP product code and CP level code.

System action

The server kernel refuses to start because CP is too far back-level.

System programmer response

Upgrade to a newer release of z/VM.

BKW0011E	DMSQEFL returns CMS_level &1

Explanation

CSL routine DMSQEFL returned the displayed CMS level.

System action

The server kernel refuses to start because CMS is too far back-level.

CONFIG Service Messages

BKW0100E	Operation now irrelevant.

Explanation

The configuration variable whose value you changed is relevant only before `PROFILE RSK` issues RUNSERV. After RUNSERV, the server kernel no longer pays attention to the value of this variable.

System action

The system did nothing.

System programmer response

Change this configuration variable before RUNSERV.

System programmer response

Upgrade to a newer release of z/VM.

BKW0012E	Insufficient VM/ESA functional level to run RSK - returning

Explanation

The level of VM/ESA is insufficient to support execution of the reusable server kernel.

System action

The server kernel refuses to start.

System programmer response

Upgrade to a newer release of z/VM.

BKW0013I	CMS 13 detected - ensure VM61422 is applied

Explanation

The reusable server kernel detected CMS 13. For best results, CMS 13 must have the displayed APAR applied. The server kernel will work if the APAR is not applied but it might not work well.

System action

The server kernel starts anyway.

System programmer response

Install the named APAR for best results (the message will still appear even after the APAR is applied).

Line Driver Messages

BKW0200E   **Service not found.**

Explanation

The service you are attempting to manipulate does not exist.

System action

The system did nothing.

System programmer response

Correct the name of the service, or use the SERVER SERVICES command to determine whether the service is known to the server kernel.

BKW0201E   **Subtask not found.**

Explanation

The subtask you attempted to manipulate does not exist.

System action

The system did nothing.

System programmer response

Use the line driver's LIST command to confirm the existence of the subtask you are attempting to manipulate. Also, confirm that you have supplied the correct line driver name in your command. Make appropriate corrections and resubmit the command.

BKW0202E   **Stop of self is prohibited.**

Explanation

You asked a self-sourced line driver to stop itself. A self-sourced driver cannot stop itself.

System action

The system did nothing.

System programmer response

You probably meant to stop some other subtask. Correct the subtask number and try again.

BKW0203I   **Subtask asked to STOP.**

Explanation

The line driver has sent STOP messages to the threads running this subtask.

System action

The subtask will stop when all such threads respond with stop acknowledgements.

System programmer response

Wait for the subtask to stop.

BKW0204I   **Subtask killed.**

Explanation

The line driver has deleted the threads of the subtask.

System action

The server kernel has stopped a subtask in a forceful way. Threads running the service were not given an opportunity to complete their work normally.

System programmer response

Nothing.

BKW0205E   **Prefix already in use.**

Explanation

The prefix you requested is already in use by this line driver.

System action

The system did nothing.

System programmer response

Select a different prefix and reissue the command.

BKW0206E   **Service INIT routine failed - RC=&1 RE=&2.**

Explanation

During handling of a START command, the server kernel drove the service's INIT routine but the INIT routine produced a nonzero return and reason code.

System action

The system refused to start the service.

System programmer response

Use the documentation of the service itself to interpret the return and reason code. Take appropriate corrective actions and try the START again.

BKW0207E   **Start of self is prohibited.**

Explanation

You asked a self-sourced line driver to start itself.

System action

The system refused to do this. The server kernel starts self-sourced line drivers automatically as part of server initialization.

System programmer response

You probably submitted the START command to the wrong service or attempted to start the wrong service.

SERVER Service Messages

BKW0300I   **Shutdown initiated.**

Explanation

You issued SERVER STOP and the server kernel is attempting to stop the server.

System action

The line drivers are attempting to stop all services normally. When all services are stopped shutdown of the server will complete.

System programmer response

None needed.

BKW0301I   **Monitor buffer at &1.&2, &3 rows, &4 free**

USERID Service Messages

BKW0400E   **Reload failed - DMSOPEN or DMSREAD RC=&1 RE=&2.**

Explanation

The server kernel was not able to reload the user ID mapping file because either DMSOPEN or DMSREAD failed with the displayed return and reason code.

System action

The previous user ID mapping remains in effect.

Make the appropriate corrections in your command and issue it again.

BKW0208I   **Subtask is handling no clients.**

Explanation

The subtask you attempted to interrogate through QUERY is not handling any clients right now.

System action

The system did nothing.

System programmer response

None needed.

Explanation

The message indicates the location in storage of the server kernel's monitor buffer.

System action

None, other than having issued the message.

System programmer response

None needed. The CP DISPLAY command can be used to display the monitor buffer. The MONITOR DISPLAY command can be used to display specific monitor rows without knowing their addresses in memory.

System programmer response

Research the return and reason code and take the appropriate action. Also, issue SERVER CONFIG and look at the value of the UMAP_FILE variable and see if it references the file you expected.

BKW0401I   **&1 &2 &3 maps to &4**

Explanation

The user ID mapping facility maps your inputs to this output.

System action

None, other than displaying the mapping.

System programmer response

If the mapping needs to be corrected, use XEDIT to change the mapping file, then issue USERID RELOAD.

BKW0402E   **RC=&1 RE=&2 mapping &3 &4 &5**

Explanation

`ssUseridMap` produced the displayed return and reason code when interrogating the user ID map with the inputs you provided.

System action

None, other than displaying the error message.

System programmer response

Research the return and reason code and take appropriate corrective action. If you need to update the user ID map, edit the mapping file and issue USERID RELOAD.

BKW0403E   **Open of UMAP_FILE failed - server will not start.**

Explanation

The server kernel attempted to read the user ID mapping file as part of its startup processing, but was not able to read the file.

System action

Startup fails and the RUNSERV command will complete with a nonzero return code.

TCP and UDP Line Driver Messages

BKW0500I   **A-block &1 Client &2 &3 done, lifetime &4 msec**

Explanation

A TCP or UDP subtask has finished handling the client at the displayed port and IP address. The transaction lasted for the displayed number of milliseconds.

System programmer response

The configuration variable UMAP_FILE is probably not set correctly. Make sure it points to the user ID mapping file and then try again to start the server.

BKW0404E   **Reload ignored some records due to syntax errors**

Explanation

The server kernel attempted to reload the user ID mapping file, but while reading the file it found some records having invalid syntax.

System action

The load finished, ignoring the bad records. Message BKW0405E was issued for each bad record.

System programmer response

Use the record numbers named in message BKW0405E to locate to locate the bad records. Repair each one.

BKW0405E   **Record &1 in UMAP_FILE has bad syntax**

Explanation

The server kernel found a bad record in the user ID mapping file. This message announces the record number of the bad record.

System action

The server kernel skipped the bad record and continued to load the user ID mapping file.

System programmer response

Repair the bad record.

System action

The system handled the client.

System programmer response

None.

BKW0501I   **A-block &1 Client &2 &3 done, inbytes &4, inrate &5 KB/s**

Explanation

A TCP or UDP subtask has finished handling the client at the displayed port and IP address. The data rate from the client was as displayed.

System action

The system handled the client.

System programmer response

None.

BKW0502I	A-block &1 Client &2 &3 done, outbytes &4, outrate &5 KB/s

Explanation

A TCP or UDP subtask has finished handling the client at the displayed port and IP address. The data rate to the client was as displayed.

System action

The system handled the client.

System programmer response

None.

BKW0504I	A-block &1 Client &2 &3 started, C-block &4

Explanation

A TCP or UDP subtask has begun handling the client at the displayed port and IP address.

System action

The system is beginning to handle the client.

System programmer response

None.

BKW0505E	A-block &1 errno &2 accept failed

Explanation

The TCP line driver received the displayed *errno* value when it attempted to accept a connection from a client.

System action

The line driver did not accept the connection but continues handling work for other clients.

System programmer response

Research the *errno* and determine whether a configuration change is necessary.

BKW0506E	A-block &1 C-block &2 errno &3 ioctl(FIONBIO) failed

Explanation

The TCP line driver received the displayed *errno* value when it attempted to set a socket to blocking I/O.

System action

The line driver closed the connection to the client but continues handling work for other clients.

System programmer response

Research the *errno* and determine whether a configuration change is necessary.

BKW0508E	A-block &1 C-block &2 ThreadCreate RC=&3 RE=&4 failed (major)

Explanation

The TCP or UDP line driver was not able to create a CMS thread when one was absolutely required.

System action

The line driver ended the subtask.

System programmer response

Research the return and reason code and take corrective action.

BKW0509E	A-block &1 C-block &2 ThreadCreate RC=&3 RE=&4 failed (minor)

Explanation

The TCP or UDP line driver was not able to create a CMS thread when it felt one would be helpful, but there appear to be enough suitable threads to take up the slack.

System action

The line driver uses the threads it's already created to handle the new client.

System programmer response

Research the return and reason code and take corrective action.

BKW0510E **A-block &1 errno &2 select()-start failed**

Explanation

The TCP line driver was not able to start a socket *select()* function.

System action

The line driver stops the affected subtask. Clients already connected are permitted to complete their transactions, but no new clients are served.

System programmer response

Research the *errno* and take corrective action.

BKW0511E **A-block &1 rsn &2 QueueReceiveBlock RC=&3 RE=&4 failed**

Explanation

The TCP or UDP line driver was not able to receive a message from a CMS queue.

System action

The line driver stops the affected subtask immediately.

System programmer response

Re-IPL CMS. If the problem persists, contact IBM support.

BKW0512E **A-block &1 errno &2 select() failed**

Explanation

The TCP line driver started a socket *select()* function but the function completed with error.

System action

The line driver stops the affected subtask. Clients already connected are permitted to complete their transactions, but no new clients are served.

System programmer response

Research the *errno* and take corrective action.

BKW0513E **Port number must be in range [0..65535].**

Explanation

Your START command specified an out-of-range port value.

System action

None, other than issuing an error message.

System programmer response

Correct your START command and try again.

BKW0514E **Socket count must be in range [50..2000].**

Explanation

Your START command specified an out-of-range value for the number of sockets permitted.

System action

None, other than issuing an error message.

System programmer response

Correct your START command and try again.

BKW0515E **Maximum subtask number would be exceeded.**

Explanation

The TCP or UDP line driver was not able to start a new subtask because it has run out of subtask numbers.

System action

The subtask was not started.

System programmer response

Restart the server.

BKW0516E **Creation of subtask controller thread failed.**

Explanation

The TCP or UDP line driver attempted to create a thread to control the new subtask but was not able to do so.

System action

The subtask was not started.

System programmer response

Re-IPL CMS. If the problem persists, contact IBM support.

BKW0517E **Creation of TCP/IP socket group failed.**

Explanation

The TCP or UDP line driver was not able to connect to the TCP/IP service machine.

System action

The subtask was not started.

System programmer response

The usual cause here is that the name of the TCP/IP machine was specified incorrectly. Another cause might be that the TCP/IP machine you are attempting to use is configured with `PermittedUsersOnly` but your server is not in the permitted users list. Check your START command and your TCP/IP configuration carefully and try your command again.

BKW0518E **Creation of listen socket failed.**

Explanation

The TCP or UDP line driver was not able to create the socket on which it will listen for connections from clients.

System action

The subtask was not started.

System programmer response

Check your TCP/IP configuration.

BKW0519E **Setting listen socket to SO_REUSEADDR failed.**

Explanation

The TCP or UDP line driver was not able to set the listen socket to enable option SO_REUSEADDR.

System action

The subtask was not started.

System programmer response

Check your TCP/IP configuration.

BKW0520E **Setting listen socket to nonblocking failed.**

Explanation

The TCP line driver was not able to set the listen socket to non-blocking I/O.

System action

The subtask was not started.

System programmer response

Check your TCP/IP configuration.

BKW0521E **bind() for listen socket failed.**

Explanation

The TCP or UDP line driver was not able to bind the port number you specified in your START command to the IP address you specified in your START command.

System action

The subtask was not started.

System programmer response

The most likely cause is that the port number is in the reserved port number list in your TCP/IP configuration but the user ID in which your server is running is not listed as one of the user IDs that can bind the reserved port. Check your TCP/IP configuration and try again if this was the situation. Another possible cause is that some other server on your system has already bound that port but did not set its listen socket to SO_REUSEADDR. If this is the case, contact your TCP/IP support programmer for help in locating the offending server, or use another port number in your own START command.

BKW0522E **listen() for listen socket failed.**

Explanation

The TCP line driver was not able to set the backlog queue size for its listen socket.

System action

The subtask was not started.

System programmer response

Check your TCP/IP configuration.

BKW0523I **Instance STOP requested.**

Explanation

In response to your STOP command, the TCP or UDP line driver has asked an instance thread to stop.

System action

The line driver will close the connection to the client after the instance acknowledges the STOP request.

System programmer response

None.

BKW0524E **Wait expired for STOP.**

Explanation

You asked the TCP or UDP line driver to stop a subtask, so it initiated the stop and waited for the subtask to quiesce, but the quiesce wait time ran out.

System action

The stop did not complete.

System programmer response

The stop remains pending and will complete eventually if all of the instance threads cooperate. If you require the subtask to stop immediately, reissue the command using the NOW option.

BKW0525E **A-block &1 C-block &2 read start failed - errno &3**

Explanation

The TCP line driver was not able to start a socket *read()* for the displayed client, or the UDP line driver was not able to start a socket *recvfrom()*.

System action

The TCP line driver closes the connection to the client; the UDP line driver ends the subtask.

System programmer response

Check your TCP/IP configuration.

BKW0526E **A-block &1 C-block &2 write start failed - errno &3**

Explanation

The TCP line driver was not able to start a socket *write()* for the displayed client, or the UDP line driver was not able to start a socket *sendto()*.

System action

The TCP line driver closes the connection to the client; the UDP line driver ends the subtask.

System programmer response

Check your TCP/IP configuration.

BKW0527I **A-block &1 stopped.**

Explanation

You asked the TCP or UDP line driver to stop a subtask.

System action

The subtask has stopped.

System programmer response

None.

BKW0528I **A-block &1 C-block &2 stopped.**

Explanation

You asked the TCP or UDP line driver to end its relationship with a specific client.

System action

The relationship is ended.

System programmer response

None.

BKW0529I **Subtask identifier is out of range.**

Explanation

You asked the TCP or UDP line driver to stop a subtask whose identifier is zero.

System action

None, other than to issue an error message.

System programmer response

Specify a nonzero subtask identifier.

BKW0530E **A-block &1 C-block &2 recv failed - errno &3**

Explanation

The UDP line driver attempted to receive a datagram using *recvfrom()*, but the call failed.

System action

The UDP line driver stops the subtask and displays the *errno* value it encountered.

System programmer response

Research the *errno* value and restart the subtask.

BKW0531E	A-block &1 C-block &2 sendto failed - errno &3

Explanation

The UDP line driver attempted to send a datagram using *sendto()*, but the call failed.

System action

The UDP line driver stops the subtask and displays the *errno* value it encountered.

SGP Service Messages

BKW0600I	No storage groups found.

Explanation

Your LIST command found no storage groups.

System action

None, other than issuing the error message.

System programmer response

None. If you expected to find storage groups, use the SERVER CONFIG command to check the value of configuration variable SGP_FILE. You might have specified the wrong file name.

BKW0601E	Open of SGP_FILE failed - server will not start.

RSK SUBCOM Messages

BKW0700E	Commands cannot be issued - server not started yet

Explanation

Your PROFILE RSK contains commands other than CONFIG before RUNSERV.

System action

The non-CONFIG commands are ignored.

System programmer response

Research the *errno* value and restart the subtask.

BKW0532E	No userid mapping for IP address &1 - ignored

Explanation

The TCP or UDP line driver attempted to map an IP address to a user ID but was not able to do so.

System action

Because the line driver's NOMAP configuration parameter was OFF, the line driver ignored the client.

System programmer response

Update the user ID mapping file or set the line driver's NOMAP parameter ON.

Explanation

The server kernel could not find the storage group configuration file.

System action

The server kernel will not start and the RUNSERV command will see a nonzero return code.

System programmer response

Check your PROFILE RSK to make sure you set configuration variable SGP_FILE correctly.

System programmer response

Reorganize your PROFILE RSK.

BKW0701E	The server has already been started

Explanation

You attempted RUNSERV more than once in your PROFILE RSK.

System action

The extraneous RUNSERV commands are ignored.

System programmer response

Reorganize your `PROFILE RSK`.

BKW0702E   **RUNSERV failed**

Explanation

The server kernel was unable to start.

AUTH Service Messages

BKW0800E   **The class specified already exists**

Explanation

You tried to create an object class but the object class already exists.

System action

None.

System programmer response

Choose a different name for your new object class.

BKW0801E   **Unable to read the authorization files**

Explanation

The server kernel could not read the authorization database.

System action

The server kernel has disabled all calls to the authorization API.

System programmer response

Perhaps an SFS failure or DASD failure has occurred. Contact your system programmer.

BKW0802E   **Unable to write to the authorization files**

Explanation

The server kernel could not write the authorization database.

System action

The server did not start. Other error messages were issued to explain the reason. `PROFILE RSK` will see a nonzero return code from RUNSERV.

System programmer response

Investigate the reason for the failure and take corrective action.

System action

The server kernel has disabled all calls to the authorization API.

System programmer response

Perhaps an SFS failure or DASD failure has occurred. Contact your system programmer. When access to the files is repaired, issue AUTH RELOAD.

BKW0803E   **Too many operations or options specified**

Explanation

You have exceeded the limit on options or operations for this particular command.

System action

The command was not processed.

System programmer response

The most likely cause is that you exceeded the limit of 32 operations per object class. Reduce the number of operations and try again.

BKW0804E   **The length of the object name is out of range**

Explanation

The object name you specified is too long.

System action

The command was not processed.

System programmer response

The object name must be 256 characters or less. Reduce its length and try again.

BKW0805E   **The class specified does not exist**

Explanation

Your command refers to an object class which does not exist.

System action

The command was not processed.

System programmer response

Change the class name. You might also have inadvertently loaded the wrong authorization set. Use SERVER CONFIG to examine the names of the authorization files.

BKW0806E   **The object specified already exists**

Explanation

You tried to create an object but the object already exists.

System action

The command was not processed.

System programmer response

Choose a different name for your object. You might also have inadvertently loaded the wrong authorization set. Use SERVER CONFIG to examine the names of the authorization files.

BKW0807E   **At least one of the options specified is unrecognized**

Explanation

You supplied a command containing options that are unrecognized.

System action

The command was not processed.

System programmer response

Check the syntax diagram for the command you entered, make any necessary corrections, and try again.

BKW0808E   **The object specified does not exist**

Explanation

The object you attempted to manipulate does not exist.

System action

The command was not processed.

System programmer response

Check the command to be sure you are referring to the correct object name. You might also have inadvertently loaded the wrong authorization set. Use SERVER CONFIG to examine the names of the authorization files.

BKW0809E   **The length of the userid specifed is out of range**

Explanation

You specified a user ID that is too long.

System action

The command was not processed.

System programmer response

The user ID must be 64 characters or less in length. Change your command and try again.

BKW0810E   **No rules exist for the userid specified**

Explanation

You asked for a display of the rules for a given user and object, but there were no such rules in the authorization database.

System action

None.

System programmer response

None.

BKW0811E   **Unable to open the authorization files**

Explanation

The server kernel was not able to open the authorization data files.

System action

The authorization API is disabled.

System programmer response

Perhaps an SFS failure or DASD failure has occurred. Contact your system programmer. When access to the files is repaired, issue AUTH RELOAD.

BKW0812E	Operation limit for the class specified has been exceeded

Explanation

You attempted to add a new operation to a class, but it would result in exceeding the limit of 32 operations per object class.

System action

The command was not processed.

System programmer response

Depending on your situation, perhaps a new object class would solve your problem.

BKW0813E	No classes exist for the match key specified

Explanation

You asked for a list of the object classes that match your key, but no such object classes exist.

System action

No object classes were displayed.

System programmer response

Try a different match key. You might also have inadvertently loaded the wrong authorization set. Use SERVER CONFIG to examine the names of the authorization files.

BKW0814E	No objects exist for the match key specified

Explanation

You asked for a list of the objects that match your key, but no such objects exist.

System action

No object names were displayed.

System programmer response

Try a different match key. You might also have inadvertently loaded the wrong authorization set. Use SERVER CONFIG to examine the names of the authorization files.

BKW0815E	No userids exist for the object specified

Explanation

You asked for a list of the user IDs for which there exist rules for the specified object, but there are no rules for the specified object.

System action

No user IDs were displayed.

System programmer response

You might have inadvertently loaded the wrong authorization set. Use SERVER CONFIG to examine the names of the authorization files.

BKW0816E	No rules exist for the userid specified

Explanation

You asked for the rule for the specified user ID and object, but there is no such rule.

System action

No rule is displayed.

System programmer response

You might have inadvertently loaded the wrong authorization set. Use SERVER CONFIG to examine the names of the authorization files.

BKW0817E	Open of authorization data failed - server will not start.

Explanation

The server kernel attempted to open the authorization files as part of server startup, but the open failed.

System action

The server will not start and RUNSERV will be given a nonzero return code.

System programmer response

Correct PROFILE RSK and try again.

CP Service Messages

BKW0900I   **RC=&1 from CP.**

Explanation

CP produced the displayed return code when it processed your command.

System action

The command was executed.

System programmer response

Investigate the return code and take appropriate action.

BKW0901E   **CP response was truncated.**

Explanation

The server kernel passed your command to CP, and CP executed the command, but the response was too long for the server kernel to capture.

System action

The command was executed, but some of its response was not displayed.

System programmer response

Use the displayed portion of the response to determine whether correct results were obtained.

BKW0902E   **CP command was too long.**

Explanation

The CP command you attemped to execute was too long.

System action

The command was not executed.

System programmer response

The length limit is 240 characters. Shorten the command and try again.

CMS Service Messages

BKW1000I   **RC=&1 from CMS.**

Explanation

CMS produced the displayed return code when it processed your command.

System action

The command was executed.

System programmer response

Investigate the return code and take appropriate action.

BKW1001E   **RC=&1 RE=&2 acquiring CMS mutex.**

Explanation

The server kernel was not able to acquire the mutex it needs to pass commands to CMS.

System action

The CMS command was not executed.

System programmer response

Contact IBM support.

MSG Line Driver Messages

BKW1100E   **No userid mapping for user &1 at &2 - message ignored**

Explanation

The MSG line driver used `ssUseridMap` to map the message's origin user ID and node into a local user ID, but `ssUseridMap` was not able to perform a mapping because no applicable entry was found in the user ID mapping file.

System action

The MSG line driver ignored the message.

System programmer response

Adjust the user ID mapping file if necessary, or set configuration parameter MSG_NOMAP to ON so as to let the MSG driver accept the message anyway.

SPOOL Line Driver Messages

BKW1200E   **(file &1) DIAG 14 (order) failed - RC=&2 - file held**

Explanation

The SPOOL line driver attempted to use DIAG X'0014' to move the displayed spool file to the front of the reader queue, but it was unable to do so.

System action

The SPOOL driver placed the file in USER HOLD state.

System programmer response

The DIAG X'0014' return code appears in the message text. Investigate the return code and take appropriate action.

BKW1201E   **(file &1) DIAG 14 (select next) failed - RC=&2 - file held**

Explanation

The SPOOL line driver attempted to use DIAG X'0014' to select the next file in the reader queue, but it was unable to do so.

System action

The SPOOL driver placed the file in USER HOLD state.

System programmer response

The DIAG X'0014' return code appears in the message text. Investigate the return code and take appropriate action.

BKW1202E   **(file &1) Unrecognized spool file format - file held**

Explanation

The SPOOL line driver did not recognize the format of the displayed spool file.

System action

The SPOOL driver placed the file in USER HOLD state.

System programmer response

The file is probably not one that the server kernel is prepared to handle. Transfer it out of the server's reader queue, locate the sender, and find out what his intention was.

BKW1203E   **(file &1) DIAG 14 (read SPLINK) failed - RC=&2 - file held**

Explanation

The SPOOL line driver attempted to use DIAG X'0014' to read the next buffer of spool file data, but it was unable to do so.

System action

The SPOOL driver placed the file in USER HOLD state.

System programmer response

The DIAG X'0014' return code appears in the message text. Investigate the return code and take appropriate action.

BKW1204E   **(file &1) No userid mapping for user &1 at &2 - file held**

Explanation

The SPOOL line driver used `ssUseridMap` to map the spool file's origin user ID and node into a local userid,

but `ssUseridMap` was not able to perform a mapping because no applicable entry was found in the user ID mapping file.

System action

The SPOOL driver placed the file in USER HOLD status.

System programmer response

Adjust the user ID mapping file if necessary, or set configuration parameter SPL_NOMAP to ON so as to let the SPOOL driver accept the file anyway.

BKW1205E   **Punch via DIAG A8 failed - RC=&1**

Explanation

The SPOOL driver attempted to punch a response through DIAG X'00A8' but was not able to do so.

System action

The response was not sent.

System programmer response

The return code from DIAG X'00A8' is displayed in the message. Investigate the return code and take appropriate action. The most likely cause is that spool space is full.

BKW1206E   **Could not encode instance data stream**

Explanation

The service in which the response originated used the correct encoding procedure to generate a record-oriented response for its client, but the response contains a record longer than 65,535 bytes.

System action

The response was not sent to the client.

System programmer response

This is a server defect, not an IBM defect. Contact the server author.

BKW1207E   **(file &1) Unrecognized spool file format - file transferred to &2**

Explanation

The SPOOL line driver did not recognize the format of the displayed spool file.

System action

The SPOOL driver transferred the file to the named user ID.

System programmer response

The file is probably not one that the server kernel is prepared to handle. Locate the sender and find out what his intention was.

Enrollment API Messages

BKW1300E   **Enrollment set &1, record &2 skipped**

Explanation

The server kernel encountered an unrecognizable record in the enrollment data file as it was loading the file into the data space. It skipped the record.

System action

The record was skipped, but loading of subsequent records continued.

System programmer response

Unload the enrollment set and examine the enrollment file with XEDIT. Repair the record so that it conforms to the format specified in the enrollment file appendix of this book.

MONITOR Service Messages

BKW1400E   **Matching monitor row not found.**

Explanation

You asked the MONITOR service to display the monitor rows matching the tokens you specified, but no such monitor row exists.

System action

None.

System programmer response

None.

BKW1401E   **DIAG DC RC &1 starting APPLDATA monitoring**

Explanation

The server kernel tried to establish a CP APPLDATA buffer but was not able to do so. DIAG X'00DC' returned the displayed return code.

System action

CP will not collect the server virtual machine's APPLDATA. The server virtual machine will run normally.

System programmer response

If you want CP to collect the server virtual machine's APPLDATA, make sure `OPTION  APPLMON` is enabled in the server virtual machine's CP directory entry.

BKW1402E   **Monitor adjusted to &1 kernel rows and &2 bytes user data**

Explanation

The server kernel tried to set up the monitor buffer according to the configuration you specified, but the resulting buffer ended up exceeding CP's limit on the size of a monitor buffer.

System action

The server kernel resized the monitor buffer and displayed the actual buffer configuration in the message text.

System programmer response

None.

BKW1403I   **No free monitor row for &1**

Explanation

Some operator command or API call caused the server kernel to attempt to allocate another monitor row, but the monitor buffer cannot accommodate any more monitor rows.

System action

The server kernel will not accumulate monitor data for the displayed component, but operation of the server continues.

System programmer response

If possible, increase the number of monitor rows.

CACHE Service Messages

BKW1500E   **No file caches found.**

Explanation

You asked the CACHE service to display a list of the file caches it is managing, but it is managing no file caches.

System action

None.

System programmer response

None.

IUCV Line Driver Messages

BKW1600I   **Instance STOP requested.**

Explanation

The IUCV line driver has asked an instance thread to STOP.

System action

The server kernel will sever the path to the client after the instance thread acknowledges the STOP request.

System programmer response

None.

BKW1601E   **A-block &1 rsn &2 QueueReceiveBlock RC=&3 RE=&4 failed**

Explanation

The thread controlling an IUCV subtask detected the displayed return and reason code when it attempted to receive a message from its CMS queue.

System action

The server kernel terminates the subtask.

System programmer response

Research the displayed return and reason code and take appropriate corrective action.

BKW1602I   **A-block &1 Client &2 started, C-block &3**

Explanation

The IUCV line driver has accepted a connection from a client.

System action

The server kernel handles the client.

System programmer response

None.

BKW1603I   **A-block &1 Client &2 done, lifetime &3 msec**

Explanation

The IUCV line driver was handling a client, and the connection to the client has ended. The connection lasted for the displayed number of milliseconds.

System action

The server kernel cleans up and prepares to handle another client.

System programmer response

None.

BKW1604I   **A-block &1 Client &2 done, inbytes &3, inrate &4 KB/s**

Explanation

The IUCV line driver was handling a client, and the connection to the client has ended. The server experienced the displayed input byte count and input data rate.

System action

Nothing.

System programmer response

None.

BKW1605I   **A-block &1 Client &2 done, outbytes &3, outrate &4 KB/s**

Explanation

The IUCV line driver was handling a client, and the connection to the client has ended. The server experienced the displayed output byte count and output data rate.

System action

Nothing.

System programmer response

None.

BKW1606E   **Wait expired for STOP.**

Explanation

You issued a STOP command to the IUCV line driver, and it attempted to stop the subtask gracefully, but the wait expired before the graceful stop completed.

System action

The IUCV line driver continues to wait for the subtask to stop normally.

System programmer response

To finish the stop at a later time, reissue the STOP command.

BKW1607E   **Client count must be greater than zero.**

Explanation

You issued an IUCV START command but the client count was zero.

System action

Nothing, except to issue this message.

System programmer response

Specify a nonzero client count.

BKW1608E   **Unable to HNDIUCV SET.**

Explanation

You issued an IUCV START command but the IUCV line driver was not able to identify the needed HNDIUCV exit.

System action

The subtask was not started.

System programmer response

You probably inadvertently duplicated an exit name. Try another exit name.

BKW1609E   **Unable to create controlling thread.**

Explanation

You issued an IUCV START command but the IUCV line driver was not able to create a CMS thread to control the subtask.

System action

The subtask was not started.

System programmer response

Contact IBM support.

BKW1610E   **A-block &1 C-block &2 ThreadCreate RC=&3 RE=&4 failed (major)**

Explanation

A client connected to the server through the IUCV line driver but the line driver was not able to create a thread to run on behalf of the client.

System action

The subtask is terminated.

System programmer response

Contact IBM support.

BKW1611E   **A-block &1 C-block &2 ThreadCreate RC=&3 RE=&4 failed (minor)**

Explanation

A client connected to the server through the IUCV line driver but the line driver was not able to create a thread to run on behalf of the client.

System action

The client will be served by another thread, as soon as said other thread becomes available.

System programmer response

None.

BKW1612E   **A-block &1 C-block &2 IUCV SEND IPRCODE &3 - severing**

Explanation

The IUCV line driver encountered the displayed IPRCODE when it attempted to send data to a client using IUCV SEND.

System action

The IUCV line driver severs the connection to the client.

System programmer response

Research the IPRCODE and take appropriate corrective action.

BKW1613E   **No userid mapping for userid &1 - severing**

Explanation

The IUCV line driver was unable to map the client's VM user ID.

System action

Because `NOMAP_IUCV` was set OFF, the server kernel severed the connection.

APPC Line Driver Messages

BKW1700E   **(Resource &1) CMSIUCV CONNECT to *IDENT RC=&2**

Explanation

The APPC line driver encountered the displayed return code when attempting to connect to *IDENT to begin managing the displayed APPC/VM resource.

System action

The APPC START command failed.

System programmer response

Using the CP QUERY RESOURCE command to determine whether some other virtual machine is already managing the resource. If so, resolve the conflict. If not, contact your system programmer.

BKW1701E   **(Resource &1) Unexpected IUCV interrupt, IPTYPE=&2**

Explanation

The server kernel encountered the displayed external interrupt type while managing an APPC/VM conversation and was not expecting such an external interrupt.

System action

The conversation was severed.

System programmer response

Contact IBM support.

BKW1702E   **Unable to identify APPC/VM resource.**

Explanation

The server kernel was not able to begin managing an APPC/VM resource.

System action

The APPC START command failed.

System programmer response

Update the user ID mapping file or set `NOMAP_IUCV` to ON.

System programmer response

This message is issued in conjunction with some other message that tells what kind of failure was encountered. Refer to the other message for more information.

BKW1703E   **No userid mapping for LU &1, userid &2 - severing**

Explanation

The attempt to pass the displayed user ID and LU name through the user ID mapping file failed, and NOMAP_APPC was OFF.

System action

The conversation was severed.

System programmer response

Update the user ID mapping file or set NOMAP_APPC ON.

BKW1704I   **A-block &1 Client &2 &3 started, C-block &4**

Explanation

The APPC line driver has accepted a connection from a client.

System action

The server kernel handles the client.

System programmer response

None.

BKW1705I   **A-block &1 Client &2 &3 done, lifetime &4 msec**

Explanation

The APPC line driver was handling a client, and the connection to the client has ended. The connection lasted for the displayed number of milliseconds.

System action

The server kernel cleans up and prepares to handle another client.

System programmer response

None.

BKW1706I   **A-block &1 Client &2 &3 done, inbytes &4, inrate &5 KB/s**

Explanation

The APPC line driver was handling a client, and the connection to the client has ended. The server experienced the displayed input byte count and input data rate.

System action

Nothing.

Worker API Messages

BKW1800E   **Worker machine is already in the specified class.**

Explanation

You attempted to add a worker machine to a given worker class, but the worker already belongs to that class.

System action

Nothing.

System programmer response

Probably nothing. If you are attempting to increase the worker's capacity, delete it first and then add it again.

BKW1801E   **Worker machine not found.**

Explanation

You attempted to delete a worker machine but it does not seem to belong to any class.

System action

None.

System programmer response

Check the command and try again.

BKW1802E   **Worker class not found.**

System programmer response

None.

BKW1707I   **A-block &1 Client &2 &3 done, outbytes &4, outrate &5 KB/s**

Explanation

The APPC line driver was handling a client, and the connection to the client has ended. The server experienced the displayed output byte count and output data rate.

System action

Nothing.

System programmer response

None.

Explanation

You attempted to operate on a specific worker class, but the class doesn't seem to exist.

System action

None.

System programmer response

Check the command and try again.

BKW1803E   **No worker classes defined.**

Explanation

You attempted to display information about the worker machine configuration, but there are no worker classes defined.

System action

None.

System programmer response

Confirm that you did in fact issue the WORKER ADD commands necessary to create your worker pools.

BKW1804E   **No worker connections found.**

Explanation

You attempted to use the STATUS command to see information about active connections to worker machines, but there currently are no such connections.

System action

None.

System programmer response

None.

BKW1805E   **No worker machines found.**

Explanation

You attempted to display information about a set of worker machines, but there are no such worker machines defined.

System action

None.

Trie Messages

BKW1900E   **No tries found.**

Explanation

You asked to see a list of existing tries, but no tries exist.

System action

Nothing.

System programmer response

If you were expecting tries, check to see whether their creation was attempted, and if so, whether it succeeded or failed.

System programmer response

None.

BKW1806E   **P-block &1 IUCV SEND IPRCODE &3 - severing**

Explanation

The server kernel encountered the displayed IPRCODE when attempting to use IUCV to send information to a worker machine.

System action

The server kernel severs the IUCV connection and informs the instance accordingly.

System programmer response

Investigate the IPRCODE and determine whether a configuration change is appropriate.

Appendix I. Language Bindings

This appendix documents the language bindings used for PL/X and assembler.

Assembler Language Bindings

All of these binding macros invoke the VMASMMAX macro to ease the allocation of storage for parameter lists. For more information on VMASMMAX, see *z/VM: CMS Application Multitasking*.

Anchor Bindings (SSASMANC MACRO)

```
         MACRO                                                          SSA00010
         SSASMANC  &WEAK=                                               SSA00020
         AGO   .@ASMAN1                                                 SSA00030
.*  Branch around prolog so it is not included in listings        *    SSA00040
.*****************************************************************      SSA00050
.*                                                                *    SSA00060
.*  NAME      -  Reusable Server Kernel   anchor bindings         *    SSA00070
.*                                                                *    SSA00080
.*  FUNCTION  -  Defines the anchor constants and dsects          *    SSA00090
.*                                                                *    SSA00100
.*  COPYRIGHT -                                          @VR2OZOZ SSA00110
.*                                                       @VR2OZOZ SSA00120
.*      5684-112 (C) COPYRIGHT IBM CORP.1991, 1992       @VR2OZOZ SSA00130
.*      LICENSED MATERIALS - PROPERTY OF IBM             @VR2OZOZ SSA00140
.*      SEE COPYRIGHT INSTRUCTIONS, G120-2083            @VR2OZOZ SSA00150
.*      ALL RIGHTS RESERVED                              @VR2OZOZ SSA00160
.*                                                                *    SSA00170
.*  STATUS - VM/ESA Version 2 Release 4                   @VR2OZOZ SSA00180
.*                                                                *    SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4        *    SSA00200
.*****************************************************************      SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4      @VR74PVM SSA00220
.*****************************************************************      SSA00230
.@ASMAN1 ANOP                                                          SSA00240
         PUSH  PRINT                                                   SSA00250
         AIF   ('&SYSPARM' NE 'SUP').ASMAN2                            SSA00260
         PRINT OFF,NOGEN                                               SSA00270
.ASMAN2  ANOP                                                         SSA00280
         LCLC  &$XXTRN                                                 SSA00290
&$XXTRN  SETC  'EXTRN'                                                 SSA00300
         AIF   ('&WEAK' NE 'YES').ASMAN3                               SSA00310
&$XXTRN  SETC  'WXTRN'                                                 SSA00320
.ASMAN3  ANOP                                                         SSA00330
*----------------------------------------------------------------*    SSA00340
*     Return and reason codes for anchor functions               *    SSA00350
*----------------------------------------------------------------*    SSA00360
         SPACE 1                                                       SSA00370
*                                                                      SSA00380
* return codes                                                         SSA00390
SS_ANC_RC_SUCCESS              EQU    0                                SSA00400
SS_ANC_RC_WARNING              EQU    4                                SSA00410
SS_ANC_RC_ERROR               EQU    8                                SSA00420
SS_ANC_RC_ABEND               EQU    12                               SSA00430
*                                                                      SSA00440
* reason codes                                                         SSA00450
SS_ANC_RE_SUCCESS              EQU    0                                SSA00460
*----------------------------------------------------------------*    SSA00470
*     Constants for anchor functions                             *    SSA00480
*----------------------------------------------------------------*    SSA00490
         SPACE 1                                                       SSA00500
*----------------------------------------------------------------*    SSA00510
*     Definitions for anchor functions                           *    SSA00520
*----------------------------------------------------------------*    SSA00530
         SPACE 1                                                       SSA00540
*----------------------------------------------------------------*    SSA00550
*     Declaration for ssAnchorSet                                *    SSA00560
*----------------------------------------------------------------*    SSA00570
         SPACE 1                                                       SSA00580
         &$XXTRN  BKWAST                                               SSA00590
SSANCHORSET                    EQU       BKWAST                        SSA00600
         SPACE 1                                                       SSA00610
```

```
BKWAST_PLIST                        DSECT                           SSA00620
BKWAST_PLIST_RC                     DS       A  * return code       SSA00630
BKWAST_PLIST_RE                     DS       A  * reason code       SSA00640
BKWAST_PLIST_AV                     DS       A  * anchor value      SSA00650
BKWAST_PLIST_LENGTH                 EQU      *-BKWAST_PLIST          SSA00660
                                    VMASMMAX                        SSA00670
        SPACE 1                                                     SSA00680
*-------------------------------------------------------------------*   SSA00690
*     Declaration for ssAnchorGet                               *   SSA00700
*-------------------------------------------------------------------*   SSA00710
        SPACE 1                                                     SSA00720
        &$XXTRN  BKWAGT                                             SSA00730
SSANCHORGET                         EQU      BKWAGT                 SSA00740
        SPACE 1                                                     SSA00750
BKWAGT_PLIST                        DSECT                           SSA00760
BKWAGT_PLIST_RC                     DS       A  * return code       SSA00770
BKWAGT_PLIST_RE                     DS       A  * reason code       SSA00780
BKWAGT_PLIST_AV                     DS       A  * anchor value      SSA00790
BKWAGT_PLIST_MB                     DS       A  * monitor buffer    SSA00800
BKWAGT_PLIST_MBL                    DS       A  * monitor buffer length  SSA00810
BKWAGT_PLIST_LENGTH                 EQU      *-BKWAGT_PLIST          SSA00820
                                    VMASMMAX                        SSA00830
*-------------------------------------------------------------------*   SSA00840
*     End of declarations                                       *   SSA00850
*-------------------------------------------------------------------*   SSA00860
        EJECT                                                       SSA00870
        POP   PRINT                                                 SSA00880
        MEND                                                        SSA00890
```

Authorization Bindings (SSASMAUT MACRO)

```
        MACRO                                                       SSA00010
        SSASMAUT &WEAK=                                     SSA00020
        AGO   .@ASMAU1                                              SSA00030
.*  Branch around prolog so it is not included in listings     *   SSA00040
.*******************************************************************   SSA00050
.*                                                             *   SSA00060
.*  NAME     -  Reusable Server Kernel   authorization bindings *   SSA00070
.*                                                             *   SSA00080
.*  FUNCTION -  Defines the authorization constants and dsects  *   SSA00090
.*                                                             *   SSA00100
.*  COPYRIGHT -                                      @VR2OZOZ SSA00110
.*                                                   @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992    @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM          @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083         @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                           @VR2OZOZ SSA00160
.*                                                             *   SSA00170
.*  STATUS - VM/ESA Version 2 Release 4              @VR2OZOZ SSA00180
.*                                                             *   SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4      *   SSA00200
.*******************************************************************   SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4    @VR74PVM SSA00220
.*******************************************************************   SSA00230
.@ASMAU1 ANOP                                                       SSA00240
        PUSH  PRINT                                                 SSA00250
        AIF   ('&SYSPARM' NE 'SUP').ASMAU2                  SSA00260
        PRINT OFF,NOGEN                                             SSA00270
.ASMAU2  ANOP                                                       SSA00280
        LCLC  &$XXTRN                                               SSA00290
&$XXTRN  SETC  'EXTRN'                                              SSA00300
        AIF   ('&WEAK' NE 'YES').ASMAU3                     SSA00310
&$XXTRN  SETC  'WXTRN'                                              SSA00320
.ASMAU3  ANOP                                                       SSA00330
*-------------------------------------------------------------------*   SSA00340
*     Return and reason codes for authorization functions       *   SSA00350
*-------------------------------------------------------------------*   SSA00360
        SPACE 1                                                     SSA00370
*                                                                   SSA00380
* return codes                                                      SSA00390
SS_AUT_RC_SUCCESS                   EQU      0                      SSA00400
SS_AUT_RC_WARNING                   EQU      4                      SSA00410
SS_AUT_RC_ERROR                     EQU      8                      SSA00420
SS_AUT_RC_ABEND                     EQU      12                     SSA00430
*                                                                   SSA00440
* reason codes                                                      SSA00450
SS_AUT_RE_SUCCESS                   EQU      0                      SSA00460
SS_AUT_RE_BAD_COUNT                 EQU      301                    SSA00470
SS_AUT_RE_BAD_USER_LENGTH           EQU      302                    SSA00480
```

```
SS_AUT_RE_BAD_OBJ_LENGTH         EQU      303                            SSA00490
SS_AUT_RE_BAD_OPTION             EQU      304                            SSA00500
SS_AUT_RE_BAD_QUAL               EQU      305                            SSA00510
SS_AUT_RE_BAD_USE                EQU      306                            SSA00520
SS_AUT_RE_EXISTS                 EQU      307                            SSA00530
SS_AUT_RE_NO_CLASS               EQU      308                            SSA00540
SS_AUT_RE_NO_OBJECT              EQU      309                            SSA00550
SS_AUT_RE_MAQ_FAIL               EQU      310                            SSA00560
SS_AUT_RE_CVW_FAIL               EQU      311                            SSA00570
SS_AUT_RE_CVS_FAIL               EQU      312                            SSA00580
SS_AUT_RE_MR_FAIL                EQU      313                            SSA00590
SS_AUT_RE_TOO_MANY               EQU      314                            SSA00600
SS_AUT_RE_OUT_OF_STORAGE         EQU      315                            SSA00610
SS_AUT_RE_NO_USER                EQU      316                            SSA00620
SS_AUT_RE_PREV_IO_ERROR          EQU      317                            SSA00630
SS_AUT_RE_PREV_SYNC_ERROR        EQU      318                            SSA00640
SS_AUT_RE_READ_FAIL              EQU      319                            SSA00650
SS_AUT_RE_WRITE_FAIL             EQU      320                            SSA00660
SS_AUT_RE_TRUNC                  EQU      321                            SSA00670
SS_AUT_RE_GWU_FAIL               EQU      322                            SSA00680
SS_AUT_RE_OPEN_FAIL              EQU      323                            SSA00690
SS_AUT_RE_BAD_CACHE              EQU      324                            SSA00700
SS_AUT_RE_BAD_FREE               EQU      325                            SSA00710
SS_AUT_RE_BAD_OP                 EQU      326                            SSA00720
*                                                                       SSA00730
*---------------------------------------------------------------*       SSA00740
*     Constants for authorization functions                     *       SSA00750
*---------------------------------------------------------------*       SSA00760
        SPACE 1                                                         SSA00770
*-------------------------------------------*                           SSA00780
* Return values from ssAuthTestOperations  *                           SSA00790
* and ssAuthPermitUser                      *                           SSA00800
*-------------------------------------------*                           SSA00810
SS_AUT_OP_PERMITTED              EQU      0                              SSA00820
SS_AUT_OP_NOT_PERMITTED          EQU      1                              SSA00830
SS_AUT_OP_NOT_DEFINED            EQU      2                              SSA00840
SS_AUT_OP_NO_CHANGE              EQU      3                              SSA00850
*                                                                       SSA00860
*-------------------------------------------*                           SSA00870
* Qualifiers for ssAuthPermitUser          *                           SSA00880
*-------------------------------------------*                           SSA00890
SS_AUT_ADD_OPERATION             EQU      0                              SSA00900
SS_AUT_REMOVE_OPERATION          EQU      1                              SSA00910
*                                                                       SSA00920
*-------------------------------------------*                           SSA00930
* Use arrays in ssAuthPermitUser           *                           SSA00940
*-------------------------------------------*                           SSA00950
SS_AUT_USE_ARRAYS                EQU      0                              SSA00960
SS_AUT_DELETE_ALL                EQU      1                              SSA00970
SS_AUT_ADD_ALL                   EQU      2                              SSA00980
*                                                                       SSA00990
*-------------------------------------------*                           SSA01000
* Qualifiers for ssAuthDeleteObject        *                           SSA01010
*-------------------------------------------*                           SSA01020
SS_AUT_RULES_ONLY                EQU      0                              SSA01030
SS_AUT_RULES_AND_OBJECT          EQU      1                              SSA01040
*                                                                       SSA01050
*-------------------------------------------*                           SSA01060
* Qualifiers for ssAuthDeleteUser          *                           SSA01070
*-------------------------------------------*                           SSA01080
SS_AUT_SPECIFIC_CLASS            EQU      0                              SSA01090
SS_AUT_ALL_CLASSES               EQU      1                              SSA01100
*                                                                       SSA01110
*-------------------------------------------*                           SSA01120
* Qualifiers for ssAuthDeleteClass         *                           SSA01130
*-------------------------------------------*                           SSA01140
SS_AUT_OBJECTS_ONLY              EQU      0                              SSA01150
SS_AUT_OBJECTS_AND_CLASS         EQU      1                              SSA01160
*                                                                       SSA01170
*---------------------------------------------------------------*       SSA01180
*     Definitions for authorization functions                   *       SSA01190
*---------------------------------------------------------------*       SSA01200
        SPACE 1                                                         SSA01210
*---------------------------------------------------------------*       SSA01220
*     Operations on classes                                     *       SSA01230
*---------------------------------------------------------------*       SSA01240
*                                                                       SSA01250
*     create class                                                      SSA01260
*                                                                       SSA01270
        SPACE 1                                                         SSA01280
        &$XXTRN  BKWUCC                                                 SSA01290
SSAUTHCREATECLASS                EQU      BKWUCC                         SSA01300
```

```
        SPACE 1                                                         SSA01310
BKWUCC_PLIST                    DSECT                                    SSA01320
BKWUCC_PLIST_RC                 DS      A   * return code                SSA01330
BKWUCC_PLIST_RE                 DS      A   * reason code                SSA01340
BKWUCC_PLIST_CID                DS      A   * class identifier           SSA01350
BKWUCC_PLIST_OC                 DS      A   * operation count            SSA01360
BKWUCC_PLIST_OA                 DS      A   * operation array            SSA01370
BKWUCC_PLIST_LENGTH             EQU       *-BKWUCC_PLIST                 SSA01380
                                VMASMMAX                                 SSA01390
        SPACE 1                                                         SSA01400
*                                                                        SSA01410
*     modify class                                                       SSA01420
*                                                                        SSA01430
        SPACE 1                                                         SSA01440
        &$XXTRN   BKWUMC                                                 SSA01450
SSAUTHMODIFYCLASS               EQU       BKWUMC                         SSA01460
        SPACE 1                                                         SSA01470
BKWUMC_PLIST                    DSECT                                    SSA01480
BKWUMC_PLIST_RC                 DS      A   * return code                SSA01490
BKWUMC_PLIST_RE                 DS      A   * reason code                SSA01500
BKWUMC_PLIST_CID                DS      A   * class identifier           SSA01510
BKWUMC_PLIST_OC                 DS      A   * operation count            SSA01520
BKWUMC_PLIST_OA                 DS      A   * operation array            SSA01530
BKWUMC_PLIST_LENGTH             EQU       *-BKWUMC_PLIST                 SSA01540
                                VMASMMAX                                 SSA01550
        SPACE 1                                                         SSA01560
*                                                                        SSA01570
*     list classes                                                       SSA01580
*                                                                        SSA01590
        SPACE 1                                                         SSA01600
        &$XXTRN   BKWULC                                                 SSA01610
SSAUTHLISTCLASSES               EQU       BKWULC                         SSA01620
        SPACE 1                                                         SSA01630
BKWULC_PLIST                    DSECT                                    SSA01640
BKWULC_PLIST_RC                 DS      A   * return code                SSA01650
BKWULC_PLIST_RE                 DS      A   * reason code                SSA01660
BKWULC_PLIST_MK                 DS      A   * match key                  SSA01670
BKWULC_PLIST_MKL                DS      A   * match key length           SSA01680
BKWULC_PLIST_NE                 DS      A   * number expected            SSA01690
BKWULC_PLIST_OB                 DS      A   * output buffer              SSA01700
BKWULC_PLIST_NR                 DS      A   * number returned            SSA01710
BKWULC_PLIST_LENGTH             EQU       *-BKWULC_PLIST                 SSA01720
                                VMASMMAX                                 SSA01730
        SPACE 1                                                         SSA01740
*                                                                        SSA01750
*     delete class                                                       SSA01760
*                                                                        SSA01770
        SPACE 1                                                         SSA01780
        &$XXTRN   BKWUDC                                                 SSA01790
SSAUTHDELETECLASS               EQU       BKWUDC                         SSA01800
        SPACE 1                                                         SSA01810
BKWUDC_PLIST                    DSECT                                    SSA01820
BKWUDC_PLIST_RC                 DS      A   * return code                SSA01830
BKWUDC_PLIST_RE                 DS      A   * reason code                SSA01840
BKWUDC_PLIST_CID                DS      A   * class identifier           SSA01850
BKWUDC_PLIST_OC                 DS      A   * option count               SSA01860
BKWUDC_PLIST_OA                 DS      A   * option array               SSA01870
BKWUDC_PLIST_LENGTH             EQU       *-BKWUDC_PLIST                 SSA01880
                                VMASMMAX                                 SSA01890
*----------------------------------------------------------------*       SSA01900
*     Operations on objects                                      *       SSA01910
*----------------------------------------------------------------*       SSA01920
*                                                                        SSA01930
*     create object                                                      SSA01940
*                                                                        SSA01950
        SPACE 1                                                         SSA01960
        &$XXTRN   BKWUCO                                                 SSA01970
SSAUTHCREATEOBJECT              EQU       BKWUCO                         SSA01980
        SPACE 1                                                         SSA01990
BKWUCO_PLIST                    DSECT                                    SSA02000
BKWUCO_PLIST_RC                 DS      A   * return code                SSA02010
BKWUCO_PLIST_RE                 DS      A   * reason code                SSA02020
BKWUCO_PLIST_ON                 DS      A   * object name                SSA02030
BKWUCO_PLIST_ONL                DS      A   * object name length         SSA02040
BKWUCO_PLIST_CID                DS      A   * object class               SSA02050
BKWUCO_PLIST_LENGTH             EQU       *-BKWUCO_PLIST                 SSA02060
                                VMASMMAX                                 SSA02070
        SPACE 1                                                         SSA02080
*                                                                        SSA02090
*     list objects in class                                             SSA02100
*                                                                        SSA02110
        SPACE 1                                                         SSA02120
```

```
            &$XXTRN  BKWULO                                              SSA02130
SSAUTHLISTOBJECTS               EQU       BKWULO                        SSA02140
        SPACE 1                                                         SSA02150
BKWULO_PLIST                    DSECT                                   SSA02160
BKWULO_PLIST_RC                 DS        A    * return code            SSA02170
BKWULO_PLIST_RE                 DS        A    * reason code            SSA02180
BKWULO_PLIST_CID                DS        A    * class identifier       SSA02190
BKWULO_PLIST_MK                 DS        A    * match key              SSA02200
BKWULO_PLIST_MKL                DS        A    * match key length       SSA02210
BKWULO_PLIST_NE                 DS        A    * number expected        SSA02220
BKWULO_PLIST_BP                 DS        A    * buffer pointers        SSA02230
BKWULO_PLIST_BS                 DS        A    * buffer sizes           SSA02240
BKWULO_PLIST_RL                 DS        A    * returned lengths       SSA02250
BKWULO_PLIST_NR                 DS        A    * number returned        SSA02260
BKWULO_PLIST_LENGTH             EQU       *-BKWULO_PLIST                SSA02270
                                VMASMMAX                               SSA02280
        SPACE 1                                                         SSA02290
*                                                                       SSA02300
*    query an object                                                    SSA02310
*                                                                       SSA02320
        SPACE 1                                                         SSA02330
        &$XXTRN  BKWUQO                                                 SSA02340
SSAUTHQUERYOBJECT               EQU       BKWUQO                        SSA02350
        SPACE 1                                                         SSA02360
BKWUQO_PLIST                    DSECT                                   SSA02370
BKWUQO_PLIST_RC                 DS        A    * return code            SSA02380
BKWUQO_PLIST_RE                 DS        A    * reason code            SSA02390
BKWUQO_PLIST_ON                 DS        A    * object name            SSA02400
BKWUQO_PLIST_ONL                DS        A    * object name length     SSA02410
BKWUQO_PLIST_CID                DS        A    * class identifier       SSA02420
BKWUQO_PLIST_UX                 DS        A    * userids expected       SSA02430
BKWUQO_PLIST_UBP                DS        A    * userid buf pointers    SSA02440
BKWUQO_PLIST_UBS                DS        A    * userid buf sizes       SSA02450
BKWUQO_PLIST_UL                 DS        A    * userid lengths         SSA02460
BKWUQO_PLIST_UR                 DS        A    * userids returned       SSA02470
BKWUQO_PLIST_LENGTH             EQU       *-BKWUQO_PLIST                SSA02480
                                VMASMMAX                               SSA02490
        SPACE 1                                                         SSA02500
*                                                                       SSA02510
*    delete an object                                                   SSA02520
*                                                                       SSA02530
        SPACE 1                                                         SSA02540
        &$XXTRN  BKWUDO                                                 SSA02550
SSAUTHDELETEOBJECT              EQU       BKWUDO                        SSA02560
        SPACE 1                                                         SSA02570
BKWUDO_PLIST                    DSECT                                   SSA02580
BKWUDO_PLIST_RC                 DS        A    * return code            SSA02590
BKWUDO_PLIST_RE                 DS        A    * reason code            SSA02600
BKWUDO_PLIST_ON                 DS        A    * object name            SSA02610
BKWUDO_PLIST_ONL                DS        A    * its length             SSA02620
BKWUDO_PLIST_OC                 DS        A    * option count           SSA02630
BKWUDO_PLIST_OA                 DS        A    * option array           SSA02640
BKWUDO_PLIST_LENGTH             EQU       *-BKWUDO_PLIST                SSA02650
                                VMASMMAX                               SSA02660
        SPACE 1                                                         SSA02670
*---------------------------------------------------------------*      SSA02680
*    Operations on users                                        *      SSA02690
*---------------------------------------------------------------*      SSA02700
*                                                                       SSA02710
*    permit user                                                        SSA02720
*                                                                       SSA02730
        SPACE 1                                                         SSA02740
        &$XXTRN  BKWUPU                                                 SSA02750
SSAUTHPERMITUSER                EQU       BKWUPU                        SSA02760
        SPACE 1                                                         SSA02770
BKWUPU_PLIST                    DSECT                                   SSA02780
BKWUPU_PLIST_RC                 DS        A    * return code            SSA02790
BKWUPU_PLIST_RE                 DS        A    * reason code            SSA02800
BKWUPU_PLIST_UN                 DS        A    * user name              SSA02810
BKWUPU_PLIST_UNL                DS        A    * its length             SSA02820
BKWUPU_PLIST_ON                 DS        A    * object name            SSA02830
BKWUPU_PLIST_ONL                DS        A    * its length             SSA02840
BKWUPU_PLIST_UA                 DS        A    * use arrays?            SSA02850
BKWUPU_PLIST_OC                 DS        A    * operation count        SSA02860
BKWUPU_PLIST_OA                 DS        A    * operation array        SSA02870
BKWUPU_PLIST_OQ                 DS        A    * operation qualifiers   SSA02880
BKWUPU_PLIST_OR                 DS        A    * operation results      SSA02890
BKWUPU_PLIST_LENGTH             EQU       *-BKWUPU_PLIST                SSA02900
                                VMASMMAX                               SSA02910
        SPACE 1                                                         SSA02920
*                                                                       SSA02930
*    query specific rule                                                SSA02940
```

```
*                                                                      SSA02950
        &$XXTRN   BKWUQR                                                SSA02960
SSAUTHQUERYRULE              EQU       BKWUQR                           SSA02970
        SPACE 1                                                         SSA02980
BKWUQR_PLIST                 DSECT                                      SSA02990
BKWUQR_PLIST_RC              DS        A   * return code                SSA03000
BKWUQR_PLIST_RE              DS        A   * reason code                SSA03010
BKWUQR_PLIST_UN              DS        A   * user name                  SSA03020
BKWUQR_PLIST_UNL             DS        A   * its length                 SSA03030
BKWUQR_PLIST_ON              DS        A   * object name                SSA03040
BKWUQR_PLIST_ONL             DS        A   * its length                 SSA03050
BKWUQR_PLIST_OE              DS        A   * ops expected               SSA03060
BKWUQR_PLIST_OA              DS        A   * operation array            SSA03070
BKWUQR_PLIST_OR              DS        A   * ops returned               SSA03080
BKWUQR_PLIST_LENGTH          EQU       *-BKWUQR_PLIST                   SSA03090
                             VMASMMAX                                   SSA03100
        SPACE 1                                                         SSA03110
*                                                                      SSA03120
*     test operations                                                   SSA03130
*                                                                      SSA03140
        SPACE 1                                                         SSA03150
        &$XXTRN   BKWUTO                                                SSA03160
SSAUTHTESTOPERATIONS         EQU       BKWUTO                           SSA03170
        SPACE 1                                                         SSA03180
BKWUTO_PLIST                 DSECT                                      SSA03190
BKWUTO_PLIST_RC              DS        A   * return code                SSA03200
BKWUTO_PLIST_RE              DS        A   * reason code                SSA03210
BKWUTO_PLIST_UN              DS        A   * user name                  SSA03220
BKWUTO_PLIST_UNL             DS        A   * its length                 SSA03230
BKWUTO_PLIST_ON              DS        A   * object name                SSA03240
BKWUTO_PLIST_ONL             DS        A   * its length                 SSA03250
BKWUTO_PLIST_OC              DS        A   * operation count            SSA03260
BKWUTO_PLIST_OA              DS        A   * operation array            SSA03270
BKWUTO_PLIST_TR              DS        A   * test results               SSA03280
BKWUTO_PLIST_LENGTH          EQU       *-BKWUTO_PLIST                   SSA03290
                             VMASMMAX                                   SSA03300
        SPACE 1                                                         SSA03310
*                                                                      SSA03320
*     delete user                                                       SSA03330
*                                                                      SSA03340
        SPACE 1                                                         SSA03350
        &$XXTRN   BKWUDU                                                SSA03360
SSAUTHDELETEUSER             EQU       BKWUDU                           SSA03370
        SPACE 1                                                         SSA03380
BKWUDU_PLIST                 DSECT                                      SSA03390
BKWUDU_PLIST_RC              DS        A   * return code                SSA03400
BKWUDU_PLIST_RE              DS        A   * reason code                SSA03410
BKWUDU_PLIST_UN              DS        A   * user name                  SSA03420
BKWUDU_PLIST_UNL             DS        A   * its length                 SSA03430
BKWUDU_PLIST_CID             DS        A   * class identifier           SSA03440
BKWUDU_PLIST_OC              DS        A   * option count               SSA03450
BKWUDU_PLIST_OA              DS        A   * option array               SSA03460
BKWUDU_PLIST_LENGTH          EQU       *-BKWUDU_PLIST                   SSA03470
                             VMASMMAX                                   SSA03480
        SPACE 1                                                         SSA03490
*----------------------------------------------------------------*      SSA03500
*     Utility functions                                          *      SSA03510
*----------------------------------------------------------------*      SSA03520
*                                                                      SSA03530
*   try to reset access to data files                                   SSA03540
*                                                                      SSA03550
        SPACE 1                                                         SSA03560
        &$XXTRN   BKWURL                                                SSA03570
SSAUTHRELOAD                 EQU       BKWURL                           SSA03580
        SPACE 1                                                         SSA03590
BKWURL_PLIST                 DSECT                                      SSA03600
BKWURL_PLIST_RC              DS        A   * return code                SSA03610
BKWURL_PLIST_RE              DS        A   * reason code                SSA03620
BKWURL_PLIST_LENGTH          EQU       *-BKWURL_PLIST                   SSA03630
                             VMASMMAX                                   SSA03640
        SPACE 1                                                         SSA03650
*----------------------------------------------------------------*      SSA03660
*     End of declarations                                        *      SSA03670
*----------------------------------------------------------------*      SSA03680
        EJECT                                                           SSA03690
        POP   PRINT                                                     SSA03700
        MEND                                                            SSA03710
```

Cache Bindings (SSASMCAC MACRO)

```
         MACRO                                                        SSA00010
         SSASMCAC &WEAK=                                      SSA00020
         AGO   .@ASMOB1                                               SSA00030
.*  Branch around prolog so it is not included in listings        *  SSA00040
.*****************************************************************    SSA00050
.*                                                                *  SSA00060
.*  NAME     -  Reusable Server Kernel cache bindings             *  SSA00070
.*                                                                *  SSA00080
.*  FUNCTION  -  Defines the file cache constants and dsects      *  SSA00090
.*                                                                *  SSA00100
.*  COPYRIGHT -                                         @VR2OZOZ SSA00110
.*                                                      @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992      @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM            @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083           @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                             @VR2OZOZ SSA00160
.*                                                                *  SSA00170
.*  STATUS - Version 2 Release 4                        @VR2OZOZ SSA00180
.*                                                                *  SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4       *  SSA00200
.*****************************************************************    SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4     @VR74PVM SSA00220
.*****************************************************************    SSA00230
.@ASMOB1 ANOP                                                        SSA00240
         PUSH  PRINT                                                 SSA00250
         AIF   ('&SYSPARM' NE 'SUP').ASMOB2                  SSA00260
         PRINT OFF,NOGEN                                             SSA00270
.ASMOB2  ANOP                                                        SSA00280
         LCLC  &$XXTRN                                               SSA00290
&$XXTRN  SETC  'EXTRN'                                               SSA00300
         AIF   ('&WEAK' NE 'YES').ASMOB3                     SSA00310
&$XXTRN  SETC  'WXTRN'                                               SSA00320
.ASMOB3  ANOP                                                        SSA00330
*----------------------------------------------------------------*   SSA00340
*     Return and reason codes for file functions               *     SSA00350
*----------------------------------------------------------------*   SSA00360
         SPACE 1                                                     SSA00370
* return codes                                                       SSA00380
SS_CAC_RC_SUCCESS                 EQU    0                           SSA00390
SS_CAC_RC_WARNING                 EQU    4                           SSA00400
SS_CAC_RC_ERROR                   EQU    8                           SSA00410
SS_CAC_RC_ABEND                   EQU    12                          SSA00420
*                                                                    SSA00430
* reason codes                                                       SSA00440
SS_CAC_RE_SUCCESS                 EQU    0                           SSA00450
SS_CAC_RE_OUT_OF_STORAGE          EQU    1501                        SSA00460
SS_CAC_RE_TABLE_REPLACED          EQU    1502                        SSA00470
SS_CAC_RE_CACHE_NOT_FOUND         EQU    1503                        SSA00480
SS_CAC_RE_DSCR_FAIL               EQU    1504                        SSA00490
SS_CAC_RE_CACHE_EXISTS            EQU    1505                        SSA00500
SS_CAC_RE_BAD_SIZE                EQU    1506                        SSA00510
SS_CAC_RE_BAD_TOKEN               EQU    1511                        SSA00520
SS_CAC_RE_BAD_LENGTH              EQU    1512                        SSA00530
SS_CAC_RE_BAD_COUNT               EQU    1513                        SSA00540
SS_CAC_RE_BAD_ESMDL               EQU    1514                        SSA00550
SS_CAC_RE_BAD_FNAME               EQU    1515                        SSA00560
SS_CAC_RE_BAD_FVAL                EQU    1516                        SSA00570
SS_CAC_RE_EXIST_FAIL              EQU    1517                        SSA00580
SS_CAC_RE_FILE_NOT_FOUND          EQU    1518                        SSA00590
SS_CAC_RE_DELETE_IN_PROGRESS      EQU    1519                        SSA00600
SS_CAC_RE_BAD_OFFSET              EQU    1520                        SSA00610
SS_CAC_RE_BAD_TABLE_ID            EQU    1521                        SSA00620
SS_CAC_RE_TABLE_NOT_FOUND         EQU    1522                        SSA00630
SS_CAC_RE_OPEN_FAIL               EQU    1523                        SSA00640
SS_CAC_RE_BAD_RECFM               EQU    1524                        SSA00650
SS_CAC_RE_BAD_LRECL               EQU    1525                        SSA00660
SS_CAC_RE_OUT_OF_STORAGE_DS       EQU    1526                        SSA00670
SS_CAC_RE_READ_FAIL               EQU    1527                        SSA00680
SS_CAC_RE_BAD_DATA_STREAM         EQU    1528                        SSA00690
         SPACE 1                                                     SSA00700
*----------------------------------------------------------------*   SSA00710
*     Constants for file functions                             *     SSA00720
*----------------------------------------------------------------*   SSA00730
         SPACE 1                                                     SSA00740
* open flag names                                                    SSA00750
SS_CAC_OFN_XLATE                  EQU    0                           SSA00760
SS_CAC_OFN_PRESERVE_DOLR          EQU    1                           SSA00770
SS_CAC_OFN_BFS                    EQU    2                           SSA00780
```

```
SS_CAC_OFN_RECMETHOD_FS          EQU     3                           SSA00790
SS_CAC_OFN_RECMETHOD_CACHE       EQU     4                           SSA00800
*                                                                    SSA00810
* open flag values                                                  SSA00820
SS_CAC_OFV_NO                    EQU     0                           SSA00830
SS_CAC_OFV_YES                   EQU     1                           SSA00840
        SPACE 1                                                      SSA00850
*-------------------------------------------------------------*     SSA00860
*     Definitions for file functions                          *     SSA00870
*-------------------------------------------------------------*     SSA00880
        SPACE 1                                                      SSA00890
*                                                                    SSA00900
*     create cache                                                   SSA00910
*                                                                    SSA00920
        SPACE 1                                                      SSA00930
        &$XXTRN   BKWOCC                                             SSA00940
SSCACHECREATE                    EQU     BKWOCC                      SSA00950
        SPACE 1                                                      SSA00960
BKWOCC_PLIST                     DSECT                               SSA00970
BKWOCC_PLIST_RC                  DS      A   * return code           SSA00980
BKWOCC_PLIST_RE                  DS      A   * reason code           SSA00990
BKWOCC_PLIST_CNAME               DS      A   * cache name            SSA01000
BKWOCC_PLIST_PAGES               DS      A   * file name length      SSA01010
BKWOCC_PLIST_ALET                DS      A   * storage group num     SSA01020
BKWOCC_PLIST_LENGTH              EQU     *-BKWOCC_PLIST              SSA01030
                                 VMASMMAX                            SSA01040
        SPACE 1                                                      SSA01050
*                                                                    SSA01060
*     delete cache                                                   SSA01070
*                                                                    SSA01080
        SPACE 1                                                      SSA01090
        &$XXTRN   BKWOCD                                             SSA01100
SSCACHEDELETE                    EQU     BKWOCD                      SSA01110
        SPACE 1                                                      SSA01120
BKWOCD_PLIST                     DSECT                               SSA01130
BKWOCD_PLIST_RC                  DS      A   * return code           SSA01140
BKWOCD_PLIST_RE                  DS      A   * reason code           SSA01150
BKWOCD_PLIST_CNAME               DS      A   * cache name            SSA01160
BKWOCD_PLIST_LENGTH              EQU     *-BKWOCD_PLIST              SSA01170
                                 VMASMMAX                            SSA01180
        SPACE 1                                                      SSA01190
*                                                                    SSA01200
*     query cache utilization                                        SSA01210
*                                                                    SSA01220
        SPACE 1                                                      SSA01230
        &$XXTRN   BKWOCQ                                             SSA01240
SSCACHEQUERY                     EQU     BKWOCQ                      SSA01250
        SPACE 1                                                      SSA01260
BKWOCQ_PLIST                     DSECT                               SSA01270
BKWOCQ_PLIST_RC                  DS      A   * return code           SSA01280
BKWOCQ_PLIST_RE                  DS      A   * reason code           SSA01290
BKWOCQ_PLIST_CNAME               DS      A   * cache name            SSA01300
BKWOCQ_PLIST_FCOUNT              DS      A   * files cached          SSA01310
BKWOCQ_PLIST_CSIZE               DS      A   * cache size            SSA01320
BKWOCQ_PLIST_INUSE               DS      A   * amt in use            SSA01330
BKWOCQ_PLIST_OCOUNT              DS      A   * open count            SSA01340
BKWOCQ_PLIST_HCOUNT              DS      A   * hit count             SSA01350
BKWOCQ_PLIST_LENGTH              EQU     *-BKWOCQ_PLIST              SSA01360
                                 VMASMMAX                            SSA01370
        SPACE 1                                                      SSA01380
*                                                                    SSA01390
*     set translation table                                          SSA01400
*                                                                    SSA01410
        SPACE 1                                                      SSA01420
        &$XXTRN   BKWOTS                                             SSA01430
SSCACHEXLTABSET                  EQU     BKWOTS                      SSA01440
        SPACE 1                                                      SSA01450
BKWOTS_PLIST                     DSECT                               SSA01460
BKWOTS_PLIST_RC                  DS      A   * return code           SSA01470
BKWOTS_PLIST_RE                  DS      A   * reason code           SSA01480
BKWOTS_PLIST_XLTABID             DS      A   * xltab id              SSA01490
BKWOTS_PLIST_XLTAB               DS      A   * xltab                 SSA01500
BKWOTS_PLIST_LENGTH              EQU     *-BKWOTS_PLIST              SSA01510
                                 VMASMMAX                            SSA01520
        SPACE 1                                                      SSA01530
*                                                                    SSA01540
*     open a cached file                                             SSA01550
*                                                                    SSA01560
        SPACE 1                                                      SSA01570
        &$XXTRN   BKWOFO                                             SSA01580
SSCACHEFILEOPEN                  EQU     BKWOFO                      SSA01590
        SPACE 1                                                      SSA01600
```

```
BKWOFO_PLIST                          DSECT                            SSA01610
BKWOFO_PLIST_RC             DS        A   * return code                SSA01620
BKWOFO_PLIST_RE             DS        A   * reason code                SSA01630
BKWOFO_PLIST_CNAME          DS        A   * cache name                 SSA01640
BKWOFO_PLIST_FSPEC          DS        A   * file spec                  SSA01650
BKWOFO_PLIST_FSPECLEN       DS        A   * its length                 SSA01660
BKWOFO_PLIST_ESMD           DS        A   * ESM data                   SSA01670
BKWOFO_PLIST_ESMDLEN        DS        A   * its length                 SSA01680
BKWOFO_PLIST_FCOUNT         DS        A   * flag count                 SSA01690
BKWOFO_PLIST_FNAMES         DS        A   * flag names                 SSA01700
BKWOFO_PLIST_FVALS          DS        A   * flag values                SSA01710
BKWOFO_PLIST_FTOKEN         DS        A   * file token                 SSA01720
BKWOFO_PLIST_ALET           DS        A   * file ALET                  SSA01730
BKWOFO_PLIST_DSADDR         DS        A   * file DS address            SSA01740
BKWOFO_PLIST_DSLEN          DS        A   * file DS length             SSA01750
BKWOFO_PLIST_LASTUPD        DS        A   * last update date           SSA01760
BKWOFO_PLIST_LENGTH         EQU       *-BKWOFO_PLIST                   SSA01770
                            VMASMMAX                                   SSA01780
        SPACE 1                                                        SSA01790
*                                                                      SSA01800
*    read cached file                                                  SSA01810
*                                                                      SSA01820
        SPACE 1                                                        SSA01830
        &$XXTRN  BKWOFR                                                SSA01840
SSCACHEFILEREAD             EQU       BKWOFR                           SSA01850
        SPACE 1                                                        SSA01860
BKWOFR_PLIST                DSECT                                      SSA01870
BKWOFR_PLIST_RC             DS        A   * return code                SSA01880
BKWOFR_PLIST_RE             DS        A   * reason code                SSA01890
BKWOFR_PLIST_CNAME          DS        A   * cache name                 SSA01900
BKWOFR_PLIST_FTOKEN         DS        A   * file token                 SSA01910
BKWOFR_PLIST_OFFSET         DS        A   * byte offset                SSA01920
BKWOFR_PLIST_COUNT          DS        A   * byte count                 SSA01930
BKWOFR_PLIST_BUFFER         DS        A   * out buffer                 SSA01940
BKWOFR_PLIST_RETURNED       DS        A   * bytes returned             SSA01950
BKWOFR_PLIST_LENGTH         EQU       *-BKWOFR_PLIST                   SSA01960
                            VMASMMAX                                   SSA01970
        SPACE 1                                                        SSA01980
*                                                                      SSA01990
*    close cached file                                                 SSA02000
*                                                                      SSA02010
        SPACE 1                                                        SSA02020
        &$XXTRN  BKWOFC                                                SSA02030
SSCACHEFILECLOSE            EQU       BKWOFC                           SSA02040
        SPACE 1                                                        SSA02050
BKWOFC_PLIST                DSECT                                      SSA02060
BKWOFC_PLIST_RC             DS        A   * return code                SSA02070
BKWOFC_PLIST_RE             DS        A   * reason code                SSA02080
BKWOFC_PLIST_CNAME          DS        A   * cache name                 SSA02090
BKWOFC_PLIST_FTOKEN         DS        A   * file token                 SSA02100
BKWOFC_PLIST_LENGTH         EQU       *-BKWOFC_PLIST                   SSA02110
                            VMASMMAX                                   SSA02120
        SPACE 1                                                        SSA02130
*------------------------------------------------------------------*   SSA02140
*    End of definitions                                            *   SSA02150
*------------------------------------------------------------------*   SSA02160
        EJECT                                                          SSA02170
        POP   PRINT                                                    SSA02180
        MEND                                                           SSA02190
```

Client Bindings (SSASMCLI MACRO)

```
        MACRO                                                          SSA00010
        SSASMCLI &WEAK=                                       SSA00020
        AGO   .@ASMSR1                                                 SSA00030
.*  Branch around prolog so it is not included in listings      *     SSA00040
.********************************************************************   SSA00050
.*                                                              *      SSA00060
.*  NAME     -  Reusable Server Kernel   services bindings      *      SSA00070
.*                                                              *      SSA00080
.*  FUNCTION -  LANGUAGE BINDINGS FOR THE CLIENT SERVICES       *      SSA00090
.*                                                              *      SSA00100
.*  COPYRIGHT -                                          @VR2OZOZ SSA00110
.*                                                       @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992        @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM              @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083             @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                               @VR2OZOZ SSA00160
.*                                                              *      SSA00170
```

```
.*  STATUS - Version 2 Release 4                              @VR2OZOZ SSA00180
.*                                                                 *   SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4           *   SSA00200
.****************************************************************   SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4      @VR24PVM SSA00220
.****************************************************************   SSA00230
.@ASMSR1 ANOP                                                         SSA00240
        PUSH  PRINT                                                   SSA00250
        AIF   ('&SYSPARM' NE 'SUP').ASMSR2                   SSA00260
        PRINT OFF,NOGEN                                              SSA00270
.ASMSR2  ANOP                                                        SSA00280
        LCLC  &$XXTRN                                                SSA00290
&$XXTRN  SETC  'EXTRN'                                               SSA00300
        AIF   ('&WEAK' NE 'YES').ASMSR3                     SSA00310
&$XXTRN  SETC  'WXTRN'                                               SSA00320
.ASMSR3  ANOP                                                        SSA00330
*-------------------------------------------------------------*     SSA00340
*     Return and reason codes for services functions          *     SSA00350
*-------------------------------------------------------------*     SSA00360
        SPACE 1                                                     SSA00370
*                                                                   SSA00380
* return codes                                                      SSA00390
SS_CLI_RC_SUCCESS             EQU     0                            SSA00400
SS_CLI_RC_WARNING             EQU     4                            SSA00410
SS_CLI_RC_ERROR               EQU     8                            SSA00420
SS_CLI_RC_ABEND               EQU     12                           SSA00430
*                                                                   SSA00440
* reason codes                                                      SSA00450
SS_CLI_RE_SUCCESS             EQU     0                            SSA00460
SS_CLI_RE_OUT_OF_RANGE        EQU     901                          SSA00470
SS_CLI_RE_OUT_OF_STORAGE      EQU     902                          SSA00480
SS_CLI_RE_BAD_IAM             EQU     903                          SSA00490
SS_CLI_RE_BAD_METHOD          EQU     904                          SSA00500
SS_CLI_RE_SEMC_FAIL           EQU     905                          SSA00510
*                                                                   SSA00520
* Who i am                                                          SSA00530
        SPACE 1                                                     SSA00540
SS_CLI_IAM_INSTANCE           EQU     0                            SSA00550
SS_CLI_IAM_LINEDRIVER         EQU     1                            SSA00560
*                                                                   SSA00570
* Ways to get data                                                  SSA00580
        SPACE 1                                                     SSA00590
SS_CLI_METHOD_READ            EQU     0                            SSA00600
SS_CLI_METHOD_PEEK            EQU     1                            SSA00610
SS_CLI_METHOD_DISCARD         EQU     2                            SSA00620
*-------------------------------------------------------------*     SSA00630
*     Definitions for services function                       *     SSA00640
*-------------------------------------------------------------*     SSA00650
        SPACE 1                                                     SSA00660
*                                                                   SSA00670
*     initialize client data queues                                 SSA00680
*                                                                   SSA00690
        SPACE 1                                                     SSA00700
        &$XXTRN  BKWIIN                                             SSA00710
SSCLIENTDATAINIT              EQU     BKWIIN                        SSA00720
        SPACE 1                                                     SSA00730
BKWIIN_PLIST                  DSECT                                 SSA00740
BKWIIN_PLIST_RC               DS      A   * return code             SSA00750
BKWIIN_PLIST_RE               DS      A   * reason code             SSA00760
BKWIIN_PLIST_CB               DS      A   * C-block addr            SSA00770
BKWIIN_PLIST_SUBPOOL          DS      A   * subpool name            SSA00780
BKWIIN_PLIST_LENGTH           EQU     *-BKWIIN_PLIST                SSA00790
                              VMASMMAX                              SSA00800
        SPACE 1                                                     SSA00810
*                                                                   SSA00820
*     terminate client data queues                                  SSA00830
*                                                                   SSA00840
        SPACE 1                                                     SSA00850
        &$XXTRN  BKWITM                                             SSA00860
SSCLIENTDATATERM              EQU     BKWITM                        SSA00870
        SPACE 1                                                     SSA00880
BKWITM_PLIST                  DSECT                                 SSA00890
BKWITM_PLIST_RC               DS      A   * return code             SSA00900
BKWITM_PLIST_RE               DS      A   * reason code             SSA00910
BKWITM_PLIST_CB               DS      A   * C-block addr            SSA00920
BKWITM_PLIST_LENGTH           EQU     *-BKWITM_PLIST                SSA00930
                              VMASMMAX                              SSA00940
        SPACE 1                                                     SSA00950
*                                                                   SSA00960
*     get input from client C-block                                 SSA00970
*                                                                   SSA00980
        SPACE 1                                                     SSA00990
```

```
        &$XXTRN  BKWIDG                                          SSA01000
SSCLIENTDATAGET                EQU      BKWIDG                   SSA01010
        SPACE 1                                                 SSA01020
BKWIDG_PLIST                   DSECT                             SSA01030
BKWIDG_PLIST_RC                DS       A  * return code         SSA01040
BKWIDG_PLIST_RE                DS       A  * reason code         SSA01050
BKWIDG_PLIST_INS               DS       A  * instance or ld?     SSA01060
BKWIDG_PLIST_CB                DS       A  * C-block addr        SSA01070
BKWIDG_PLIST_GM                DS       A  * get method          SSA01080
BKWIDG_PLIST_ALET              DS       A  * ALET                SSA01090
BKWIDG_PLIST_BUF               DS       A  * buffer              SSA01100
BKWIDG_PLIST_AM                DS       A  * amt wanted          SSA01110
BKWIDG_PLIST_AG                DS       A  * amt given           SSA01120
BKWIDG_PLIST_AL                DS       A  * amt left            SSA01130
BKWIDG_PLIST_LENGTH            EQU      *-BKWIDG_PLIST           SSA01140
                               VMASMMAX                         SSA01150
        SPACE 1                                                 SSA01160
*                                                               SSA01170
*    put output onto client C-block                             SSA01180
*                                                               SSA01190
        SPACE 1                                                 SSA01200
        &$XXTRN  BKWIDP                                          SSA01210
SSCLIENTDATAPUT                EQU      BKWIDP                   SSA01220
        SPACE 1                                                 SSA01230
BKWIDP_PLIST                   DSECT                             SSA01240
BKWIDP_PLIST_RC                DS       A  * return code         SSA01250
BKWIDP_PLIST_RE                DS       A  * reason code         SSA01260
BKWIDP_PLIST_INS               DS       A  * instance or ld?     SSA01270
BKWIDP_PLIST_CB                DS       A  * C-block addr        SSA01280
BKWIDP_PLIST_ALET              DS       A  * ALET                SSA01290
BKWIDP_PLIST_BUF               DS       A  * buffer              SSA01300
BKWIDP_PLIST_AP                DS       A  * amt to put          SSA01310
BKWIDP_PLIST_NA                DS       A  * new amount          SSA01320
BKWIDP_PLIST_LENGTH            EQU      *-BKWIDP_PLIST           SSA01330
                               VMASMMAX                         SSA01340
        SPACE 1                                                 SSA01350
*----------------------------------------------------------------*  SSA01360
*    End of declarations                                      *  SSA01370
*----------------------------------------------------------------*  SSA01380
        EJECT                                                  SSA01390
        POP   PRINT                                            SSA01400
        MEND                                                   SSA01410
```

Enrollment Bindings (SSASMENR MACRO)

```
        MACRO                                                  SSA00010
        SSASMENR &WEAK=                                    SSA00020
        AGO   .@ASMSR1                                         SSA00030
.*  Branch around prolog so it is not included in listings     *  SSA00040
.***********************************************************************  SSA00050
.*                                                          *  SSA00060
.*  NAME     -  Reusable Server Kernel   services bindings  *  SSA00070
.*                                                          *  SSA00080
.*  FUNCTION -  Language bindings for enrollment services   *  SSA00090
.*                                                          *  SSA00100
.*  COPYRIGHT -                                   @VR2OZOZ SSA00110
.*                                                @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992 @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM       @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083      @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                        @VR2OZOZ SSA00160
.*                                                       *  SSA00170
.*  STATUS - Version 2 Release 4                   @VR2OZOZ SSA00180
.*                                                       *  SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4   *  SSA00200
.***********************************************************************  SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4    @VR24PVM SSA00220
.***********************************************************************  SSA00230
.@ASMSR1 ANOP                                               SSA00240
        PUSH  PRINT                                           SSA00250
        AIF   ('&SYSPARM' NE 'SUP').ASMSR2            SSA00260
        PRINT OFF,NOGEN                                       SSA00270
.ASMSR2  ANOP                                               SSA00280
        LCLC  &$XXTRN                                         SSA00290
&$XXTRN  SETC  'EXTRN'                                        SSA00300
        AIF   ('&WEAK' NE 'YES').ASMSR3                SSA00310
&$XXTRN  SETC  'WXTRN'                                        SSA00320
.ASMSR3  ANOP                                               SSA00330
*----------------------------------------------------------------*  SSA00340
```

```
*      Return and reason codes for services functions           *    SSA00350
*------------------------------------------------------------------*    SSA00360
        SPACE 1                                                         SSA00370
*                                                                       SSA00380
* return codes                                                          SSA00390
SS_ENR_RC_SUCCESS               EQU       0                             SSA00400
SS_ENR_RC_WARNING               EQU       4                             SSA00410
SS_ENR_RC_ERROR                 EQU       8                             SSA00420
SS_ENR_RC_ABEND                 EQU       12                            SSA00430
*                                                                       SSA00440
* reason codes                                                          SSA00450
SS_ENR_RE_SUCCESS               EQU       0                             SSA00460
SS_ENR_RE_DB_NOT_FOUND          EQU       1001                          SSA00470
SS_ENR_RE_REC_NOT_FOUND         EQU       1002                          SSA00480
SS_ENR_RE_TRUNCATED             EQU       1003                          SSA00490
SS_ENR_RE_DIRTY                 EQU       1004                          SSA00500
SS_ENR_RE_REC_EXISTS            EQU       1005                          SSA00510
SS_ENR_RE_BAD_LENGTH            EQU       1006                          SSA00520
SS_ENR_RE_BAD_DROPTYPE          EQU       1007                          SSA00530
SS_ENR_RE_NO_STORAGE            EQU       1008                          SSA00540
SS_ENR_RE_CLOSE_FAIL            EQU       1009                          SSA00550
SS_ENR_RE_WRITE_FAIL            EQU       1010                          SSA00560
SS_ENR_RE_BAD_METHOD            EQU       1011                          SSA00570
SS_ENR_RE_OPEN_FAIL             EQU       1012                          SSA00580
SS_ENR_RE_GWU_FAIL              EQU       1013                          SSA00590
SS_ENR_RE_POINT_FAIL            EQU       1014                          SSA00600
SS_ENR_RE_EXIST_FAIL            EQU       1015                          SSA00610
SS_ENR_RE_NOT_SFS               EQU       1016                          SSA00620
SS_ENR_RE_NOT_V                 EQU       1017                          SSA00630
SS_ENR_RE_DSCR_FAIL             EQU       1018                          SSA00640
SS_ENR_RE_READ_FAIL             EQU       1019                          SSA00650
SS_ENR_RE_DB_EXISTS             EQU       1020                          SSA00660
SS_ENR_RE_COMM_FAIL             EQU       1021                          SSA00670
SS_ENR_RE_NOT_DISK              EQU       1022                          SSA00680
SS_ENR_RE_BAD_KIND              EQU       1023                          SSA00690
SS_ENR_RE_NEW_FILE              EQU       1024                          SSA00700
SS_ENR_RE_NO_SETS               EQU       1025                          SSA00710
SS_ENR_RE_SET_EMPTY             EQU       1026                          SSA00720
        SPACE 1                                                         SSA00730
*                                                                       SSA00740
* API maxima                                                            SSA00750
SS_ENR_INDEX_WIDTH              EQU       64                            SSA00760
SS_ENR_MAX_DATA                 EQU       65450                         SSA00770
        SPACE 1                                                         SSA00780
*                                                                       SSA00790
* KIND types                                                            SSA00800
SS_ENR_KIND_MEMORY              EQU       0                             SSA00810
SS_ENR_KIND_DISK                EQU       1                             SSA00820
        SPACE 1                                                         SSA00830
*                                                                       SSA00840
* INSERT types                                                          SSA00850
SS_ENR_INSERT_NEW               EQU       0                             SSA00860
SS_ENR_INSERT_REPLACE           EQU       1                             SSA00870
        SPACE 1                                                         SSA00880
*                                                                       SSA00890
* DROP types                                                            SSA00900
SS_ENR_DROP_COMMIT              EQU       0                             SSA00910
SS_ENR_DROP_ROLLBACK            EQU       1                             SSA00920
        SPACE 1                                                         SSA00930
*------------------------------------------------------------------*    SSA00940
*      Definitions for enrollment services                        *    SSA00950
*------------------------------------------------------------------*    SSA00960
        SPACE 1                                                         SSA00970
*                                                                       SSA00980
*      load enrollment data base                                        SSA00990
*                                                                       SSA01000
        SPACE 1                                                         SSA01010
        &$XXTRN   BKWJLO                                                SSA01020
SSENROLLLOAD                    EQU       BKWJLO                        SSA01030
        SPACE 1                                                         SSA01040
BKWJLO_PLIST                    DSECT                                   SSA01050
BKWJLO_PLIST_RC                 DS        A   * return code             SSA01060
BKWJLO_PLIST_RE                 DS        A   * reason code             SSA01070
BKWJLO_PLIST_DBASE              DS        A   * dbase name              SSA01080
BKWJLO_PLIST_DS_KIND            DS        A   * DS kind                 SSA01090
BKWJLO_PLIST_DS_SIZE            DS        A   * DS size                 SSA01100
BKWJLO_PLIST_FN                 DS        A   * filename                SSA01110
BKWJLO_PLIST_FNL                DS        A   * filename length         SSA01120
BKWJLO_PLIST_LENGTH             EQU       *-BKWJLO_PLIST                SSA01130
                                VMASMMAX                                SSA01140
        SPACE 1                                                         SSA01150
*                                                                       SSA01160
```

```
*      drop enrollment data base                                        SSA01170
*                                                                       SSA01180
       SPACE 1                                                          SSA01190
       &$XXTRN  BKWJDP                                                  SSA01200
SSENROLLDROP                  EQU       BKWJDP                          SSA01210
       SPACE 1                                                          SSA01220
BKWJDP_PLIST                  DSECT                                     SSA01230
BKWJDP_PLIST_RC               DS        A   * return code               SSA01240
BKWJDP_PLIST_RE               DS        A   * reason code               SSA01250
BKWJDP_PLIST_DBASE            DS        A   * dbase name                SSA01260
BKWJDP_PLIST_DT               DS        A   * drop type                 SSA01270
BKWJDP_PLIST_LENGTH           EQU       *-BKWJDP_PLIST                  SSA01280
                              VMASMMAX                                  SSA01290
       SPACE 1                                                          SSA01300
*                                                                       SSA01310
*      commit enrollment data base                                      SSA01320
*                                                                       SSA01330
       SPACE 1                                                          SSA01340
       &$XXTRN  BKWJCM                                                  SSA01350
SSENROLLCOMMIT                EQU       BKWJCM                          SSA01360
       SPACE 1                                                          SSA01370
BKWJCM_PLIST                  DSECT                                     SSA01380
BKWJCM_PLIST_RC               DS        A   * return code               SSA01390
BKWJCM_PLIST_RE               DS        A   * reason code               SSA01400
BKWJCM_PLIST_DBASE            DS        A   * dbase name                SSA01410
BKWJCM_PLIST_LENGTH           EQU       *-BKWJCM_PLIST                  SSA01420
                              VMASMMAX                                  SSA01430
       SPACE 1                                                          SSA01440
*                                                                       SSA01450
*      list data bases                                                  SSA01460
*                                                                       SSA01470
       SPACE 1                                                          SSA01480
       &$XXTRN  BKWJDL                                                  SSA01490
SSENROLLLIST                  EQU       BKWJDL                          SSA01500
       SPACE 1                                                          SSA01510
BKWJDL_PLIST                  DSECT                                     SSA01520
BKWJDL_PLIST_RC               DS        A   * return code               SSA01530
BKWJDL_PLIST_RE               DS        A   * reason code               SSA01540
BKWJDL_PLIST_CB               DS        A   * C-block                   SSA01550
BKWJDL_PLIST_LENGTH           EQU       *-BKWJDL_PLIST                  SSA01560
                              VMASMMAX                                  SSA01570
       SPACE 1                                                          SSA01580
*                                                                       SSA01590
*      insert record                                                    SSA01600
*                                                                       SSA01610
       SPACE 1                                                          SSA01620
       &$XXTRN  BKWJRI                                                  SSA01630
SSENROLLRECORDINSERT          EQU       BKWJRI                          SSA01640
       SPACE 1                                                          SSA01650
BKWJRI_PLIST                  DSECT                                     SSA01660
BKWJRI_PLIST_RC               DS        A   * return code               SSA01670
BKWJRI_PLIST_RE               DS        A   * reason code               SSA01680
BKWJRI_PLIST_DBASE            DS        A   * dbase name                SSA01690
BKWJRI_PLIST_INDEX            DS        A   * index                     SSA01700
BKWJRI_PLIST_DATA             DS        A   * data                      SSA01710
BKWJRI_PLIST_DATAL            DS        A   * data length               SSA01720
BKWJRI_PLIST_REP              DS        A   * replace?                  SSA01730
BKWJRI_PLIST_LENGTH           EQU       *-BKWJRI_PLIST                  SSA01740
                              VMASMMAX                                  SSA01750
       SPACE 1                                                          SSA01760
*                                                                       SSA01770
*      remove record                                                    SSA01780
*                                                                       SSA01790
       SPACE 1                                                          SSA01800
       &$XXTRN  BKWJRR                                                  SSA01810
SSENROLLRECORDREMOVE          EQU       BKWJRR                          SSA01820
       SPACE 1                                                          SSA01830
BKWJRR_PLIST                  DSECT                                     SSA01840
BKWJRR_PLIST_RC               DS        A   * return code               SSA01850
BKWJRR_PLIST_RE               DS        A   * reason code               SSA01860
BKWJRR_PLIST_DBASE            DS        A   * dbase name                SSA01870
BKWJRR_PLIST_INDEX            DS        A   * index                     SSA01880
BKWJRR_PLIST_LENGTH           EQU       *-BKWJRR_PLIST                  SSA01890
                              VMASMMAX                                  SSA01900
       SPACE 1                                                          SSA01910
*                                                                       SSA01920
*      list records                                                     SSA01930
*                                                                       SSA01940
       SPACE 1                                                          SSA01950
       &$XXTRN  BKWJRL                                                  SSA01960
SSENROLLRECORDLIST            EQU       BKWJRL                          SSA01970
       SPACE 1                                                          SSA01980
```

```
BKWJRL_PLIST                           DSECT                            SSA01990
BKWJRL_PLIST_RC                        DS        A   * return code      SSA02000
BKWJRL_PLIST_RE                        DS        A   * reason code      SSA02010
BKWJRL_PLIST_DBASE                     DS        A   * dbase name       SSA02020
BKWJRL_PLIST_CB                        DS        A   * C-block          SSA02030
BKWJRL_PLIST_LENGTH                    EQU       *-BKWJRL_PLIST         SSA02040
                                       VMASMMAX                         SSA02050
        SPACE 1                                                         SSA02060
*                                                                       SSA02070
*    get record                                                        SSA02080
*                                                                       SSA02090
        SPACE 1                                                         SSA02100
        &$XXTRN   BKWJRG                                                SSA02110
SSENROLLRECORDGET                      EQU       BKWJRG                 SSA02120
        SPACE 1                                                         SSA02130
BKWJRG_PLIST                           DSECT                            SSA02140
BKWJRG_PLIST_RC                        DS        A   * return code      SSA02150
BKWJRG_PLIST_RE                        DS        A   * reason code      SSA02160
BKWJRG_PLIST_DBASE                     DS        A   * dbase name       SSA02170
BKWJRG_PLIST_INDEX                     DS        A   * index            SSA02180
BKWJRG_PLIST_BUF                       DS        A   * buffer           SSA02190
BKWJRG_PLIST_BUFS                      DS        A   * buffer size      SSA02200
BKWJRG_PLIST_AR                        DS        A   * amt returned     SSA02210
BKWJRG_PLIST_LENGTH                    EQU       *-BKWJRG_PLIST         SSA02220
                                       VMASMMAX                         SSA02230
        SPACE 1                                                         SSA02240
*-------------------------------------------------------------------*  SSA02250
*     End of declarations                                        *     SSA02260
*-------------------------------------------------------------------*  SSA02270
        EJECT                                                          SSA02280
        POP   PRINT                                                    SSA02290
        MEND                                                           SSA02300
```

Memory Bindings (SSASMMEM MACRO)

```
        MACRO                                                          SSA00010
        SSASMMEM &WEAK=                                        SSA00020
        AGO   .@ASMME1                                                 SSA00030
.*  Branch around prolog so it is not included in listings      *     SSA00040
.*********************************************************************  SSA00050
.*                                                              *     SSA00060
.*  NAME     -  Reusable Server Kernel   memory bindings        *     SSA00070
.*                                                              *     SSA00080
.*  FUNCTION -  Defines memory constants and dsects             *     SSA00090
.*                                                              *     SSA00100
.*  COPYRIGHT -                                         @VR2OZOZ SSA00110
.*                                                      @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992       @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM             @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083            @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                              @VR2OZOZ SSA00160
.*                                                              *     SSA00170
.*  STATUS - Version 2 Release 4                        @VR2OZOZ SSA00180
.*                                                              *     SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4       *     SSA00200
.*********************************************************************  SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4   @VR74PVM SSA00220
.*********************************************************************  SSA00230
.@ASMME1 ANOP                                                          SSA00240
        PUSH  PRINT                                                    SSA00250
        AIF   ('&ARM' NE 'SUP').ASMME2                        SSA00260
        PRINT OFF,NOGEN                                                SSA00270
.ASMME2  ANOP                                                          SSA00280
        LCLC  &$XXTRN                                                  SSA00290
&$XXTRN  SETC  'EXTRN'                                                 SSA00300
        AIF   ('&WEAK' NE 'YES').ASMME3                       SSA00310
&$XXTRN  SETC  'WXTRN'                                                 SSA00320
.ASMME3  ANOP                                                          SSA00330
*-------------------------------------------------------------------*  SSA00340
*     Return and reason codes for memory functions               *     SSA00350
*-------------------------------------------------------------------*  SSA00360
        SPACE 1                                                        SSA00370
*                                                                       SSA00380
* return codes                                                          SSA00390
SS_MEM_RC_SUCCESS                      EQU       0                     SSA00400
SS_MEM_RC_WARNING                      EQU       4                     SSA00410
SS_MEM_RC_ERROR                        EQU       8                     SSA00420
SS_MEM_RC_ABEND                        EQU       12                    SSA00430
*                                                                       SSA00440
```

```
* reason codes                                              SSA00450
SS_MEM_RE_SUCCESS              EQU    0                      SSA00460
SS_MEM_RE_OUT_OF_STORAGE      EQU    801                    SSA00470
SS_MEM_RE_BAD_AMOUNT          EQU    802                    SSA00480
SS_MEM_RE_BAD_ALIGN           EQU    803                    SSA00490
SS_MEM_RE_NO_SUBPOOL          EQU    804                    SSA00500
SS_MEM_RE_NOT_ALLOC           EQU    805                    SSA00510
SS_MEM_RE_SUBPOOL_DELETED     EQU    806                    SSA00520
SS_MEM_RE_SPD_FAIL            EQU    807                    SSA00530
SS_MEM_RE_BAD_KEY             EQU    808                    SSA00540
SS_MEM_RE_SUBPOOL_EXISTS      EQU    809                    SSA00550
SS_MEM_RE_SPCC_FAIL           EQU    810                    SSA00560
SS_MEM_RE_SPLA_FAIL           EQU    811                    SSA00570
*                                                           SSA00580
*-------------------------------------------------------*   SSA00590
*     Constants for memory functions                    *   SSA00600
*-------------------------------------------------------*   SSA00610
        SPACE 1                                             SSA00620
*                                                           SSA00630
*     Alignment attributes                                  SSA00640
*                                                           SSA00650
SS_MEM_ALIGN_NORM             EQU    0                      SSA00660
SS_MEM_ALIGN_PAGE             EQU    1                      SSA00670
        SPACE 1                                             SSA00680
*-------------------------------------------------------*   SSA00690
*     Definitions for memory functions                  *   SSA00700
*-------------------------------------------------------*   SSA00710
*                                                           SSA00720
*     create a data space                                   SSA00730
*                                                           SSA00740
        SPACE 1                                             SSA00750
        &$XXTRN  BKWMCR                                     SSA00760
SSMEMORYCREATEDS              EQU    BKWMCR                  SSA00770
        SPACE 1                                             SSA00780
BKWMCR_PLIST                  DSECT                          SSA00790
BKWMCR_PLIST_RC               DS    A   * return code       SSA00800
BKWMCR_PLIST_RE               DS    A   * reason code       SSA00810
BKWMCR_PLIST_SUBPOOL          DS    A   * subpool name      SSA00820
BKWMCR_PLIST_SIZE             DS    A   * DS size (pages)   SSA00830
BKWMCR_PLIST_KEY              DS    A   * storage key       SSA00840
BKWMCR_PLIST_OCOUNT           DS    A   * option count      SSA00850
BKWMCR_PLIST_OARRAY           DS    A   * option array      SSA00860
BKWMCR_PLIST_ASIT             DS    A   * DS ASIT           SSA00870
BKWMCR_PLIST_ALET             DS    A   * DS ALET           SSA00880
BKWMCR_PLIST_LENGTH           EQU    *-BKWMCR_PLIST         SSA00890
                              VMASMMAX                       SSA00900
        SPACE 1                                             SSA00910
*                                                           SSA00920
*     allocate memory                                       SSA00930
*                                                           SSA00940
        SPACE 1                                             SSA00950
        &$XXTRN  BKWMAL                                     SSA00960
SSMEMORYALLOCATE              EQU    BKWMAL                  SSA00970
        SPACE 1                                             SSA00980
BKWMAL_PLIST                  DSECT                          SSA00990
BKWMAL_PLIST_RC               DS    A   * return code       SSA01000
BKWMAL_PLIST_RE               DS    A   * reason code       SSA01010
BKWMAL_PLIST_LB               DS    A   * lower bound       SSA01020
BKWMAL_PLIST_UB               DS    A   * upper bound       SSA01030
BKWMAL_PLIST_SUBPOOL          DS    A   * subpool name      SSA01040
BKWMAL_PLIST_ALIGN            DS    A   * align type        SSA01050
BKWMAL_PLIST_BA               DS    A   * buffer address    SSA01060
BKWMAL_PLIST_BG               DS    A   * bytes gotten      SSA01070
BKWMAL_PLIST_LENGTH           EQU    *-BKWMAL_PLIST         SSA01080
                              VMASMMAX                       SSA01090
        SPACE 1                                             SSA01100
*                                                           SSA01110
*     release memory                                        SSA01120
*                                                           SSA01130
        SPACE 1                                             SSA01140
        &$XXTRN  BKWMRE                                     SSA01150
SSMEMORYRELEASE               EQU    BKWMRE                  SSA01160
        SPACE 1                                             SSA01170
BKWMRE_PLIST                  DSECT                          SSA01180
BKWMRE_PLIST_RC               DS    A   * return code       SSA01190
BKWMRE_PLIST_RE               DS    A   * reason code       SSA01200
BKWMRE_PLIST_BR               DS    A   * bytes released    SSA01210
BKWMRE_PLIST_SUBPOOL          DS    A   * subpool name      SSA01220
BKWMRE_PLIST_BA               DS    A   * buffer address    SSA01230
BKWMRE_PLIST_LENGTH           EQU    *-BKWMRE_PLIST         SSA01240
                              VMASMMAX                       SSA01250
        SPACE 1                                             SSA01260
```

```
*                                                               SSA01270
*    delete subpool                                             SSA01280
*                                                               SSA01290
      SPACE 1                                                   SSA01300
      &$XXTRN   BKWMDE                                          SSA01310
SSMEMORYDELETE                EQU       BKWMDE                  SSA01320
      SPACE 1                                                   SSA01330
BKWMDE_PLIST                  DSECT                             SSA01340
BKWMDE_PLIST_RC               DS        A  * return code        SSA01350
BKWMDE_PLIST_RE               DS        A  * reason code        SSA01360
BKWMDE_PLIST_SUBPOOL          DS        A  * subpool name       SSA01370
BKWMDE_PLIST_LENGTH           EQU       *-BKWMDE_PLIST          SSA01380
                              VMASMMAX                          SSA01390
      SPACE 1                                                   SSA01400
*---------------------------------------------------------------*   SSA01410
*    End of declarations                                   *    SSA01420
*---------------------------------------------------------------*   SSA01430
      EJECT                                                     SSA01440
      POP   PRINT                                               SSA01450
      MEND                                                      SSA01460
```

Storage Group Bindings (SSASMSGP MACRO)

```
      MACRO                                                     SSA00010
      SSASMSGP &WEAK=                                 SSA00020
      AGO   .@ASMSG1                                            SSA00030
.*  Branch around prolog so it is not included in listings      *    SSA00040
.*****************************************************************    SSA00050
.*                                                        *    SSA00060
.*  NAME     -  Reusable Server Kernel   storage group bindings  *    SSA00070
.*                                                        *    SSA00080
.*  FUNCTION  -  Defines the storage group constants and dsects  *    SSA00090
.*                                                        *    SSA00100
.*  COPYRIGHT -                                       @VR2OZOZ SSA00110
.*                                                    @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992     @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM           @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083          @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                            @VR2OZOZ SSA00160
.*                                                        *    SSA00170
.*  STATUS - VM/ESA Version 2 Release 4                @VR2OZOZ SSA00180
.*                                                        *    SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4   *    SSA00200
.*****************************************************************    SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4      @VR2LMVM SSA00220
.*****************************************************************    SSA00230
.@ASMSG1 ANOP                                                  SSA00240
      PUSH   PRINT                                              SSA00250
      AIF   ('&SYSPARM' NE 'SUP').ASMSG2             SSA00260
      PRINT OFF,NOGEN                                           SSA00270
.ASMSG2  ANOP                                                  SSA00280
      LCLC   &$XXTRN                                            SSA00290
&$XXTRN   SETC   'EXTRN'                                        SSA00300
      AIF   ('&WEAK' NE 'YES').ASMSG3               SSA00310
&$XXTRN   SETC   'WXTRN'                                        SSA00320
.ASMSG3  ANOP                                                  SSA00330
*---------------------------------------------------------------*   SSA00340
*    Return and reason codes for storage group functions    *    SSA00350
*---------------------------------------------------------------*   SSA00360
      SPACE 1                                                   SSA00370
* return codes                                                 SSA00380
SS_SGP_RC_SUCCESS             EQU       0                       SSA00390
SS_SGP_RC_WARNING             EQU       4                       SSA00400
SS_SGP_RC_ERROR               EQU       8                       SSA00410
SS_SGP_RC_ABEND               EQU       12                      SSA00420
*                                                               SSA00430
* reason codes                                                 SSA00440
SS_SGP_RE_SUCCESS             EQU       0                       SSA00450
SS_SGP_RE_TOO_MANY            EQU       601                     SSA00460
SS_SGP_RE_NOT_FOUND           EQU       602                     SSA00470
SS_SGP_RE_OUT_OF_STORAGE      EQU       603                     SSA00480
SS_SGP_RE_MX_FAIL             EQU       604                     SSA00490
SS_SGP_RE_INIT_DONE           EQU       605                     SSA00500
SS_SGP_RE_EXISTS              EQU       607                     SSA00510
SS_SGP_RE_VDQ_FAIL            EQU       608                     SSA00520
SS_SGP_RE_ONLINE              EQU       609                     SSA00530
SS_SGP_RE_OFFLINE             EQU       610                     SSA00540
SS_SGP_RE_Q_FAIL              EQU       611                     SSA00550
SS_SGP_RE_CV_FAIL             EQU       612                     SSA00560
```

```
SS_SGP_RE_E_FAIL                EQU     613                             SSA00570
SS_SGP_RE_MAINT                 EQU     614                             SSA00580
SS_SGP_RE_DS_FAIL               EQU     615                             SSA00590
SS_SGP_RE_POOL_FAIL             EQU     616                             SSA00600
SS_SGP_RE_MAP_FAIL              EQU     617                             SSA00610
SS_SGP_RE_BAD_ATTRIB            EQU     618                             SSA00620
SS_SGP_RE_REWRITE_FAIL          EQU     619                             SSA00630
SS_SGP_RE_READ_ONLY             EQU     620                             SSA00640
SS_SGP_RE_OUT_OF_RANGE          EQU     622                             SSA00650
SS_SGP_RE_WRONG_MODE            EQU     623                             SSA00660
SS_SGP_RE_IO_FAIL               EQU     624                             SSA00670
SS_SGP_RE_DIAG_250_FAIL         EQU     625                             SSA00680
SS_SGP_RE_TOO_BIG               EQU     626                             SSA00690
SS_SGP_RE_BAD_NAME              EQU     628                             SSA00700
SS_SGP_RE_NAME_IN_USE           EQU     629                             SSA00710
        SPACE 1                                                         SSA00720
*                                                                       SSA00730
* attributes                                                            SSA00740
SS_SGP_ATTRIB_DS                EQU     0                               SSA00750
SS_SGP_ATTRIB_NO_DS             EQU     1                               SSA00760
SS_SGP_ATTRIB_BLOCK_RW          EQU     2                               SSA00770
SS_SGP_ATTRIB_BLOCK_RO          EQU     3                               SSA00780
SS_SGP_ATTRIB_OFFLINE           EQU     7                               SSA00790
        SPACE 1                                                         SSA00800
*-------------------------------------------------------------------*   SSA00810
*     Definitions for storage group functions                   *       SSA00820
*-------------------------------------------------------------------*   SSA00830
        SPACE 1                                                         SSA00840
*                                                                       SSA00850
*     storage group create                                             SSA00860
*                                                                       SSA00870
        SPACE 1                                                         SSA00880
        &$XXTRN  BKWSGC                                                 SSA00890
SSSGPCREATE                     EQU     BKWSGC                          SSA00900
        SPACE 1                                                         SSA00910
BKWSGC_PLIST                    DSECT                                   SSA00920
BKWSGC_PLIST_RC                 DS      A   * return code               SSA00930
BKWSGC_PLIST_RE                 DS      A   * reason code               SSA00940
BKWSGC_PLIST_SGN                DS      A   * sgp number                SSA00950
BKWSGC_PLIST_VDC                DS      A   * vdev count                SSA00960
BKWSGC_PLIST_VDA                DS      A   * vdev array                SSA00970
BKWSGC_PLIST_AC                 DS      A   * attrib count              SSA00980
BKWSGC_PLIST_AA                 DS      A   * attrib array              SSA00990
BKWSGC_PLIST_LENGTH             EQU     *-BKWSGC_PLIST                  SSA01000
                                VMASMMAX                                SSA01010
        SPACE 1                                                         SSA01020
*                                                                       SSA01030
*     storage group delete                                             SSA01040
*                                                                       SSA01050
        SPACE 1                                                         SSA01060
        &$XXTRN  BKWSGD                                                 SSA01070
SSSGPDELETE                     EQU     BKWSGD                          SSA01080
        SPACE 1                                                         SSA01090
BKWSGD_PLIST                    DSECT                                   SSA01100
BKWSGD_PLIST_RC                 DS      A   * return code               SSA01110
BKWSGD_PLIST_RE                 DS      A   * reason code               SSA01120
BKWSGD_PLIST_SGN                DS      A   * sgp number                SSA01130
BKWSGD_PLIST_LENGTH             EQU     *-BKWSGD_PLIST                  SSA01140
                                VMASMMAX                                SSA01150
        SPACE 1                                                         SSA01160
*                                                                       SSA01170
*     storage group find                                               SSA01180
*                                                                       SSA01190
        SPACE 1                                                         SSA01200
        &$XXTRN  BKWSGF                                                 SSA01210
SSSGPFIND                       EQU     BKWSGF                          SSA01220
        SPACE 1                                                         SSA01230
BKWSGF_PLIST                    DSECT                                   SSA01240
BKWSGF_PLIST_RC                 DS      A   * return code               SSA01250
BKWSGF_PLIST_RE                 DS      A   * reason code               SSA01260
BKWSGF_PLIST_SGNAME             DS      A   * sg name                   SSA01270
BKWSGF_PLIST_SGN                DS      A   * sgp number                SSA01280
BKWSGF_PLIST_IOMODE             DS      A   * I/O mode                  SSA01290
BKWSGF_PLIST_TOTAL              DS      A   * total blocks              SSA01300
BKWSGF_PLIST_LENGTH             EQU     *-BKWSGF_PLIST                  SSA01310
                                VMASMMAX                                SSA01320
        SPACE 1                                                         SSA01330
*                                                                       SSA01340
*     storage group list (what's defined?)                             SSA01350
*                                                                       SSA01360
        SPACE 1                                                         SSA01370
        &$XXTRN  BKWSGL                                                 SSA01380
```

```
SSSGPLIST                       EQU       BKWSGL                          SSA01390
        SPACE 1                                                           SSA01400
BKWSGL_PLIST                    DSECT                                     SSA01410
BKWSGL_PLIST_RC                 DS        A   * return code               SSA01420
BKWSGL_PLIST_RE                 DS        A   * reason code               SSA01430
BKWSGL_PLIST_NX                 DS        A   * number expected           SSA01440
BKWSGL_PLIST_NF                 DS        A   * number filled             SSA01450
BKWSGL_PLIST_SGNA               DS        A   * sgp number array          SSA01460
BKWSGL_PLIST_LENGTH             EQU       *-BKWSGL_PLIST                  SSA01470
                                VMASMMAX                                  SSA01480
        SPACE 1                                                           SSA01490
*                                                                         SSA01500
*     storage group query                                                SSA01510
*                                                                         SSA01520
        SPACE 1                                                           SSA01530
        &$XXTRN   BKWSGQ                                                  SSA01540
SSSGPQUERY                      EQU       BKWSGQ                          SSA01550
        SPACE 1                                                           SSA01560
BKWSGQ_PLIST                    DSECT                                     SSA01570
BKWSGQ_PLIST_RC                 DS        A   * return code               SSA01580
BKWSGQ_PLIST_RE                 DS        A   * reason code               SSA01590
BKWSGQ_PLIST_SGN                DS        A   * sgp number                SSA01600
BKWSGQ_PLIST_SGNAME             DS        A   * sg name                   SSA01610
BKWSGQ_PLIST_IOMODE             DS        A   * I/O mode                  SSA01620
BKWSGQ_PLIST_TOTAL              DS        A   * total blocks              SSA01630
BKWSGQ_PLIST_STATUS             DS        A   * status word               SSA01640
BKWSGQ_PLIST_AX                 DS        A   * attributes expected       SSA01650
BKWSGQ_PLIST_AF                 DS        A   * attributes filled in      SSA01660
BKWSGQ_PLIST_AA                 DS        A   * attribute array           SSA01670
BKWSGQ_PLIST_VX                 DS        A   * vdevs expected            SSA01680
BKWSGQ_PLIST_VF                 DS        A   * vdevs filled in           SSA01690
BKWSGQ_PLIST_VA                 DS        A   * vdev array                SSA01700
BKWSGQ_PLIST_BA                 DS        A   * blocks array              SSA01710
BKWSGQ_PLIST_LENGTH             EQU       *-BKWSGQ_PLIST                  SSA01720
                                VMASMMAX                                  SSA01730
        SPACE 1                                                           SSA01740
*                                                                         SSA01750
*     storage group read                                                 SSA01760
*                                                                         SSA01770
        SPACE 1                                                           SSA01780
        &$XXTRN   BKWSGR                                                  SSA01790
SSSGPREAD                       EQU       BKWSGR                          SSA01800
        SPACE 1                                                           SSA01810
BKWSGR_PLIST                    DSECT                                     SSA01820
BKWSGR_PLIST_RC                 DS        A   * return code               SSA01830
BKWSGR_PLIST_RE                 DS        A   * reason code               SSA01840
BKWSGR_PLIST_SGN                DS        A   * sgp number                SSA01850
BKWSGR_PLIST_PN                 DS        A   * page number               SSA01860
BKWSGR_PLIST_PC                 DS        A   * number of pages           SSA01870
BKWSGR_PLIST_ALET               DS        A   * buffer ALET               SSA01880
BKWSGR_PLIST_BUF                DS        A   * buffer                    SSA01890
BKWSGR_PLIST_LENGTH             EQU       *-BKWSGR_PLIST                  SSA01900
                                VMASMMAX                                  SSA01910
        SPACE 1                                                           SSA01920
*                                                                         SSA01930
*     storage group start (like a mount)                                 SSA01940
*                                                                         SSA01950
        SPACE 1                                                           SSA01960
        &$XXTRN   BKWSGS                                                  SSA01970
SSSGPSTART                      EQU       BKWSGS                          SSA01980
        SPACE 1                                                           SSA01990
BKWSGS_PLIST                    DSECT                                     SSA02000
BKWSGS_PLIST_RC                 DS        A   * return code               SSA02010
BKWSGS_PLIST_RE                 DS        A   * reason code               SSA02020
BKWSGS_PLIST_SGN                DS        A   * sgp number                SSA02030
BKWSGS_PLIST_SGNAME             DS        A   * sgp name                  SSA02040
BKWSGS_PLIST_AC                 DS        A   * attribute count           SSA02050
BKWSGS_PLIST_AA                 DS        A   * attribute array           SSA02060
BKWSGS_PLIST_LENGTH             EQU       *-BKWSGS_PLIST                  SSA02070
                                VMASMMAX                                  SSA02080
        SPACE 1                                                           SSA02090
*                                                                         SSA02100
*     storage group stop (like a dismount)                               SSA02110
*                                                                         SSA02120
        SPACE 1                                                           SSA02130
        &$XXTRN   BKWSGT                                                  SSA02140
SSSGPSTOP                       EQU       BKWSGT                          SSA02150
        SPACE 1                                                           SSA02160
BKWSGT_PLIST                    DSECT                                     SSA02170
BKWSGT_PLIST_RC                 DS        A   * return code               SSA02180
BKWSGT_PLIST_RE                 DS        A   * reason code               SSA02190
BKWSGT_PLIST_SGN                DS        A   * sgp number                SSA02200
```

```
BKWSGT_PLIST_AC                 DS      A   * attribute count       SSA02210
BKWSGT_PLIST_AA                 DS      A   * attribute array       SSA02220
BKWSGT_PLIST_LENGTH             EQU     *-BKWSGT_PLIST              SSA02230
                                VMASMMAX                           SSA02240
        SPACE 1                                                    SSA02250
*                                                                  SSA02260
*     storage group write                                         SSA02270
*                                                                  SSA02280
        SPACE 1                                                    SSA02290
        &$XXTRN   BKWSGW                                           SSA02300
SSSGPWRITE                      EQU     BKWSGW                     SSA02310
        SPACE 1                                                    SSA02320
BKWSGW_PLIST                    DSECT                              SSA02330
BKWSGW_PLIST_RC                 DS      A   * return code          SSA02340
BKWSGW_PLIST_RE                 DS      A   * reason code          SSA02350
BKWSGW_PLIST_SGN                DS      A   * sgp number           SSA02360
BKWSGW_PLIST_PN                 DS      A   * page number          SSA02370
BKWSGW_PLIST_PC                 DS      A   * page count           SSA02380
BKWSGW_PLIST_ALET               DS      A   * buffer ALET          SSA02390
BKWSGW_PLIST_BUF                DS      A   * buffer               SSA02400
BKWSGW_PLIST_LENGTH             EQU     *-BKWSGW_PLIST             SSA02410
                                VMASMMAX                           SSA02420
        SPACE 1                                                    SSA02430
*----------------------------------------------------------------* SSA02440
*     End of storage group declarations                         * SSA02450
*----------------------------------------------------------------* SSA02460
        EJECT                                                      SSA02470
        POP   PRINT                                                SSA02480
        MEND                                                       SSA02490
```

Services Bindings (SSASMSRV MACRO)

```
        MACRO                                                      SSA00010
        SSASMSRV &WEAK=                                   SSA00020
        AGO   .@ASMSR1                                             SSA00030
.*  Branch around prolog so it is not included in listings    *   SSA00040
.****************************************************************** SSA00050
.*                                                            *   SSA00060
.*  NAME    -  Reusable Server Kernel   services bindings     *   SSA00070
.*                                                            *   SSA00080
.*  FUNCTION  -  Defines the services constants and dsects    *   SSA00090
.*                                                            *   SSA00100
.*  COPYRIGHT -                                        @VR2OZOZ SSA00110
.*                                                     @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992     @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM           @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083          @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                            @VR2OZOZ SSA00160
.*                                                            *   SSA00170
.*  STATUS - Version 2 Release 4                      @VR2OZOZ SSA00180
.*                                                            *   SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4    *   SSA00200
.****************************************************************** SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4    @VR24PVM SSA00220
.****************************************************************** SSA00230
.@ASMSR1 ANOP                                                      SSA00240
        PUSH  PRINT                                                SSA00250
        AIF   ('&SYSPARM' NE 'SUP').ASMSR2            SSA00260
        PRINT OFF,NOGEN                                            SSA00270
.ASMSR2  ANOP                                                      SSA00280
        LCLC  &$XXTRN                                              SSA00290
&$XXTRN  SETC  'EXTRN'                                             SSA00300
        AIF   ('&WEAK' NE 'YES').ASMSR3               SSA00310
&$XXTRN  SETC  'WXTRN'                                             SSA00320
.ASMSR3  ANOP                                                      SSA00330
*----------------------------------------------------------------* SSA00340
*     Return and reason codes for services functions           * SSA00350
*----------------------------------------------------------------* SSA00360
        SPACE 1                                                    SSA00370
* return codes                                                     SSA00380
SS_SRV_RC_SUCCESS               EQU     0                          SSA00390
SS_SRV_RC_WARNING               EQU     4                          SSA00400
SS_SRV_RC_ERROR                 EQU     8                          SSA00410
SS_SRV_RC_ABEND                 EQU     12                         SSA00420
*                                                                  SSA00430
* reason codes                                                     SSA00440
SS_SRV_RE_SUCCESS               EQU     0                          SSA00450
SS_SRV_RE_BAD_TYPE              EQU     701                        SSA00460
SS_SRV_RE_NOT_FOUND             EQU     702                        SSA00470
```

```
SS_SRV_RE_OUT_OF_RANGE              EQU       703                          SSA00480
SS_SRV_RE_OUT_OF_STORAGE            EQU       706                          SSA00490
SS_SRV_RE_EXISTS                    EQU       709                          SSA00500
*                                                                          SSA00510
* types of messages                                                       SSA00520
SS_SRV_MSGTYPE_INSTANCE             EQU       0                            SSA00530
SS_SRV_MSGTYPE_LINEDRIVER           EQU       1                            SSA00540
*                                                                          SSA00550
* types of services                                                       SSA00560
SS_SRV_SRVTYPE_NORMAL               EQU       0                            SSA00570
SS_SRV_SRVTYPE_LD                   EQU       1                            SSA00580
SS_SRV_SRVTYPE_LDSS                 EQU       2                            SSA00590
*                                                                          SSA00600
* values of various msg bits... these have to line                        SSA00610
* up with the message structures below... be careful                      SSA00620
SS_SRV_IBIT_CCLOSE                  EQU       32768                        SSA00630
SS_SRV_IBIT_ACLOSE                  EQU       16384                        SSA00640
SS_SRV_IBIT_CDONE                   EQU       8192                         SSA00650
SS_SRV_IBIT_LDSTOP                  EQU       4096                         SSA00660
SS_SRV_IBIT_NEWDATA                 EQU       2048                         SSA00670
SS_SRV_LBIT_STOPACK                 EQU       32768                        SSA00680
SS_SRV_LBIT_NEWDATA                 EQU       16384                        SSA00690
*                                                                          SSA00700
* length of keys                                                          SSA00710
SS_SRV_KEYLENGTH                    EQU       32                           SSA00720
        SPACE 1                                                           SSA00730
*------------------------------------------------------------------*     SSA00740
*    Stuctures                                                     *     SSA00750
*------------------------------------------------------------------*     SSA00760
        SPACE 1                                                           SSA00770
*                                                                          SSA00780
*  S-block                                                                 SSA00790
*                                                                          SSA00800
VMSS_SBLOCK                         DSECT                                  SSA00810
SBL_NEXT                            DS        A   * next service           SSA00820
SBL_PREV                            DS        A   * prev service           SSA00830
SBL_SN                              DS        CL8 * its name               SSA00840
SBL_SNL                             DS        F   * name length           SSA00850
SBL_INITADDR                        DS        A   * init addr             SSA00860
SBL_AGTADDR                         DS        A   * agent addr            SSA00870
SBL_CMPLADDR                        DS        A   * cmpltn addr           SSA00880
SBL_TYPE                            DS        F   * service type          SSA00890
SBL_LOCKWORD                        DS        F   * lock word             SSA00900
SBL_STARTCOUNT                      DS        F   * start count           SSA00910
SBL_MONPTR                          DS        F   * MON BUF PTR           SSA00920
VMSS_SBLOCK_LEN                     EQU       *-VMSS_SBLOCK                SSA00930
        SPACE 1                                                           SSA00940
*                                                                          SSA00950
*  C-block                                                                 SSA00960
*                                                                          SSA00970
VMSS_CBLOCK                         DSECT                                  SSA00980
VC_SBLOCK                           DS        A                            SSA00990
VC_LDNAME                           DS        CL8                          SSA01000
VC_STATBITS                         DS        XL4                          SSA01010
                                    ORG       VC_STATBITS                  SSA01020
                                    DS        XL1                          SSA01030
VC_B_RECORD                         EQU       X'80'                        SSA01040
                                    DS        XL3                          SSA01050
VC_QH                               DS        F                            SSA01060
VC_SID                              DS        F                            SSA01070
VC_INSTANCE                         DS        F                            SSA01080
VC_THREADID                         DS        F                            SSA01090
VC_IKEY                             DS        CL32                         SSA01100
VC_LKEY                             DS        CL32                         SSA01110
VC_USERID                           DS        CL64                         SSA01120
VC_BYTESIN                          DS        F                            SSA01130
VC_BYTESOUT                         DS        F                            SSA01140
VC_IBW                              DS        F                            SSA01150
VC_LDBW                             DS        F                            SSA01160
VC_STARTSTCK                        DS        CL8                          SSA01170
VC_STOPSTCK                         DS        CL8                          SSA01180
VC_RESERVED                         DS        CL128                        SSA01190
VC_LDDATA                           DS        0C                           SSA01200
VMSS_CBLOCK_LEN                     EQU       *-VMSS_CBLOCK                SSA01210
        SPACE 1                                                           SSA01220
*                                                                          SSA01230
*  msg to instance                                                         SSA01240
*                                                                          SSA01250
VMSS_IMSG                           DSECT                                  SSA01260
VI_IKEY                             DS        CL32                         SSA01270
VI_TYPE                             DS        F                            SSA01280
VI_CBITS                            DS        XL2                          SSA01290
```

```
                                    ORG      VI_CBITS                       SSA01300
                                    DS       XL1                            SSA01310
VI_B_CCLOSE                         EQU      X'80'                          SSA01320
VI_B_ACLOSE                         EQU      X'40'                          SSA01330
VI_B_CDONE                          EQU      X'20'                          SSA01340
VI_B_LDSTOP                         EQU      X'10'                          SSA01350
VI_B_NEWDATA                        EQU      X'08'                          SSA01360
                                    DS       XL1                            SSA01370
VMSS_IMSG_LEN                       EQU      *-VMSS_IMSG                    SSA01380
        SPACE 1                                                            SSA01390
*                                                                          SSA01400
*   msg to line driver                                                     SSA01410
*                                                                          SSA01420
VMSS_LMSG                           DSECT                                  SSA01430
VL_LKEY                             DS       CL32                           SSA01440
VL_TYPE                             DS       F                              SSA01450
VL_IKEY                             DS       CL32                           SSA01460
VL_IBITS                            DS       XL2                            SSA01470
                                    ORG      VL_IBITS                       SSA01480
                                    DS       XL1                            SSA01490
VL_B_STOPACK                        EQU      X'80'                          SSA01500
VL_B_NEWDATA                        EQU      X'40'                          SSA01510
                                    DS       XL1                            SSA01520
VMSS_LMSG_LEN                       EQU      *-VMSS_LMSG                    SSA01530
        SPACE 1                                                            SSA01540
*-----------------------------------------------------------------*       SSA01550
*      Definitions for services function                          *       SSA01560
*-----------------------------------------------------------------*       SSA01570
        SPACE 1                                                            SSA01580
*                                                                          SSA01590
*      bind service to addresses                                           SSA01600
*                                                                          SSA01610
        SPACE 1                                                            SSA01620
        &$XXTRN  BKWVBN                                                    SSA01630
SSSERVICEBIND                       EQU      BKWVBN                         SSA01640
        SPACE 1                                                            SSA01650
BKWVBN_PLIST                        DSECT                                  SSA01660
BKWVBN_PLIST_RC                     DS       A   * return code             SSA01670
BKWVBN_PLIST_RE                     DS       A   * reason code             SSA01680
BKWVBN_PLIST_SN                     DS       A   * service name            SSA01690
BKWVBN_PLIST_SNL                    DS       A   * its length              SSA01700
BKWVBN_PLIST_IA                     DS       A   * init addr               SSA01710
BKWVBN_PLIST_SA                     DS       A   * service addr            SSA01720
BKWVBN_PLIST_TA                     DS       A   * completion addr         SSA01730
BKWVBN_PLIST_ST                     DS       A   * service type            SSA01740
BKWVBN_PLIST_LENGTH                 EQU      *-BKWVBN_PLIST                 SSA01750
                                    VMASMMAX                               SSA01760
        SPACE 1                                                            SSA01770
*                                                                          SSA01780
*     find service block                                                   SSA01790
*                                                                          SSA01800
        SPACE 1                                                            SSA01810
        &$XXTRN  BKWVFN                                                    SSA01820
SSSERVICEFIND                       EQU      BKWVFN                         SSA01830
        SPACE 1                                                            SSA01840
BKWVFN_PLIST                        DSECT                                  SSA01850
BKWVFN_PLIST_RC                     DS       A   * return code             SSA01860
BKWVFN_PLIST_RE                     DS       A   * reason code             SSA01870
BKWVFN_PLIST_SN                     DS       A   * service name            SSA01880
BKWVFN_PLIST_SNL                    DS       A   * its length              SSA01890
BKWVFN_PLIST_SBLK                   DS       A   * S-blk address           SSA01900
BKWVFN_PLIST_LENGTH                 EQU      *-BKWVFN_PLIST                 SSA01910
                                    VMASMMAX                               SSA01920
        SPACE 1                                                            SSA01930
*                                                                          SSA01940
*     start the server                                                     SSA01950
*                                                                          SSA01960
        SPACE 1                                                            SSA01970
        &$XXTRN  BKWVRN                                                    SSA01980
SSSERVERRUN                         EQU      BKWVRN                         SSA01990
        SPACE 1                                                            SSA02000
BKWVRN_PLIST                        DSECT                                  SSA02010
BKWVRN_PLIST_RC                     DS       A   * return code             SSA02020
BKWVRN_PLIST_RE                     DS       A   * reason code             SSA02030
BKWVRN_PLIST_EPLIST                 DS       A   * ADDR OF EPLIST          SSA02040
BKWVRN_PLIST_LENGTH                 EQU      *-BKWVRN_PLIST                 SSA02050
                                    VMASMMAX                               SSA02060
        SPACE 1                                                            SSA02070
*                                                                          SSA02080
*     stop the server                                                      SSA02090
*                                                                          SSA02100
        SPACE 1                                                            SSA02110
```

```
          &$XXTRN  BKWVSP                                              SSA02120
SSSERVERSTOP                     EQU       BKWVSP                      SSA02130
          SPACE 1                                                      SSA02140
BKWVSP_PLIST                     DSECT                                 SSA02150
BKWVSP_PLIST_RC                  DS        A  * return code            SSA02160
BKWVSP_PLIST_RE                  DS        A  * reason code            SSA02170
BKWVSP_PLIST_LENGTH              EQU       *-BKWVSP_PLIST              SSA02180
                                 VMASMMAX                             SSA02190
          SPACE 1                                                      SSA02200
*---------------------------------------------------------------*     SSA02210
*     End of declarations                                       *     SSA02220
*---------------------------------------------------------------*     SSA02230
          EJECT                                                        SSA02240
          POP   PRINT                                                  SSA02250
          MEND                                                         SSA02260
```

Trie Bindings (SSASMTRI MACRO)

```
          MACRO                                                        SSA00010
          SSASMTRI &WEAK=                                    SSA00020
          AGO   .@ASMSR1                                               SSA00030
.*  Branch around prolog so it is not included in listings      *     SSA00040
.**********************************************************************  SSA00050
.*                                                              *     SSA00060
.*  NAME      -  Reusable Server Kernel   services bindings     *     SSA00070
.*                                                              *     SSA00080
.*  FUNCTION  -  Language bindings for trie API                 *     SSA00090
.*                                                              *     SSA00100
.*                                                              *     SSA00110
.*  COPYRIGHT -                                                 *     SSA00120
.*                                                              *     SSA00130
.*     THIS MODULE IS "RESTRICTED MATERIALS OF IBM"             *     SSA00140
.*     5654-030 (C) COPYRIGHT IBM CORP. - 1998, 1999            *     SSA00150
.*     LICENSED MATERIALS - PROPERTY OF IBM                     *     SSA00160
.*     ALL RIGHTS RESERVED.                                     *     SSA00170
.*                                                              *     SSA00180
.*  STATUS - VM/ESA Version 2, Release 4.0                      *     SSA00190
.*                                                              *     SSA00200
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4.0     *     SSA00210
.**********************************************************************  SSA00220
.* A000000-999999   New for VM/ESA Version 2 Release 4.0     @VR74PVM SSA00230
.**********************************************************************  SSA00240
.@ASMSR1 ANOP                                                          SSA00250
          PUSH  PRINT                                                  SSA00260
          AIF   ('&SYSPARM' NE 'SUP').ASMSR2                 SSA00270
          PRINT OFF,NOGEN                                              SSA00280
.ASMSR2  ANOP                                                          SSA00290
          LCLC  &$XXTRN                                                SSA00300
&$XXTRN  SETC  'EXTRN'                                                 SSA00310
          AIF   ('&WEAK' NE 'YES').ASMSR3                    SSA00320
&$XXTRN  SETC  'WXTRN'                                                 SSA00330
.ASMSR3  ANOP                                                          SSA00340
*---------------------------------------------------------------*     SSA00350
*     return and reason codes, and other constants             *     SSA00360
*---------------------------------------------------------------*     SSA00370
          SPACE 1                                                      SSA00380
*                                                                      SSA00390
* return codes                                                         SSA00400
SS_TRI_RC_SUCCESS                EQU       0                           SSA00410
SS_TRI_RC_WARNING                EQU       4                           SSA00420
SS_TRI_RC_ERROR                  EQU       8                           SSA00430
SS_TRI_RC_ABEND                  EQU       12                          SSA00440
*                                                                      SSA00450
* reason codes                                                         SSA00460
SS_TRI_RE_SUCCESS                EQU       0                           SSA00470
SS_TRI_RE_BAD_SIZE               EQU       1701                        SSA00480
SS_TRI_RE_TRIE_EXISTS            EQU       1702                        SSA00490
SS_TRI_RE_OUT_OF_STORAGE         EQU       1703                        SSA00500
SS_TRI_RE_DSCR_FAIL              EQU       1704                        SSA00510
SS_TRI_RE_TRIE_NOT_FOUND         EQU       1705                        SSA00520
SS_TRI_RE_TRIE_BUSY              EQU       1706                        SSA00530
SS_TRI_RE_BAD_INDEX_LEN          EQU       1707                        SSA00540
SS_TRI_RE_BAD_CAPACITY           EQU       1708                        SSA00550
SS_TRI_RE_OUT_OF_DS_STORAGE      EQU       1709                        SSA00560
*                                                                      SSA00570
          SPACE 1                                                      SSA00580
*---------------------------------------------------------------*     SSA00590
*     entry point definitions                                  *     SSA00600
*---------------------------------------------------------------*     SSA00610
```

```
        SPACE 1                                                         SSA00620
*                                                                       SSA00630
*     routine to create a trie                                          SSA00640
*                                                                       SSA00650
        SPACE 1                                                         SSA00660
        &$XXTRN  BKWYCR                                                 SSA00670
SSTRIECREATE                    EQU       BKWYCR                        SSA00680
        SPACE 1                                                         SSA00690
BKWYCR_PLIST                    DSECT                                   SSA00700
BKWYCR_PLIST_RC                 DS        A   * return code             SSA00710
BKWYCR_PLIST_RE                 DS        A   * reason code             SSA00720
BKWYCR_PLIST_NAME               DS        A   * trie name               SSA00730
BKWYCR_PLIST_DS_SIZE            DS        A   * DS size                 SSA00740
BKWYCR_PLIST_ASIT               DS        A   * DS ASIT                 SSA00750
BKWYCR_PLIST_ALET               DS        A   * DS ALET                 SSA00760
BKWYCR_PLIST_LENGTH             EQU       *-BKWYCR_PLIST                SSA00770
                                VMASMMAX                                SSA00780
        SPACE 1                                                         SSA00790
*                                                                       SSA00800
*     routine to delete a trie                                          SSA00810
*                                                                       SSA00820
        SPACE 1                                                         SSA00830
        &$XXTRN  BKWYDE                                                 SSA00840
SSTRIEDELETE                    EQU       BKWYDE                        SSA00850
        SPACE 1                                                         SSA00860
BKWYDE_PLIST                    DSECT                                   SSA00870
BKWYDE_PLIST_RC                 DS        A   * return code             SSA00880
BKWYDE_PLIST_RE                 DS        A   * reason code             SSA00890
BKWYDE_PLIST_NAME               DS        A   * trie name               SSA00900
BKWYDE_PLIST_LENGTH             EQU       *-BKWYDE_PLIST                SSA00910
        SPACE 1                                                         SSA00920
*                                                                       SSA00930
*     routine to insert a record number                                SSA00940
*                                                                       SSA00950
        SPACE 1                                                         SSA00960
        &$XXTRN  BKWYRI                                                 SSA00970
SSTRIERECORDINSERT              EQU       BKWYRI                        SSA00980
        SPACE 1                                                         SSA00990
BKWYRI_PLIST                    DSECT                                   SSA01000
BKWYRI_PLIST_RC                 DS        A   * return code             SSA01010
BKWYRI_PLIST_RE                 DS        A   * reason code             SSA01020
BKWYRI_PLIST_NAME               DS        A   * trie name               SSA01030
BKWYRI_PLIST_ALET               DS        A   * DS ALET                 SSA01040
BKWYRI_PLIST_RECNUM             DS        A   * record number           SSA01050
BKWYRI_PLIST_IX_BUFFER          DS        A   * index buffer            SSA01060
BKWYRI_PLIST_IX_LENGTH          DS        A   * index length            SSA01070
BKWYRI_PLIST_LENGTH             EQU       *-BKWYRI_PLIST                SSA01080
                                VMASMMAX                                SSA01090
        SPACE 1                                                         SSA01100
*                                                                       SSA01110
*     routine to list all record numbers matching proposed key         SSA01120
*                                                                       SSA01130
        SPACE 1                                                         SSA01140
        &$XXTRN  BKWYRL                                                 SSA01150
SSTRIERECORDLIST                EQU       BKWYRL                        SSA01160
        SPACE 1                                                         SSA01170
BKWYRL_PLIST                    DSECT                                   SSA01180
BKWYRL_PLIST_RC                 DS        A   * return code             SSA01190
BKWYRL_PLIST_RE                 DS        A   * reason code             SSA01200
BKWYRL_PLIST_NAME               DS        A   * trie name               SSA01210
BKWYRL_PLIST_ALET               DS        A   * DS ALET                 SSA01220
BKWYRL_PLIST_IX_BUFFER          DS        A   * index buffer            SSA01230
BKWYRL_PLIST_IX_LENGTH          DS        A   * index length            SSA01240
BKWYRL_PLIST_RECNUM_ARRAY       DS        A   * recnum array            SSA01250
BKWYRL_PLIST_RECNUM_ARRAY_CAP   DS        A   * array capacity          SSA01260
BKWYRL_PLIST_RECNUMS_FOUND      DS        A   * recnums found           SSA01270
BKWYRL_PLIST_LENGTH             EQU       *-BKWYRL_PLIST                SSA01280
                                VMASMMAX                                SSA01290
        SPACE 1                                                         SSA01300
*-------------------------------------------------------------------*  SSA01310
*     End of declarations                                           *  SSA01320
*-------------------------------------------------------------------*  SSA01330
        EJECT                                                           SSA01340
        POP   PRINT                                                     SSA01350
        MEND                                                            SSA01360
```

User ID Bindings (SSASMUID MACRO)

```
         MACRO                                                   SSA00010
         SSASMUID &WEAK=                                SSA00020
         AGO   .@ASMSR1                                          SSA00030
.*  Branch around prolog so it is not included in listings      *    SSA00040
.*********************************************************************   SSA00050
.*                                                        *    SSA00060
.*  NAME     -  Reusable Server Kernel   services bindings      *    SSA00070
.*                                                        *    SSA00080
.*  FUNCTION  -  Language bindings for userid service      *    SSA00090
.*                                                        *    SSA00100
.*  COPYRIGHT -                                    @VR2OZOZ SSA00110
.*                                                @VR2OZOZ SSA00120
.*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992      @VR2OZOZ SSA00130
.*     LICENSED MATERIALS - PROPERTY OF IBM            @VR2OZOZ SSA00140
.*     SEE COPYRIGHT INSTRUCTIONS, G120-2083           @VR2OZOZ SSA00150
.*     ALL RIGHTS RESERVED                             @VR2OZOZ SSA00160
.*                                                        *    SSA00170
.*  STATUS - Version 2 Release 4                    @VR2OZOZ SSA00180
.*                                                        *    SSA00190
.*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4     *    SSA00200
.*********************************************************************   SSA00210
.* A000000-999999   New for VM/ESA Version 2 Release 4      @VR24PVM SSA00220
.*********************************************************************   SSA00230
.@ASMSR1 ANOP                                                    SSA00240
         PUSH  PRINT                                             SSA00250
         AIF   ('&SYSPARM' NE 'SUP').ASMSR2              SSA00260
         PRINT OFF,NOGEN                                         SSA00270
.ASMSR2  ANOP                                                    SSA00280
         LCLC  &$XXTRN                                           SSA00290
&$XXTRN  SETC  'EXTRN'                                           SSA00300
         AIF   ('&WEAK' NE 'YES').ASMSR3               SSA00310
&$XXTRN  SETC  'WXTRN'                                           SSA00320
.ASMSR3  ANOP                                                    SSA00330
*---------------------------------------------------------------*    SSA00340
*     return and reason codes for userid service          *    SSA00350
*---------------------------------------------------------------*    SSA00360
         SPACE 1                                                 SSA00370
*                                                                SSA00380
* return codes                                                   SSA00390
SS_UID_RC_SUCCESS              EQU     0                         SSA00400
SS_UID_RC_WARNING             EQU     4                         SSA00410
SS_UID_RC_ERROR               EQU     8                         SSA00420
SS_UID_RC_ABEND              EQU     12                        SSA00430
*                                                                SSA00440
* reason codes                                                   SSA00450
SS_UID_RE_SUCCESS              EQU     0                         SSA00460
SS_UID_RE_NOT_FOUND          EQU     101                       SSA00470
*                                                                SSA00480
* config constants                                              SSA00490
SS_UID_INDEX_WIDTH            EQU     64                        SSA00500
         SPACE 1                                                 SSA00510
*---------------------------------------------------------------*    SSA00520
*     definitions for userid service                      *    SSA00530
*---------------------------------------------------------------*    SSA00540
         SPACE 1                                                 SSA00550
*                                                                SSA00560
*     routine to map user IDs                                    SSA00570
*                                                                SSA00580
         SPACE 1                                                 SSA00590
         &$XXTRN  BKWBMU                                         SSA00600
SSUSERIDMAP                   EQU     BKWBMU                    SSA00610
         SPACE 1                                                 SSA00620
BKWBMU_PLIST                 DSECT                              SSA00630
BKWBMU_PLIST_RC              DS      A  * return code           SSA00640
BKWBMU_PLIST_RE              DS      A  * reason code           SSA00650
BKWBMU_PLIST_IC              DS      A  * input conn            SSA00660
BKWBMU_PLIST_ICL             DS      A  * its length           SSA00670
BKWBMU_PLIST_IN              DS      A  * input node            SSA00680
BKWBMU_PLIST_INL             DS      A  * its length           SSA00690
BKWBMU_PLIST_IU              DS      A  * input user            SSA00700
BKWBMU_PLIST_IUL             DS      A  * its length           SSA00710
BKWBMU_PLIST_OU              DS      A  * output user           SSA00720
BKWBMU_PLIST_OUL             DS      A  * its length           SSA00730
BKWBMU_PLIST_LENGTH          EQU     *-BKWBMU_PLIST            SSA00740
                             VMASMMAX                          SSA00750
         SPACE 1                                                 SSA00760
*---------------------------------------------------------------*    SSA00770
*     End of declarations                                 *    SSA00780
```

```
*-------------------------------------------------------------------*    SSA00790
       EJECT                                                              SSA00800
       POP   PRINT                                                        SSA00810
       MEND                                                               SSA00820
```

Worker Bindings (SSASMWRK MACRO)

```
       MACRO                                                              SSA00010
       SSASMWRK &WEAK=                                              SSA00020
       AGO   .@ASMSR1                                                     SSA00030
.* Branch around prolog so it is not included in listings        *   SSA00040
.*******************************************************************    SSA00050
.*                                                              *   SSA00060
.* NAME     - Reusable Server Kernel   services bindings        *   SSA00070
.*                                                              *   SSA00080
.* FUNCTION  - Language bindings for worker API                 *   SSA00090
.*                                                              *   SSA00100
.*                                                              *   SSA00110
.* COPYRIGHT -                                                  *   SSA00120
.*                                                              *   SSA00130
.*    THIS MODULE IS "RESTRICTED MATERIALS OF IBM"              *   SSA00140
.*    5654-030 (C) COPYRIGHT IBM CORP. - 1998, 1999             *   SSA00150
.*    LICENSED MATERIALS - PROPERTY OF IBM                      *   SSA00160
.*    ALL RIGHTS RESERVED.                                      *   SSA00170
.*                                                              *   SSA00180
.* STATUS - VM/ESA Version 2, Release 4.0                       *   SSA00190
.*                                                              *   SSA00200
.* CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 3.0      *   SSA00210
.*   @SI124VM - alternate userid                                *   SSA00220
.*                                                              *   SSA00230
.*******************************************************************    SSA00240
.* A000000-999999   New for VM/ESA Version 2 Release 3.0      @VR74PVM SSA00250
.*******************************************************************    SSA00260
.@ASMSR1 ANOP                                                            SSA00270
       PUSH  PRINT                                                       SSA00280
       AIF   ('&SYSPARM' NE 'SUP').ASMSR2                          SSA00290
       PRINT OFF,NOGEN                                                   SSA00300
.ASMSR2  ANOP                                                            SSA00310
       LCLC  &$XXTRN                                                     SSA00320
&$XXTRN  SETC  'EXTRN'                                                   SSA00330
       AIF   ('&WEAK' NE 'YES').ASMSR3                            SSA00340
&$XXTRN  SETC  'WXTRN'                                                   SSA00350
.ASMSR3  ANOP                                                            SSA00360
*-------------------------------------------------------------------*    SSA00370
*    return and reason codes for userid service                  *   SSA00380
*-------------------------------------------------------------------*    SSA00390
       SPACE 1                                                           SSA00400
*                                                                        SSA00410
* return codes                                                           SSA00420
SS_WRK_RC_SUCCESS                 EQU    0                               SSA00430
SS_WRK_RC_WARNING                 EQU    4                               SSA00440
SS_WRK_RC_ERROR                   EQU    8                               SSA00450
SS_WRK_RC_ABEND                   EQU    12                              SSA00460
*                                                                        SSA00470
* reason codes                                                           SSA00480
SS_WRK_RE_SUCCESS                 EQU    0                               SSA00490
SS_WRK_RE_OUT_OF_STORAGE          EQU    1601                            SSA00500
SS_WRK_RE_BAD_COUNT               EQU    1602                            SSA00510
SS_WRK_RE_BAD_FLAG_NAME           EQU    1603                            SSA00520
SS_WRK_RE_BAD_FLAG_VALUE          EQU    1604                            SSA00530
SS_WRK_RE_NO_CLASS                EQU    1605                            SSA00540
SS_WRK_RE_NO_SUBORDINATES         EQU    1606                            SSA00550
SS_WRK_RE_ALGTRIES_EXCEEDED       EQU    1607                            SSA00560
SS_WRK_RE_AUTOLOG_FAIL            EQU    1608                            SSA00570
SS_WRK_RE_TIMER_FAIL              EQU    1609                            SSA00580
SS_WRK_RE_IUCVCON_FAIL            EQU    1610                            SSA00590
SS_WRK_RE_FORCE_FAIL              EQU    1611                            SSA00600
SS_WRK_RE_FORCE_TIMEOUT           EQU    1612                            SSA00610
SS_WRK_RE_OPER_DELETE             EQU    1613                            SSA00620
*                                                                        SSA00630
* option flag names                                                      SSA00640
SS_WRK_OFN_PREFER_EMPTY           EQU    0                               SSA00650
SS_WRK_OFN_RETRY_COUNT            EQU    1                               SSA00660
SS_WRK_OFN_ALT_USERID             EQU    2                    @SI124VM SSA00670
SS_WRK_OFN_ALT_SECLABEL           EQU    3                    @SI124VM SSA00680
*                                                                        SSA00690
* option value names                                                     SSA00700
SS_WRK_OFV_NO                     EQU    0                               SSA00710
SS_WRK_OFV_YES                    EQU    1                               SSA00720
```

```
       SPACE 1                                                          SSA00730
*--------------------------------------------------------------------*  SSA00740
*     definitions for worker API                                     *  SSA00750
*--------------------------------------------------------------------*  SSA00760
       SPACE 1                                                          SSA00770
*                                                                       SSA00780
*     routine to allocate a worker connection                           SSA00790
*                                                                       SSA00800
       SPACE 1                                                          SSA00810
       &$XXTRN  BKWCAL                                                   SSA00820
SSWORKERALLOCATE                 EQU      BKWCAL                         SSA00830
       SPACE 1                                                          SSA00840
BKWCAL_PLIST                     DSECT                                  SSA00850
BKWCAL_PLIST_RC                  DS       A  * return code              SSA00860
BKWCAL_PLIST_RE                  DS       A  * reason code              SSA00870
BKWCAL_PLIST_ICBLOCK             DS       A  * instance C-block ptr     SSA00880
BKWCAL_PLIST_CLASSNAME           DS       A  * class name               SSA00890
BKWCAL_PLIST_OCOUNT              DS       A  * option count             SSA00900
BKWCAL_PLIST_ONAMES              DS       A  * option names             SSA00910
BKWCAL_PLIST_OVALUES             DS       A  * option values            SSA00920
BKWCAL_PLIST_WCBLOCK             DS       A  * worker C-block ptr        SSA00930
BKWCAL_PLIST_CONNID              DS       A  * connection ID            SSA00940
BKWCAL_PLIST_LENGTH              EQU      *-BKWCAL_PLIST                SSA00950
                                 VMASMMAX                               SSA00960
       SPACE 1                                                          SSA00970
*--------------------------------------------------------------------*  SSA00980
*     End of declarations                                            *  SSA00990
*--------------------------------------------------------------------*  SSA01000
       EJECT                                                           SSA01010
       POP   PRINT                                                     SSA01020
       MEND                                                            SSA01030
```

PL/X Language Bindings

Anchor Bindings (SSPLXANC COPY)

```
*COPY SSPLXANC                                                          SSP00010
                                                                        SSP00020
 /******************************************************************/   SSP00030
 /*                                                            */        SSP00040
 /*  NAME     -    Reusable Server Kernel   PL/X bindings       */       SSP00050
 /*                                                            */        SSP00060
 /*  FUNCTION  -    Language bindings for anchor services.      */       SSP00070
 /*                                                            */        SSP00080
 /*  COPYRIGHT -                               @VR2OZOZ */                SSP00090
 /*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992   @VR2OZOZ */         SSP00100
 /*     LICENSED MATERIALS - PROPERTY OF IBM        @VR2OZOZ */          SSP00110
 /*     SEE COPYRIGHT INSTRUCTIONS, G120-2083       @VR2OZOZ */          SSP00120
 /*     ALL RIGHTS RESERVED                         @VR2OZOZ */          SSP00130
 /*                                                            */        SSP00140
 /*  STATUS - VM/ESA Version 2 Release 4            @VR2OZOZ */          SSP00150
 /*                                                            */        SSP00160
 /*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4      */       SSP00170
 /******************************************************************/   SSP00180
                                                                        SSP00190
 Declare                                                                SSP00200
                                                                        SSP00210
  /* constants */                                                       SSP00220
                                                                        SSP00230
  /* return codes */                                                    SSP00240
  ss_anc_rc_success         fixed(31) constant(0),                      SSP00250
  ss_anc_rc_warning         fixed(31) constant(4),                      SSP00260
  ss_anc_rc_error           fixed(31) constant(8),                      SSP00270
  ss_anc_rc_abend           fixed(31) constant(12),                     SSP00280
                                                                        SSP00290
  /* reason codes */                                                    SSP00300
  ss_anc_re_success         fixed(31) constant(0),                      SSP00310
                                                                        SSP00320
  /* entry points */                                                    SSP00330
                                                                        SSP00340
  /* set anchor */                                                      SSP00350
  ssAnchorSet   entry                                                   SSP00360
  (                                                                     SSP00370
   fixed(31),               /* return code  */                         SSP00380
   fixed(31),               /* reason code  */                         SSP00390
   pointer(31)              /* anchor value */                         SSP00400
  )                                                                     SSP00410
```

```
  external as ('BKWAST'),                                       SSP00420
                                                                SSP00430
  /* get anchor */                                              SSP00440
  ssAnchorGet  entry                                            SSP00450
  (                                                             SSP00460
   fixed(31),                 /* return code  */                SSP00470
   fixed(31),                 /* reason code  */                SSP00480
   pointer(31),               /* anchor value */                SSP00490
   pointer(31),               /* monitor buf  */                SSP00500
   fixed(31)                  /* monitor len  */                SSP00510
  )                                                             SSP00520
  external as ('BKWAGT');                                       SSP00530
                                                                SSP00540
```

Authorization Bindings (SSPLXAUT COPY)

```
*COPY SSPLXAUT                                                  SSP00010
                                                                SSP00020
/**********************************************************************/ SSP00030
/*                                                            */ SSP00040
/*  NAME     -    Reusable Server Kernel PL/X Bindings        */ SSP00050
/*                                                            */ SSP00060
/*  FUNCTION  -    Language bindings for authorization services.  */ SSP00070
/*                                                            */ SSP00080
/*  COPYRIGHT -                                   @VR2OZOZ */ SSP00090
/*      5684-112 (C) COPYRIGHT IBM CORP.1991, 1992     @VR2OZOZ */ SSP00100
/*      LICENSED MATERIALS - PROPERTY OF IBM           @VR2OZOZ */ SSP00110
/*      SEE COPYRIGHT INSTRUCTIONS, G120-2083          @VR2OZOZ */ SSP00120
/*      ALL RIGHTS RESERVED                            @VR2OZOZ */ SSP00130
/*                                                            */ SSP00140
/*  STATUS - VM/ESA Version 2 Release 4               @VR2OZOZ */ SSP00150
/*                                                            */ SSP00160
/*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4    */ SSP00170
/**********************************************************************/ SSP00180
                                                                SSP00190
/*******************************************************/ SSP00200
/* CONSTANTS                                          */ SSP00210
/*******************************************************/ SSP00220
                                                                SSP00230
Declare                                                         SSP00240
                                                                SSP00250
  /* return codes */                                            SSP00260
  ss_aut_rc_success         fixed(31) constant(0),              SSP00270
  ss_aut_rc_warning         fixed(31) constant(4),              SSP00280
  ss_aut_rc_error           fixed(31) constant(8),              SSP00290
  ss_aut_rc_abend           fixed(31) constant(12),             SSP00300
                                                                SSP00310
  /* reason codes */                                            SSP00320
  ss_aut_re_success         fixed(31) constant(0),              SSP00330
  ss_aut_re_bad_count       fixed(31) constant(300+1),          SSP00340
  ss_aut_re_bad_user_length fixed(31) constant(300+2),          SSP00350
  ss_aut_re_bad_obj_length  fixed(31) constant(300+3),          SSP00360
  ss_aut_re_bad_option      fixed(31) constant(300+4),          SSP00370
  ss_aut_re_bad_qual        fixed(31) constant(300+5),          SSP00380
  ss_aut_re_bad_use         fixed(31) constant(300+6),          SSP00390
  ss_aut_re_exists          fixed(31) constant(300+7),          SSP00400
  ss_aut_re_no_class        fixed(31) constant(300+8),          SSP00410
  ss_aut_re_no_object       fixed(31) constant(300+9),          SSP00420
  ss_aut_re_maq_fail        fixed(31) constant(300+10),         SSP00430
  ss_aut_re_cvw_fail        fixed(31) constant(300+11),         SSP00440
  ss_aut_re_cvs_fail        fixed(31) constant(300+12),         SSP00450
  ss_aut_re_mr_fail         fixed(31) constant(300+13),         SSP00460
  ss_aut_re_too_many        fixed(31) constant(300+14),         SSP00470
  ss_aut_re_out_of_storage  fixed(31) constant(300+15),         SSP00480
  ss_aut_re_no_user         fixed(31) constant(300+16),         SSP00490
  ss_aut_re_prev_io_error   fixed(31) constant(300+17),         SSP00500
  ss_aut_re_prev_sync_error fixed(31) constant(300+18),         SSP00510
  ss_aut_re_read_fail       fixed(31) constant(300+19),         SSP00520
  ss_aut_re_write_fail      fixed(31) constant(300+20),         SSP00530
  ss_aut_re_trunc           fixed(31) constant(300+21),         SSP00540
  ss_aut_re_gwu_fail        fixed(31) constant(300+22),         SSP00550
  ss_aut_re_open_fail       fixed(31) constant(300+23),         SSP00560
  ss_aut_re_bad_cache       fixed(31) constant(300+24),         SSP00570
  ss_aut_re_bad_free        fixed(31) constant(300+25),         SSP00580
  ss_aut_re_bad_op          fixed(31) constant(300+26),         SSP00590
                                                                SSP00600
  /* other constants */                                         SSP00610
                                                                SSP00620
  /* return values from ssAuthTestOperations */                 SSP00630
```

```
/* and ssAuthPermitUser */                                           SSP00640
ss_aut_op_permitted       fixed(31) constant(0),                     SSP00650
ss_aut_op_not_permitted   fixed(31) constant(1),                     SSP00660
ss_aut_op_not_defined     fixed(31) constant(2),                     SSP00670
ss_aut_no_change          fixed(31) constant(3),                     SSP00680
                                                                     SSP00690
/* qualifiers for ssAuthPermitUser */                                SSP00700
ss_aut_add_operation      fixed(31) constant(0),                     SSP00710
ss_aut_remove_operation   fixed(31) constant(1),                     SSP00720
                                                                     SSP00730
/* use arrays in ssAuthPermitUser? */                                SSP00740
ss_aut_use_arrays         fixed(31) constant(0),                     SSP00750
ss_aut_delete_all         fixed(31) constant(1),                     SSP00760
ss_aut_add_all            fixed(31) constant(2),                     SSP00770
                                                                     SSP00780
/* qualifiers for ssAuthDeleteObject */                              SSP00790
ss_aut_rules_only         fixed(31) constant(0),                     SSP00800
ss_aut_rules_and_object   fixed(31) constant(1),                     SSP00810
                                                                     SSP00820
/* qualifiers for ssAuthDeleteUser */                                SSP00830
ss_aut_specific_class     fixed(31) constant(0),                     SSP00840
ss_aut_all_classes        fixed(31) constant(1),                     SSP00850
                                                                     SSP00860
/* qualifiers for ssAuthDeleteClass */                               SSP00870
ss_aut_objects_only       fixed(31) constant(0),                     SSP00880
ss_aut_objects_and_class  fixed(31) constant(1);                     SSP00890
                                                                     SSP00900
/**********************************************************/          SSP00910
/* ENTRY POINTS                                          */          SSP00920
/**********************************************************/          SSP00930
                                                                     SSP00940
Declare                                                              SSP00950
                                                                     SSP00960
 /***********************************************/                    SSP00970
 /* operations on classes                      */                    SSP00980
 /***********************************************/                    SSP00990
                                                                     SSP01000
 /* create class */                                                  SSP01010
 ssAuthCreateClass  entry                                            SSP01020
 (                                                                   SSP01030
  fixed(31),              /* return code       */                    SSP01040
  fixed(31),              /* reason code       */                    SSP01050
  character(8),           /* class identifier  */                    SSP01060
  fixed(31),              /* operation count   */                    SSP01070
  character(4)            /* operation array   */                    SSP01080
 )                                                                   SSP01090
 external as ('BKWUCC'),                                             SSP01100
                                                                     SSP01110
 /* modify class */                                                  SSP01120
 ssAuthModifyClass  entry                                            SSP01130
 (                                                                   SSP01140
  fixed(31),              /* return code       */                    SSP01150
  fixed(31),              /* reason code       */                    SSP01160
  character(8),           /* class identifier  */                    SSP01170
  fixed(31),              /* operation count   */                    SSP01180
  character(4)            /* operation array   */                    SSP01190
 )                                                                   SSP01200
 external as ('BKWUMC'),                                             SSP01210
                                                                     SSP01220
 /* list classes */                                                  SSP01230
 ssAuthListClasses entry                                             SSP01240
 (                                                                   SSP01250
  fixed(31),              /* return code       */                    SSP01260
  fixed(31),              /* reason code       */                    SSP01270
  char(*),                /* match key         */                    SSP01280
  fixed(31),              /* match key length  */                    SSP01290
  fixed(31),              /* number expected   */                    SSP01300
  char(*),                /* output buffer     */                    SSP01310
  fixed(31)               /* number returned   */                    SSP01320
 )                                                                   SSP01330
 external as ('BKWULC'),                                             SSP01340
                                                                     SSP01350
 /* delete class */                                                  SSP01360
 ssAuthDeleteClass entry                                             SSP01370
 (                                                                   SSP01380
  fixed(31),              /* return code       */                    SSP01390
  fixed(31),              /* reason code       */                    SSP01400
  character(8),           /* class identifier  */                    SSP01410
  fixed(31),              /* options count     */                    SSP01420
  fixed(31)               /* options array     */                    SSP01430
 )                                                                   SSP01440
 external as ('BKWUDC'),                                             SSP01450
```

```
/***********************************************/          SSP01460
/* operations on objects                    */             SSP01470
/***********************************************/          SSP01480
                                                           SSP01490
                                                           SSP01500
/* create object */                                        SSP01510
ssAuthCreateObject  entry                                  SSP01520
(                                                          SSP01530
 fixed(31),              /* return code      */            SSP01540
 fixed(31),              /* reason code      */            SSP01550
 character(*),           /* object name      */            SSP01560
 fixed(31),              /* its length       */            SSP01570
 character(8)            /* object class     */            SSP01580
)                                                          SSP01590
external as ('BKWUCO'),                                    SSP01600
                                                           SSP01610
/* list objects in class */                                SSP01620
ssAuthListObjects entry                                    SSP01630
(                                                          SSP01640
 fixed(31),              /* return code      */            SSP01650
 fixed(31),              /* reason code      */            SSP01660
 char(8),                /* class name       */            SSP01670
 char(*),                /* match key        */            SSP01680
 fixed(31),              /* match key length */            SSP01690
 fixed(31),              /* number expected  */            SSP01700
 pointer(31),            /* buffer pointers  */            SSP01710
 fixed(31),              /* buffer sizes     */            SSP01720
 fixed(31),              /* returned lengths */            SSP01730
 fixed(31)               /* number returned  */            SSP01740
)                                                          SSP01750
external as ('BKWULO'),                                    SSP01760
                                                           SSP01770
/* query an object */                                      SSP01780
ssAuthQueryObject entry                                    SSP01790
(                                                          SSP01800
 fixed(31),              /* return code      */            SSP01810
 fixed(31),              /* reason code      */            SSP01820
 character(*),           /* object name      */            SSP01830
 fixed(31),              /* its length       */            SSP01840
 character(8),           /* class name       */            SSP01850
 fixed(31),              /* userids expected */            SSP01860
 pointer(31),            /* userid ptrs      */            SSP01870
 fixed(31),              /* userid buf sizes */            SSP01880
 fixed(31),              /* userid lengths   */            SSP01890
 fixed(31)               /* userids returned */            SSP01900
)                                                          SSP01910
external as ('BKWUQO'),                                    SSP01920
                                                           SSP01930
/* delete object */                                        SSP01940
ssAuthDeleteObject entry                                   SSP01950
(                                                          SSP01960
 fixed(31),              /* return code      */            SSP01970
 fixed(31),              /* reason code      */            SSP01980
 character(*),           /* object name      */            SSP01990
 fixed(31),              /* its length       */            SSP02000
 fixed(31),              /* options count    */            SSP02010
 fixed(31)               /* options array    */            SSP02020
)                                                          SSP02030
external as ('BKWUDO'),                                    SSP02040
                                                           SSP02050
/***********************************************/          SSP02060
/* operations on users                      */             SSP02070
/***********************************************/          SSP02080
                                                           SSP02090
/* permit user */                                          SSP02100
ssAuthPermitUser  entry                                    SSP02110
(                                                          SSP02120
 fixed(31),              /* return code      */            SSP02130
 fixed(31),              /* reason code      */            SSP02140
 character(*),           /* user name        */            SSP02150
 fixed(31),              /* its length       */            SSP02160
 character(*),           /* object name      */            SSP02170
 fixed(31),              /* its length       */            SSP02180
 fixed(31),              /* use arrays?      */            SSP02190
 fixed(31),              /* operation count  */            SSP02200
 character(4),           /* operation array  */            SSP02210
 fixed(31),              /* op qualifiers    */            SSP02220
 fixed(31)               /* op results       */            SSP02230
)                                                          SSP02240
external as ('BKWUPU'),                                    SSP02250
                                                           SSP02260
/* query specific rule */                                  SSP02270
```

```
 ssAuthQueryRule entry                                        SSP02280
 (                                                            SSP02290
  fixed(31),                 /* return code       */          SSP02300
  fixed(31),                 /* reason code       */          SSP02310
  character(*),              /* user name         */          SSP02320
  fixed(31),                 /* its length        */          SSP02330
  character(*),              /* object name       */          SSP02340
  fixed(31),                 /* its length        */          SSP02350
  fixed(31),                 /* ops expected      */          SSP02360
  character(4),              /* operation array   */          SSP02370
  fixed(31)                  /* ops returned      */          SSP02380
 )                                                            SSP02390
 external as ('BKWUQR'),                                      SSP02400
                                                              SSP02410
 /* test operations */                                        SSP02420
 ssAuthTestOperations  entry                                  SSP02430
 (                                                            SSP02440
  fixed(31),                 /* return code       */          SSP02450
  fixed(31),                 /* reason code       */          SSP02460
  character(*),              /* user name         */          SSP02470
  fixed(31),                 /* its length        */          SSP02480
  character(*),              /* object name       */          SSP02490
  fixed(31),                 /* its length        */          SSP02500
  fixed(31),                 /* operation count   */          SSP02510
  character(4),              /* desired ops       */          SSP02520
  fixed(31)                  /* test results      */          SSP02530
 )                                                            SSP02540
 external as ('BKWUTO'),                                      SSP02550
                                                              SSP02560
 /* delete user */                                            SSP02570
 ssAuthDeleteUser entry                                       SSP02580
 (                                                            SSP02590
  fixed(31),                 /* return code       */          SSP02600
  fixed(31),                 /* reason code       */          SSP02610
  character(*),              /* user name         */          SSP02620
  fixed(31),                 /* its length        */          SSP02630
  character(8),              /* class name        */          SSP02640
  fixed(31),                 /* options count     */          SSP02650
  fixed(31)                  /* options array     */          SSP02660
 )                                                            SSP02670
 external as ('BKWUDU'),                                      SSP02680
                                                              SSP02690
 /************************************************/           SSP02700
 /* utility functions                          */            SSP02710
 /************************************************/           SSP02720
                                                              SSP02730
 /* try to reset access to data files */                      SSP02740
 ssAuthReload entry                                           SSP02750
 (                                                            SSP02760
  fixed(31),                 /* return code */                SSP02770
  fixed(31)                  /* reason code */                SSP02780
 )                                                            SSP02790
 external as ('BKWURL');                                      SSP02800
                                                              SSP02810
```

Cache Bindings (SSPLXCAC COPY)

```
*COPY SSPLXCAC                                                SSP00010
                                                              SSP00020
 /*********************************************************************/ SSP00030
 /*                                                      */ SSP00040
 /*  NAME     -    Reusable Server Kernel   PL/X bindings  */ SSP00050
 /*                                                      */ SSP00060
 /*  FUNCTION  -    Language bindings for file cache.      */ SSP00070
 /*                                                      */ SSP00080
 /*  COPYRIGHT -                          @VR2OZOZ */ SSP00090
 /*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992    @VR2OZOZ */ SSP00100
 /*     LICENSED MATERIALS - PROPERTY OF IBM         @VR2OZOZ */ SSP00110
 /*     SEE COPYRIGHT INSTRUCTIONS, G120-2083        @VR2OZOZ */ SSP00120
 /*     ALL RIGHTS RESERVED                          @VR2OZOZ */ SSP00130
 /*                                                      */ SSP00140
 /*  STATUS - VM/ESA Version 2 Release 4             @VR2OZOZ */ SSP00150
 /*                                                      */ SSP00160
 /*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4    */ SSP00170
 /*********************************************************************/ SSP00180
                                                              SSP00190
 /****************************************************/        SSP00200
 /* CONSTANTS                                      */          SSP00210
 /****************************************************/        SSP00220
```

```
                                                                 SSP00230
 Declare                                                         SSP00240
                                                                 SSP00250
  /* return codes */                                             SSP00260
  ss_cac_rc_success         fixed(31) constant(0),               SSP00270
  ss_cac_rc_warning         fixed(31) constant(4),               SSP00280
  ss_cac_rc_error           fixed(31) constant(8),               SSP00290
  ss_cac_rc_abend           fixed(31) constant(12),              SSP00300
                                                                 SSP00310
  /* reason codes */                                             SSP00320
  ss_cac_re_success             fixed(31) constant(0),           SSP00330
  ss_cac_re_out_of_storage      fixed(31) constant(1501),        SSP00340
  ss_cac_re_table_replaced      fixed(31) constant(1502),        SSP00350
  ss_cac_re_cache_not_found     fixed(31) constant(1503),        SSP00360
  ss_cac_re_dscr_fail           fixed(31) constant(1504),        SSP00370
  ss_cac_re_cache_exists        fixed(31) constant(1505),        SSP00380
  ss_cac_re_bad_size            fixed(31) constant(1506),        SSP00390
  ss_cac_re_bad_token           fixed(31) constant(1511),        SSP00400
  ss_cac_re_bad_length          fixed(31) constant(1512),        SSP00410
  ss_cac_re_bad_count           fixed(31) constant(1513),        SSP00420
  ss_cac_re_bad_esmdl           fixed(31) constant(1514),        SSP00430
  ss_cac_re_bad_fname           fixed(31) constant(1515),        SSP00440
  ss_cac_re_bad_fval            fixed(31) constant(1516),        SSP00450
  ss_cac_re_exist_fail          fixed(31) constant(1517),        SSP00460
  ss_cac_re_file_not_found      fixed(31) constant(1518),        SSP00470
  ss_cac_re_delete_in_progress  fixed(31) constant(1519),        SSP00480
  ss_cac_re_bad_offset          fixed(31) constant(1520),        SSP00490
  ss_cac_re_bad_table_id        fixed(31) constant(1521),        SSP00500
  ss_cac_re_table_not_found     fixed(31) constant(1522),        SSP00510
  ss_cac_re_open_fail           fixed(31) constant(1523),        SSP00520
  ss_cac_re_bad_recfm           fixed(31) constant(1524),        SSP00530
  ss_cac_re_bad_lrecl           fixed(31) constant(1525),        SSP00540
  ss_cac_re_out_of_storage_ds   fixed(31) constant(1526),        SSP00550
  ss_cac_re_read_fail           fixed(31) constant(1527),        SSP00560
  ss_cac_re_bad_data_stream     fixed(31) constant(1528),        SSP00570
                                                                 SSP00580
  /* open flag names */                                          SSP00590
  ss_cac_ofn_xlate           fixed(31) constant(0),              SSP00600
  ss_cac_ofn_preserve_dolr   fixed(31) constant(1),              SSP00610
  ss_cac_ofn_bfs             fixed(31) constant(2),              SSP00620
  ss_cac_ofn_recmethod_fs    fixed(31) constant(3),              SSP00630
  ss_cac_ofn_recmethod_cache fixed(31) constant(4),              SSP00640
                                                                 SSP00650
  /* open flag values */                                         SSP00660
  ss_cac_ofv_no              fixed(31) constant(0),              SSP00670
  ss_cac_ofv_yes             fixed(31) constant(1);              SSP00680
                                                                 SSP00690
 /**************************************************/            SSP00700
 /* STRUCTURES                                    */            SSP00710
 /**************************************************/            SSP00720
                                                                 SSP00730
 /**************************************************/            SSP00740
 /* FUNCTIONS                                     */            SSP00750
 /**************************************************/            SSP00760
                                                                 SSP00770
 Declare                                                         SSP00780
                                                                 SSP00790
  /*****************************************/                    SSP00800
  /* cache creation and deletion          */                    SSP00810
  /*****************************************/                    SSP00820
                                                                 SSP00830
  /* create a cache */                                           SSP00840
  ssCacheCreate entry                                            SSP00850
  (                                                              SSP00860
   fixed(31),              /* return code   */                   SSP00870
   fixed(31),              /* reason code   */                   SSP00880
   char(8),                /* cache name    */                   SSP00890
   fixed(31),              /* pages rqstd   */                   SSP00900
   fixed(31)               /* ALET          */                   SSP00910
  )                                                              SSP00920
  external as ('BKWOCC'),                                        SSP00930
                                                                 SSP00940
  /* delete a cache */                                           SSP00950
  ssCacheDelete entry                                            SSP00960
  (                                                              SSP00970
   fixed(31),              /* return code   */                   SSP00980
   fixed(31),              /* reason code   */                   SSP00990
   char(8)                 /* cache name    */                   SSP01000
  )                                                              SSP01010
  external as ('BKWOCD'),                                        SSP01020
                                                                 SSP01030
  /*****************************************/                    SSP01040
```

```
/* utility functions                    */         SSP01050
/*****************************************/         SSP01060
                                                    SSP01070
/* queries cache utilitization */                  SSP01080
ssCacheQuery entry                                 SSP01090
(                                                  SSP01100
 fixed(31),              /* return code   */       SSP01110
 fixed(31),              /* reason code   */       SSP01120
 char(8),                /* cache name    */       SSP01130
 fixed(31),              /* files cached  */       SSP01140
 fixed(31),              /* cache size    */       SSP01150
 fixed(31),              /* amt in use    */       SSP01160
 fixed(31),              /* open count    */       SSP01170
 fixed(31)               /* hit count     */       SSP01180
)                                                  SSP01190
external as ('BKWOCQ'),                            SSP01200
                                                    SSP01210
/* sets translation table */                       SSP01220
ssCacheXlTabSet entry                              SSP01230
(                                                  SSP01240
 fixed(31),              /* return code   */       SSP01250
 fixed(31),              /* reason code   */       SSP01260
 fixed(31),              /* table ID      */       SSP01270
 char(256)               /* table         */       SSP01280
)                                                  SSP01290
external as ('BKWOTS'),                            SSP01300
                                                    SSP01310
/*****************************************/         SSP01320
/* file management primitives            */         SSP01330
/*****************************************/         SSP01340
                                                    SSP01350
/* begin using cached file */                      SSP01360
ssCacheFileOpen entry                              SSP01370
(                                                  SSP01380
 fixed(31),              /* return code     */     SSP01390
 fixed(31),              /* reason code     */     SSP01400
 char(8),                /* cache name      */     SSP01410
 char(*),                /* file spec       */     SSP01420
 fixed(31),              /* its length      */     SSP01430
 char(*),                /* ESM data        */     SSP01440
 fixed(31),              /* its length      */     SSP01450
 fixed(31),              /* flag count      */     SSP01460
 fixed(31),              /* flag name array */     SSP01470
 fixed(31),              /* flag value array */    SSP01480
 char(8),                /* file token      */     SSP01490
 fixed(31),              /* ALET            */     SSP01500
 pointer(31),            /* address         */     SSP01510
 fixed(31),              /* length          */     SSP01520
 char(32)                /* last update date */    SSP01530
)                                                  SSP01540
external as ('BKWOFO'),                            SSP01550
                                                    SSP01560
/* read cached file */                             SSP01570
ssCacheFileRead entry                              SSP01580
(                                                  SSP01590
 fixed(31),              /* return code   */       SSP01600
 fixed(31),              /* reason code   */       SSP01610
 char(8),                /* cache name    */       SSP01620
 char(8),                /* file token    */       SSP01630
 fixed(31),              /* byte offset   */       SSP01640
 fixed(31),              /* num of bytes  */       SSP01650
 char(*),                /* output buffer */       SSP01660
 fixed(31)               /* bytes returned */      SSP01670
)                                                  SSP01680
external as ('BKWOFR'),                            SSP01690
                                                    SSP01700
/* done using cached file */                       SSP01710
ssCacheFileClose entry                             SSP01720
(                                                  SSP01730
 fixed(31),              /* return code     */     SSP01740
 fixed(31),              /* reason code     */     SSP01750
 char(8),                /* cache name      */     SSP01760
 char(8)                 /* file token      */     SSP01770
)                                                  SSP01780
external as ('BKWOFC');                            SSP01790
                                                    SSP01800
```

Client Bindings (SSPLXCLI COPY)

```
*COPY SSPLXCLI                                                      SSP00010
                                                                   SSP00020
/*****************************************************************/ SSP00030
/*                                                             */ SSP00040
/*  NAME      -    Reusable Server Kernel   PL/X bindings       */ SSP00050
/*                                                             */ SSP00060
/*  FUNCTION  -    Language bindings for client services        */ SSP00070
/*                                                             */ SSP00080
/*  COPYRIGHT -                                    @VR2OZOZ */ SSP00090
/*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992   @VR2OZOZ */ SSP00100
/*     LICENSED MATERIALS - PROPERTY OF IBM          @VR2OZOZ */ SSP00110
/*     SEE COPYRIGHT INSTRUCTIONS, G120-2083         @VR2OZOZ */ SSP00120
/*     ALL RIGHTS RESERVED                           @VR2OZOZ */ SSP00130
/*                                                             */ SSP00140
/*  STATUS - VM/ESA Version 2 Release 4              @VR2OZOZ */ SSP00150
/*                                                             */ SSP00160
/*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4       */ SSP00170
/*****************************************************************/ SSP00180
                                                                   SSP00190
/*********************************************************/ SSP00200
/* constants                                           */ SSP00210
/*********************************************************/ SSP00220
                                                                   SSP00230
Declare                                                            SSP00240
                                                                   SSP00250
 /* return codes */                                               SSP00260
 ss_cli_rc_success          fixed(31) constant(0),                SSP00270
 ss_cli_rc_warning          fixed(31) constant(4),                SSP00280
 ss_cli_rc_error            fixed(31) constant(8),                SSP00290
 ss_cli_rc_abend            fixed(31) constant(12),               SSP00300
                                                                   SSP00310
 /* reason codes */                                               SSP00320
 ss_cli_re_success          fixed(31) constant(0),                SSP00330
 ss_cli_re_out_of_range     fixed(31) constant(900+1),            SSP00340
 ss_cli_re_out_of_storage   fixed(31) constant(900+2),            SSP00350
 ss_cli_re_bad_iam          fixed(31) constant(900+3),            SSP00360
 ss_cli_re_bad_method       fixed(31) constant(900+4),            SSP00370
 ss_cli_re_semc_fail        fixed(31) constant(900+5),            SSP00380
                                                                   SSP00390
 /* who i am */                                                   SSP00400
 ss_cli_iam_instance        fixed(31) constant(0),                SSP00410
 ss_cli_iam_linedriver      fixed(31) constant(1),                SSP00420
                                                                   SSP00430
 /* ways to get data */                                           SSP00440
 ss_cli_method_read         fixed(31) constant(0),                SSP00450
 ss_cli_method_peek         fixed(31) constant(1),                SSP00460
 ss_cli_method_discard      fixed(31) constant(2);                SSP00470
                                                                   SSP00480
/*********************************************************/ SSP00490
/* structures                                          */ SSP00500
/*********************************************************/ SSP00510
                                                                   SSP00520
/*********************************************************/ SSP00530
/* entry points                                        */ SSP00540
/*********************************************************/ SSP00550
                                                                   SSP00560
Declare                                                            SSP00570
                                                                   SSP00580
 /* initialize client data queues */                              SSP00590
 ssClientDataInit entry                                           SSP00600
 (                                                                SSP00610
  fixed(31),                 /* return code   */                  SSP00620
  fixed(31),                 /* reason code   */                  SSP00630
  pointer(31),               /* C-block addr  */                  SSP00640
  char(8)                    /* subpool name  */                  SSP00650
 )                                                                SSP00660
 external as ('BKWIIN'),                                          SSP00670
                                                                   SSP00680
 /* terminate client data queues */                               SSP00690
 ssClientDataTerm entry                                           SSP00700
 (                                                                SSP00710
  fixed(31),                 /* return code   */                  SSP00720
  fixed(31),                 /* reason code   */                  SSP00730
  pointer(31)                /* C-block addr  */                  SSP00740
 )                                                                SSP00750
 external as ('BKWITM'),                                          SSP00760
                                                                   SSP00770
 /* get input from client C-block */                              SSP00780
```

```
   ssClientDataGet entry                                     SSP00790
   (                                                          SSP00800
    fixed(31),                    /* return code     */       SSP00810
    fixed(31),                    /* reason code     */       SSP00820
    fixed(31),                    /* instance or ld? */       SSP00830
    pointer(31),                  /* C-block pointer */       SSP00840
    fixed(31),                    /* get method      */       SSP00850
    fixed(31),                    /* ALET to use     */       SSP00860
    char(*),                      /* buffer          */       SSP00870
    fixed(31),                    /* amt wanted      */       SSP00880
    fixed(31),                    /* amt given       */       SSP00890
    fixed(31)                     /* amt left        */       SSP00900
   )                                                          SSP00910
   external as ('BKWIDG'),                                    SSP00920
                                                              SSP00930
   /* put output onto client C-block */                       SSP00940
   ssClientDataPut entry                                      SSP00950
   (                                                          SSP00960
    fixed(31),                    /* return code     */       SSP00970
    fixed(31),                    /* reason code     */       SSP00980
    fixed(31),                    /* instance or ld? */       SSP00990
    pointer(31),                  /* C-block pointer */       SSP01000
    fixed(31),                    /* ALET to use     */       SSP01010
    char(*),                      /* buffer          */       SSP01020
    fixed(31),                    /* amt to put      */       SSP01030
    fixed(31)                     /* new amount      */       SSP01040
   )                                                          SSP01050
   external as ('BKWIDP');                                    SSP01060
                                                              SSP01070
```

Enrollment Bindings (SSPLXENR COPY)

```
*COPY SSPLXENR                                                SSP00010
                                                              SSP00020
 /********************************************************************/ SSP00030
 /*                                                       */  SSP00040
 /*  NAME     -    Reusable Server Kernel   PL/X bindings */  SSP00050
 /*                                                       */  SSP00060
 /*  FUNCTION  -   Language bindings for enrollment services.  */ SSP00070
 /*                                                       */  SSP00080
 /*  COPYRIGHT -                                @VR2OZOZ */   SSP00090
 /*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992   @VR2OZOZ */ SSP00100
 /*     LICENSED MATERIALS - PROPERTY OF IBM      @VR2OZOZ */  SSP00110
 /*     SEE COPYRIGHT INSTRUCTIONS, G120-2083     @VR2OZOZ */  SSP00120
 /*     ALL RIGHTS RESERVED                       @VR2OZOZ */  SSP00130
 /*                                                       */  SSP00140
 /*  STATUS - VM/ESA Version 2 Release 4          @VR2OZOZ */  SSP00150
 /*                                                       */  SSP00160
 /*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4  */ SSP00170
 /********************************************************************/ SSP00180
                                                              SSP00190
 /****************************************************/       SSP00200
 /* CONSTANTS                                    */           SSP00210
 /****************************************************/       SSP00220
                                                              SSP00230
 Declare                                                      SSP00240
                                                              SSP00250
  /* API maxima */                                            SSP00260
  ss_enr_index_width        fixed(31) constant(64),           SSP00270
  ss_enr_max_data           fixed(31) constant(65450),        SSP00280
                                                              SSP00290
  /* return codes */                                          SSP00300
  ss_enr_rc_success         fixed(31) constant(0),            SSP00310
  ss_enr_rc_warning         fixed(31) constant(4),            SSP00320
  ss_enr_rc_error           fixed(31) constant(8),            SSP00330
  ss_enr_rc_abend           fixed(31) constant(12),           SSP00340
                                                              SSP00350
  /* reason codes */                                          SSP00360
  ss_enr_re_success         fixed(31) constant(0),            SSP00370
  ss_enr_re_db_not_found    fixed(31) constant(1000+1),       SSP00380
  ss_enr_re_rec_not_found   fixed(31) constant(1000+2),       SSP00390
  ss_enr_re_truncated       fixed(31) constant(1000+3),       SSP00400
  ss_enr_re_dirty           fixed(31) constant(1000+4),       SSP00410
  ss_enr_re_rec_exists      fixed(31) constant(1000+5),       SSP00420
  ss_enr_re_bad_length      fixed(31) constant(1000+6),       SSP00430
  ss_enr_re_bad_droptype    fixed(31) constant(1000+7),       SSP00440
  ss_enr_re_no_storage      fixed(31) constant(1000+8),       SSP00450
  ss_enr_re_close_fail      fixed(31) constant(1000+9),       SSP00460
  ss_enr_re_write_fail      fixed(31) constant(1000+10),      SSP00470
```

```
 ss_enr_re_bad_method      fixed(31) constant(1000+11),            SSP00480
 ss_enr_re_open_fail       fixed(31) constant(1000+12),            SSP00490
 ss_enr_re_gwu_fail        fixed(31) constant(1000+13),            SSP00500
 ss_enr_re_point_fail      fixed(31) constant(1000+14),            SSP00510
 ss_enr_re_exist_fail      fixed(31) constant(1000+15),            SSP00520
 ss_enr_re_not_sfs         fixed(31) constant(1000+16),            SSP00530
 ss_enr_re_not_v           fixed(31) constant(1000+17),            SSP00540
 ss_enr_re_dscr_fail       fixed(31) constant(1000+18),            SSP00550
 ss_enr_re_read_fail       fixed(31) constant(1000+19),            SSP00560
 ss_enr_re_db_exists       fixed(31) constant(1000+20),            SSP00570
 ss_enr_re_comm_fail       fixed(31) constant(1000+21),            SSP00580
 ss_enr_re_not_disk        fixed(31) constant(1000+22),            SSP00590
 ss_enr_re_bad_kind        fixed(31) constant(1000+23),            SSP00600
 ss_enr_re_new_file        fixed(31) constant(1000+24),            SSP00610
 ss_enr_re_no_sets         fixed(31) constant(1000+25),            SSP00620
 ss_enr_re_set_empty       fixed(31) constant(1000+26),            SSP00630
                                                                   SSP00640
 /* KIND types */                                                  SSP00650
 ss_enr_kind_memory        fixed(31) constant(0),                  SSP00660
 ss_enr_kind_disk          fixed(31) constant(1),                  SSP00670
                                                                   SSP00680
 /* INSERT types */                                                SSP00690
 ss_enr_insert_new         fixed(31) constant(0),                  SSP00700
 ss_enr_insert_replace     fixed(31) constant(1),                  SSP00710
                                                                   SSP00720
 /* DROP types */                                                  SSP00730
 ss_enr_drop_commit        fixed(31) constant(0),                  SSP00740
 ss_enr_drop_rollback      fixed(31) constant(1);                  SSP00750
                                                                   SSP00760
/****************************************************/             SSP00770
/* ENTRY POINTS                                     */             SSP00780
/****************************************************/             SSP00790
                                                                   SSP00800
Declare                                                            SSP00810
                                                                   SSP00820
 /* commit enrollment data base */                                 SSP00830
 ssEnrollCommit entry                                              SSP00840
 (                                                                 SSP00850
  fixed(31),              /* return code   */                      SSP00860
  fixed(31),              /* reason code   */                      SSP00870
  char(8)                 /* dbase name    */                      SSP00880
 )                                                                 SSP00890
 external as ('BKWJCM'),                                           SSP00900
                                                                   SSP00910
 /* drop enrollment data base */                                   SSP00920
 ssEnrollDrop entry                                                SSP00930
 (                                                                 SSP00940
  fixed(31),              /* return code   */                      SSP00950
  fixed(31),              /* reason code   */                      SSP00960
  char(8),                /* dbase name    */                      SSP00970
  fixed(31)               /* drop type     */                      SSP00980
 )                                                                 SSP00990
 external as ('BKWJDP'),                                           SSP01000
                                                                   SSP01010
 /* list data bases */                                             SSP01020
 ssEnrollList entry                                                SSP01030
 (                                                                 SSP01040
  fixed(31),              /* return code   */                      SSP01050
  fixed(31),              /* reason code   */                      SSP01060
  pointer(31)             /* C-block       */                      SSP01070
 )                                                                 SSP01080
 external as ('BKWJDL'),                                           SSP01090
                                                                   SSP01100
 /* load enrollment data base */                                   SSP01110
 ssEnrollLoad entry                                                SSP01120
 (                                                                 SSP01130
  fixed(31),              /* return code  */                       SSP01140
  fixed(31),              /* reason code  */                       SSP01150
  char(8),                /* dbase name   */                       SSP01160
  fixed(31),              /* DS kind      */                       SSP01170
  fixed(31),              /* DS size      */                       SSP01180
  char(*),                /* filename     */                       SSP01190
  fixed(31)               /* length of    */                       SSP01200
 )                                                                 SSP01210
 external as ('BKWJLO'),                                           SSP01220
                                                                   SSP01230
 /* get record */                                                  SSP01240
 ssEnrollRecordGet entry                                           SSP01250
 (                                                                 SSP01260
  fixed(31),                /* return code   */                    SSP01270
  fixed(31),                /* reason code   */                    SSP01280
  char(8),                  /* dbase name    */                    SSP01290
```

```
        char(ss_enr_index_width),    /* index         */         SSP01300
        char(*),                     /* buffer        */         SSP01310
        fixed(31),                   /* buf size      */         SSP01320
        fixed(31)                    /* amt returned  */         SSP01330
    )                                                            SSP01340
    external as ('BKWJRG'),                                      SSP01350
                                                                 SSP01360
    /* insert record */                                          SSP01370
    ssEnrollRecordInsert entry                                   SSP01380
    (                                                            SSP01390
        fixed(31),                   /* return code   */         SSP01400
        fixed(31),                   /* reason code   */         SSP01410
        char(8),                     /* dbase name    */         SSP01420
        char(ss_enr_index_width),    /* index         */         SSP01430
        char(*),                     /* data          */         SSP01440
        fixed(31),                   /* length        */         SSP01450
        fixed(31)                    /* replace?      */         SSP01460
    )                                                            SSP01470
    external as ('BKWJRI'),                                      SSP01480
                                                                 SSP01490
    /* list records */                                           SSP01500
    ssEnrollRecordList entry                                     SSP01510
    (                                                            SSP01520
        fixed(31),                   /* return code   */         SSP01530
        fixed(31),                   /* reason code   */         SSP01540
        char(8),                     /* dbase name    */         SSP01550
        pointer(31)                  /* C-block       */         SSP01560
    )                                                            SSP01570
    external as ('BKWJRL'),                                      SSP01580
                                                                 SSP01590
    /* remove record */                                          SSP01600
    ssEnrollRecordRemove entry                                   SSP01610
    (                                                            SSP01620
        fixed(31),                   /* return code   */         SSP01630
        fixed(31),                   /* reason code   */         SSP01640
        char(8),                     /* dbase name    */         SSP01650
        char(ss_enr_index_width)     /* index         */         SSP01660
    )                                                            SSP01670
    external as ('BKWJRR');                                      SSP01680
                                                                 SSP01690
```

Memory Bindings (SSPLXMEM COPY)

```
*COPY SSPLXMEM                                                    SSP00010
                                                                 SSP00020
/********************************************************************/ SSP00030
/*                                                              */ SSP00040
/*  NAME      -    Reusable Server Kernel   PL/X bindings        */ SSP00050
/*                                                              */ SSP00060
/*  FUNCTION  -    Language bindings for memory services.        */ SSP00070
/*                                                              */ SSP00080
/*  COPYRIGHT -                                       @VR2OZOZ */ SSP00090
/*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992     @VR2OZOZ */ SSP00100
/*     LICENSED MATERIALS - PROPERTY OF IBM           @VR2OZOZ */ SSP00110
/*     SEE COPYRIGHT INSTRUCTIONS, G120-2083          @VR2OZOZ */ SSP00120
/*     ALL RIGHTS RESERVED                            @VR2OZOZ */ SSP00130
/*                                                              */ SSP00140
/*  STATUS - VM/ESA Version 2 Release 4               @VR2OZOZ */ SSP00150
/*                                                              */ SSP00160
/*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4        */ SSP00170
/********************************************************************/ SSP00180
                                                                 SSP00190
Declare                                                          SSP00200
                                                                 SSP00210
    /* return and reason codes */                                SSP00220
    ss_mem_rc_success        fixed(31) constant(0),              SSP00230
    ss_mem_rc_warning        fixed(31) constant(4),              SSP00240
    ss_mem_rc_error          fixed(31) constant(8),              SSP00250
    ss_mem_rc_abend          fixed(31) constant(12),             SSP00260
                                                                 SSP00270
    ss_mem_re_success        fixed(31) constant(0),              SSP00280
    ss_mem_re_out_of_storage fixed(31) constant(800+1),          SSP00290
    ss_mem_re_bad_amount     fixed(31) constant(800+2),          SSP00300
    ss_mem_re_bad_align      fixed(31) constant(800+3),          SSP00310
    ss_mem_re_no_subpool     fixed(31) constant(800+4),          SSP00320
    ss_mem_re_not_alloc      fixed(31) constant(800+5),          SSP00330
    ss_mem_re_subpool_deleted fixed(31) constant(800+6),         SSP00340
    ss_mem_re_spd_fail       fixed(31) constant(800+7),          SSP00350
    ss_mem_re_bad_key        fixed(31) constant(800+8),          SSP00360
```

```
  ss_mem_re_subpool_exists  fixed(31) constant(800+9),         SSP00370
  ss_mem_re_spcc_fail       fixed(31) constant(800+10),        SSP00380
  ss_mem_re_spla_fail       fixed(31) constant(800+11),        SSP00390
                                                               SSP00400
  /* alignment attributes */                                  SSP00410
  ss_mem_align_norm         fixed(31) constant(0),             SSP00420
  ss_mem_align_page         fixed(31) constant(1),             SSP00430
                                                               SSP00440
  /* create a data space we can manage */                     SSP00450
  ssMemoryCreateDS entry                                       SSP00460
  (                                                            SSP00470
   fixed(31),               /* return code   */                SSP00480
   fixed(31),               /* reason code   */                SSP00490
   char(8),                 /* subpool name  */                SSP00500
   fixed(31),               /* size (pages)  */                SSP00510
   fixed(31),               /* storage key   */                SSP00520
   fixed(31),               /* option count  */                SSP00530
   fixed(31),               /* option array  */                SSP00540
   char(8),                 /* ASIT          */                SSP00550
   fixed(31)                /* ALET          */                SSP00560
  )                                                            SSP00570
  external as ('BKWMCR'),                                      SSP00580
                                                               SSP00590
  /* allocate memory */                                       SSP00600
  ssMemoryAllocate entry                                       SSP00610
  (                                                            SSP00620
   fixed(31),               /* return code   */                SSP00630
   fixed(31),               /* reason code   */                SSP00640
   fixed(31),               /* lower bound   */                SSP00650
   fixed(31),               /* upper bound   */                SSP00660
   character(8),            /* subpool name  */                SSP00670
   fixed(31),               /* alignment rqt */                SSP00680
   pointer(31),             /* addr of block */                SSP00690
   fixed(31)                /* amount gotten */                SSP00700
  )                                                            SSP00710
  external as ('BKWMAL'),                                      SSP00720
                                                               SSP00730
  /* release memory */                                        SSP00740
  ssMemoryRelease  entry                                       SSP00750
  (                                                            SSP00760
   fixed(31),               /* return code    */               SSP00770
   fixed(31),               /* reason code    */               SSP00780
   fixed(31),               /* bytes released */               SSP00790
   character(8),            /* subpool name   */               SSP00800
   pointer(31)              /* addr of block  */               SSP00810
  )                                                            SSP00820
  external as ('BKWMRE'),                                      SSP00830
                                                               SSP00840
  /* delete subpool */                                        SSP00850
  ssMemoryDelete  entry                                        SSP00860
  (                                                            SSP00870
   fixed(31),               /* return code    */               SSP00880
   fixed(31),               /* reason code    */               SSP00890
   character(8)             /* subpool name   */               SSP00900
  )                                                            SSP00910
  external as ('BKWMDE');                                      SSP00920
                                                               SSP00930
```

Storage Group Bindings (SSPLXSGP COPY)

```
*COPY SSPLXSGP                                                 SSP00010
                                                               SSP00020
 /*******************************************************************/ SSP00030
 /*                                                            */ SSP00040
 /*  NAME      -    Reusable Server Kernel   PL/X bindings      */ SSP00050
 /*                                                            */ SSP00060
 /*  FUNCTION  -    Language bindings for storage group services. */ SSP00070
 /*                                                            */ SSP00080
 /*  COPYRIGHT -                                     @VR2OZOZ */ SSP00090
 /*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992   @VR2OZOZ */ SSP00100
 /*     LICENSED MATERIALS - PROPERTY OF IBM         @VR2OZOZ */ SSP00110
 /*     SEE COPYRIGHT INSTRUCTIONS, G120-2083        @VR2OZOZ */ SSP00120
 /*     ALL RIGHTS RESERVED                          @VR2OZOZ */ SSP00130
 /*                                                            */ SSP00140
 /*  STATUS - VM/ESA Version 2 Release 4             @VR2OZOZ */ SSP00150
 /*                                                            */ SSP00160
 /*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4     */ SSP00170
 /*******************************************************************/ SSP00180
                                                               SSP00190
```

```
/********************************************************************/   SSP00200
/* CONSTANTS                                                    */   SSP00210
/********************************************************************/   SSP00220
                                                                        SSP00230
Declare                                                                 SSP00240
                                                                        SSP00250
 /* return codes */                                                     SSP00260
 ss_sgp_rc_success        fixed(31) constant(0),                        SSP00270
 ss_sgp_rc_warning        fixed(31) constant(4),                        SSP00280
 ss_sgp_rc_error          fixed(31) constant(8),                        SSP00290
 ss_sgp_rc_abend          fixed(31) constant(12),                       SSP00300
                                                                        SSP00310
 /* reason codes */                                                     SSP00320
 ss_sgp_re_success        fixed(31) constant(0),                        SSP00330
 ss_sgp_re_too_many       fixed(31) constant(600+1),                    SSP00340
 ss_sgp_re_not_found      fixed(31) constant(600+2),                    SSP00350
 ss_sgp_re_out_of_storage fixed(31) constant(600+3),                    SSP00360
 ss_sgp_re_mx_fail        fixed(31) constant(600+4),                    SSP00370
 ss_sgp_re_init_done      fixed(31) constant(600+5),                    SSP00380
 ss_sgp_re_exists         fixed(31) constant(600+7),                    SSP00390
 ss_sgp_re_vdq_fail       fixed(31) constant(600+8),                    SSP00400
 ss_sgp_re_online         fixed(31) constant(600+9),                    SSP00410
 ss_sgp_re_offline        fixed(31) constant(600+10),                   SSP00420
 ss_sgp_re_q_fail         fixed(31) constant(600+11),                   SSP00430
 ss_sgp_re_cv_fail        fixed(31) constant(600+12),                   SSP00440
 ss_sgp_re_e_fail         fixed(31) constant(600+13),                   SSP00450
 ss_sgp_re_maint          fixed(31) constant(600+14),                   SSP00460
 ss_sgp_re_ds_fail        fixed(31) constant(600+15),                   SSP00470
 ss_sgp_re_pool_fail      fixed(31) constant(600+16),                   SSP00480
 ss_sgp_re_map_fail       fixed(31) constant(600+17),                   SSP00490
 ss_sgp_re_bad_attrib     fixed(31) constant(600+18),                   SSP00500
 ss_sgp_re_rewrite_fail   fixed(31) constant(600+19),                   SSP00510
 ss_sgp_re_read_only      fixed(31) constant(600+20),                   SSP00520
 ss_sgp_re_out_of_range   fixed(31) constant(600+22),                   SSP00530
 ss_sgp_re_wrong_mode     fixed(31) constant(600+23),                   SSP00540
 ss_sgp_re_io_fail        fixed(31) constant(600+24),                   SSP00550
 ss_sgp_re_diag_250_fail  fixed(31) constant(600+25),                   SSP00560
 ss_sgp_re_too_big        fixed(31) constant(600+26),                   SSP00570
 ss_sgp_re_bad_name       fixed(31) constant(600+28),                   SSP00580
 ss_sgp_re_name_in_use    fixed(31) constant(600+29),                   SSP00590
                                                                        SSP00600
 /* attributes */                                                       SSP00610
 ss_sgp_attrib_ds         fixed(31) constant(0),                        SSP00620
 ss_sgp_attrib_no_ds      fixed(31) constant(1),                        SSP00630
 ss_sgp_attrib_block_rw   fixed(31) constant(2),                        SSP00640
 ss_sgp_attrib_block_ro   fixed(31) constant(3),                        SSP00650
 ss_sgp_attrib_offline    fixed(31) constant(7);                        SSP00660
                                                                        SSP00670
/********************************************************************/   SSP00680
/* FUNCTIONS                                                    */   SSP00690
/********************************************************************/   SSP00700
                                                                        SSP00710
Declare                                                                 SSP00720
                                                                        SSP00730
 /* storage group create */                                            SSP00740
 ssSgpCreate  entry                                                     SSP00750
 (                                                                      SSP00760
  fixed(31),            /* return code      */                          SSP00770
  fixed(31),            /* reason code      */                          SSP00780
  fixed(31),            /* sg number        */                          SSP00790
  fixed(31),            /* num of vdevs     */                          SSP00800
  fixed(31),            /* vdev array       */                          SSP00810
  fixed(31),            /* attrib count     */                          SSP00820
  fixed(31)             /* attrib array     */                          SSP00830
 )                                                                      SSP00840
 external as ('BKWSGC'),                                                SSP00850
                                                                        SSP00860
 /* storage group delete */                                            SSP00870
 ssSgpDelete  entry                                                     SSP00880
 (                                                                      SSP00890
  fixed(31),            /* return code      */                          SSP00900
  fixed(31),            /* reason code      */                          SSP00910
  fixed(31)             /* sg number        */                          SSP00920
 )                                                                      SSP00930
 external as ('BKWSGD'),                                                SSP00940
                                                                        SSP00950
 /* storage group find */                                              SSP00960
 ssSgpFind  entry                                                       SSP00970
 (                                                                      SSP00980
  fixed(31),            /* return code */                               SSP00990
  fixed(31),            /* reason code */                               SSP01000
  char(8),              /* sg name     */                               SSP01010
```

```
  fixed(31),              /* sgp id      */                          SSP01020
  fixed(31),              /* I/O mode    */                          SSP01030
  fixed(32)               /* total blks  */                          SSP01040
 )                                                                   SSP01050
 external as ('BKWSGF'),                                             SSP01060
                                                                     SSP01070
 /* storage group list (what's defined?) */                         SSP01080
 ssSgpList  entry                                                    SSP01090
 (                                                                   SSP01100
  fixed(31),              /* return code       */                    SSP01110
  fixed(31),              /* reason code       */                    SSP01120
  fixed(31),              /* num expected      */                    SSP01130
  fixed(31),              /* number filled in  */                    SSP01140
  fixed(31)               /* array for IDs     */                    SSP01150
 )                                                                   SSP01160
 external as ('BKWSGL'),                                             SSP01170
                                                                     SSP01180
 /* storage group query (details on particular sg) */               SSP01190
 ssSgpQuery  entry                                                   SSP01200
 (                                                                   SSP01210
  fixed(31),              /* return code       */                    SSP01220
  fixed(31),              /* reason code       */                    SSP01230
  fixed(31),              /* sgp id            */                    SSP01240
  char(8),                /* sg name           */                    SSP01250
  fixed(31),              /* i/o mode          */                    SSP01260
  fixed(32),              /* total blocks      */                    SSP01270
  fixed(31),              /* status word       */                    SSP01280
  fixed(31),              /* attrib expected   */                    SSP01290
  fixed(31),              /* attrib filled in  */                    SSP01300
  fixed(31),              /* attrib array      */                    SSP01310
  fixed(31),              /* vdevs expected    */                    SSP01320
  fixed(31),              /* vdevs filled in   */                    SSP01330
  fixed(31),              /* vdev array        */                    SSP01340
  fixed(31)               /* blks array        */                    SSP01350
 )                                                                   SSP01360
 external as ('BKWSGQ'),                                             SSP01370
                                                                     SSP01380
 /* storage group read */                                           SSP01390
 ssSgpRead  entry                                                    SSP01400
 (                                                                   SSP01410
  fixed(31),                    /* return code */                    SSP01420
  fixed(31),                    /* reason code */                    SSP01430
  fixed(31),                    /* sgp ID      */                    SSP01440
  fixed(32),                    /* page number */                    SSP01450
  fixed(32),                    /* num of pgs  */                    SSP01460
  fixed(31),                    /* buffer ALET */                    SSP01470
  character(*)                  /* buffer      */                    SSP01480
 )                                                                   SSP01490
 external as ('BKWSGR'),                                             SSP01500
                                                                     SSP01510
 /* storage group start (like a mount) */                           SSP01520
 ssSgpStart  entry                                                   SSP01530
 (                                                                   SSP01540
  fixed(31),              /* return code       */                    SSP01550
  fixed(31),              /* reason code       */                    SSP01560
  fixed(31),              /* sgp id            */                    SSP01570
  char(8),                /* sgp name          */                    SSP01580
  fixed(31),              /* attrib count      */                    SSP01590
  fixed(31)               /* attrib array      */                    SSP01600
 )                                                                   SSP01610
 external as ('BKWSGS'),                                             SSP01620
                                                                     SSP01630
 /* storage group stop (like a dismount) */                         SSP01640
 ssSgpStop  entry                                                    SSP01650
 (                                                                   SSP01660
  fixed(31),              /* return code       */                    SSP01670
  fixed(31),              /* reason code       */                    SSP01680
  fixed(31),              /* sgp ID            */                    SSP01690
  fixed(31),              /* attrib count      */                    SSP01700
  fixed(31)               /* attrib array      */                    SSP01710
 )                                                                   SSP01720
 external as ('BKWSGT'),                                             SSP01730
                                                                     SSP01740
 /* storage group write */                                          SSP01750
 ssSgpWrite  entry                                                   SSP01760
 (                                                                   SSP01770
  fixed(31),                    /* return code */                    SSP01780
  fixed(31),                    /* reason code */                    SSP01790
  fixed(31),                    /* sgp ID      */                    SSP01800
  fixed(32),                    /* page number */                    SSP01810
  fixed(32),                    /* num of pgs  */                    SSP01820
  fixed(31),                    /* buffer ALET */                    SSP01830
```

```
  character(*)                   /* buffer     */                 SSP01840
  )                                                               SSP01850
external as ('BKWSGW');                                           SSP01860
                                                                  SSP01870
```

Services Bindings (SSPLXSRV COPY)

```
*COPY SSPLXSRV                                                    SSP00010
                                                                  SSP00020
/******************************************************************/ SSP00030
/*                                                              */ SSP00040
/*  NAME      -    Reusable Server Kernel   PL/X bindings        */ SSP00050
/*                                                              */ SSP00060
/*  FUNCTION  -    Language bindings for service services.       */ SSP00070
/*                                                              */ SSP00080
/*  COPYRIGHT -                                       @VR2OZOZ */ SSP00090
/*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992      @VR2OZOZ */ SSP00100
/*     LICENSED MATERIALS - PROPERTY OF IBM            @VR2OZOZ */ SSP00110
/*     SEE COPYRIGHT INSTRUCTIONS, G120-2083           @VR2OZOZ */ SSP00120
/*     ALL RIGHTS RESERVED                             @VR2OZOZ */ SSP00130
/*                                                              */ SSP00140
/*  STATUS - VM/ESA Version 2 Release 4                @VR2OZOZ */ SSP00150
/*                                                              */ SSP00160
/*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4        */ SSP00170
/******************************************************************/ SSP00180
                                                                  SSP00190
/*************************************************************/ SSP00200
/* constants                                            */ SSP00210
/*************************************************************/ SSP00220
                                                                  SSP00230
Declare                                                           SSP00240
                                                                  SSP00250
 /* return codes */                                               SSP00260
 ss_srv_rc_success             fixed(31) constant(0),             SSP00270
 ss_srv_rc_warning             fixed(31) constant(4),             SSP00280
 ss_srv_rc_error               fixed(31) constant(8),             SSP00290
 ss_srv_rc_abend               fixed(31) constant(12),            SSP00300
                                                                  SSP00310
 /* reason codes */                                               SSP00320
 ss_srv_re_success             fixed(31) constant(0),             SSP00330
 ss_srv_re_bad_type            fixed(31) constant(700+1),         SSP00340
 ss_srv_re_not_found           fixed(31) constant(700+2),         SSP00350
 ss_srv_re_out_of_range        fixed(31) constant(700+3),         SSP00360
 ss_srv_re_out_of_storage      fixed(31) constant(700+6),         SSP00370
 ss_srv_re_exists              fixed(31) constant(700+9),         SSP00380
                                                                  SSP00390
 /* types of messages */                                          SSP00400
 ss_srv_msgtype_instance       fixed(31) constant(0),             SSP00410
 ss_srv_msgtype_linedriver     fixed(31) constant(1),             SSP00420
                                                                  SSP00430
 /* types of services */                                          SSP00440
 ss_srv_srvtype_normal         fixed(31) constant(0),             SSP00450
 ss_srv_srvtype_ld             fixed(31) constant(1),             SSP00460
 ss_srv_srvtype_ldss           fixed(31) constant(2),             SSP00470
                                                                  SSP00480
 /* values of various msg bits... these have to line   */         SSP00490
 /* up with the message structures below... be careful */         SSP00500
 ss_srv_ibit_cclose            fixed(16) constant(32768),         SSP00510
 ss_srv_ibit_aclose            fixed(16) constant(16384),         SSP00520
 ss_srv_ibit_cdone             fixed(16) constant(8192),          SSP00530
 ss_srv_ibit_ldstop            fixed(16) constant(4096),          SSP00540
 ss_srv_ibit_newdata           fixed(16) constant(2048),          SSP00550
                                                                  SSP00560
 ss_srv_lbit_stopack           fixed(16) constant(32768),         SSP00570
 ss_srv_lbit_newdata           fixed(16) constant(16384),         SSP00580
                                                                  SSP00590
 /* length of keys */                                             SSP00600
 ss_srv_keylength              fixed(31) constant(32);            SSP00610
                                                                  SSP00620
/*************************************************************/ SSP00630
/* structures                                           */ SSP00640
/*************************************************************/ SSP00650
                                                                  SSP00660
Declare                                                           SSP00670
                                                                  SSP00680
 /* S-block */                                                    SSP00690
 1 vmss_sblock               Boundary(Word) Based,                SSP00700
   5  sbl_next                pointer(31),        /* next service */ SSP00710
   5  sbl_prev                pointer(31),        /* prev service */ SSP00720
```

```
     5  sbl_sn               character(8),         /* its name      */   SSP00730
     5  sbl_snl              fixed(31),            /* name length   */   SSP00740
     5  sbl_initaddr         pointer(31),          /* init addr     */   SSP00750
     5  sbl_agtaddr          pointer(31),          /* agent addr    */   SSP00760
     5  sbl_cmpladdr         pointer(31),          /* cmpltn addr   */   SSP00770
     5  sbl_type             fixed(31),            /* service type  */   SSP00780
     5  sbl_lockword         fixed(31),            /* lock word     */   SSP00790
     5  sbl_startcount       fixed(31),            /* start count   */   SSP00800
     5  sbl_monptr           fixed(31),            /* mon buf ptr   */   SSP00810
                                                                         SSP00820
  /* C-block */                                                          SSP00830
  1  vmss_cblock             boundary(word) based,                       SSP00840
     5  vc_sblock            pointer(31),                                SSP00850
     5  vc_ldname            character(8),                               SSP00860
     5  vc_statbits          bit(32),                                    SSP00870
     10 vc_b_record          bit(1),                                     SSP00880
     5  vc_qh                fixed(31),                                  SSP00890
     5  vc_sid               fixed(31),                                  SSP00900
     5  vc_instance          fixed(31),                                  SSP00910
     5  vc_threadid          fixed(31),                                  SSP00920
     5  vc_ikey              character(ss_srv_keylength),                SSP00930
     5  vc_lkey              character(ss_srv_keylength),                SSP00940
     5  vc_userid            character(64),                              SSP00950
     5  vc_bytesin           fixed(31),                                  SSP00960
     5  vc_bytesout          fixed(31),                                  SSP00970
     5  vc_ibw               fixed(31),                                  SSP00980
     5  vc_ldbw              fixed(31),                                  SSP00990
     5  vc_startstck         char(8),                                    SSP01000
     5  vc_stopstck          char(8),                                    SSP01010
     5  vc_reserved          char(128),                                  SSP01020
     5  vc_lddata            char(0),                                    SSP01030
                                                                         SSP01040
  /* msg to instance */                                                  SSP01050
  1  vmss_imsg               boundary(word) based,                       SSP01060
     5  vi_ikey              character(ss_srv_keylength),                SSP01070
     5  vi_type              fixed(31),                                  SSP01080
     5  vi_cbits             bit(16),                                    SSP01090
     10  vi_b_cclose         bit(1),                                     SSP01100
     10  vi_b_aclose         bit(1),                                     SSP01110
     10  vi_b_cdone          bit(1),                                     SSP01120
     10  vi_b_ldstop         bit(1),                                     SSP01130
     10  vi_b_newdata        bit(1),                                     SSP01140
                                                                         SSP01150
  /* msg to line driver */                                              SSP01160
  1  vmss_lmsg               boundary(word) based,                       SSP01170
     5  vl_lkey              character(ss_srv_keylength),                SSP01180
     5  vl_type              fixed(31),                                  SSP01190
     5  vl_ikey              character(ss_srv_keylength),                SSP01200
     5  vl_ibits             bit(16),                                    SSP01210
     10  vl_b_stopack        bit(1),                                     SSP01220
     10  vl_b_newdata        bit(1);                                     SSP01230
                                                                         SSP01240
  /***********************************************************/         SSP01250
  /* entry points                                         */            SSP01260
  /***********************************************************/         SSP01270
                                                                         SSP01280
  Declare                                                                SSP01290
                                                                         SSP01300
  /* bind service to addresses */                                       SSP01310
  ssServiceBind   entry                                                  SSP01320
  (                                                                      SSP01330
   fixed(31),                     /* return code    */                   SSP01340
   fixed(31),                     /* reason code    */                   SSP01350
   character(*),                  /* service name   */                   SSP01360
   fixed(31),                     /* its length     */                   SSP01370
   pointer(31),                   /* init addr      */                   SSP01380
   pointer(31),                   /* service addr   */                   SSP01390
   pointer(31),                   /* completion addr */                  SSP01400
   fixed(31)                      /* service type   */                   SSP01410
  )                                                                      SSP01420
  external as ('BKWVBN'),                                                SSP01430
                                                                         SSP01440
  /* find service block */                                              SSP01450
  ssServiceFind   entry                                                  SSP01460
  (                                                                      SSP01470
   fixed(31),                     /* return code    */                   SSP01480
   fixed(31),                     /* reason code    */                   SSP01490
   character(*),                  /* service name   */                   SSP01500
   fixed(31),                     /* its length     */                   SSP01510
   pointer(31)                    /* S-blk address  */                   SSP01520
  )                                                                      SSP01530
  external as ('BKWVFN'),                                                SSP01540
```

Appendix I. Language Bindings **455**

```
                                                                SSP01550
/* start the server */                                          SSP01560
ssServerRun  entry                                              SSP01570
(                                                               SSP01580
 fixed(31),                  /* return code     */              SSP01590
 fixed(31)                   /* reason code     */              SSP01600
)                                                               SSP01610
external as ('BKWVRN'),                                          SSP01620
                                                                SSP01630
/* stop the server */                                           SSP01640
ssServerStop  entry                                             SSP01650
(                                                               SSP01660
 fixed(31),                  /* return code     */              SSP01670
 fixed(31)                   /* reason code     */              SSP01680
)                                                               SSP01690
external as ('BKWVSP');                                          SSP01700
                                                                SSP01710
```

Trie Bindings (SSPLXTRI COPY)

```
*COPY SSPLXTRI                                                   SSP00010
                                                                SSP00020
/************************************************/               SSP00030
/*                                            */                SSP00040
/* external bindings for trie routines        */                SSP00050
/*                                            */                SSP00060
/*                                            */                SSP00070
/* Brian Wade    April 1999                   */                SSP00080
/*                                            */                SSP00090
/************************************************/               SSP00100
                                                                SSP00110
                                                                SSP00120
/************************************************/               SSP00130
/* constants                                  */                SSP00140
/************************************************/               SSP00150
                                                                SSP00160
Declare                                                         SSP00170
                                                                SSP00180
 /* ssTrie return codes */                                      SSP00190
 ss_tri_rc_success     fixed(31)   constant(0),                 SSP00200
 ss_tri_rc_warning     fixed(31)   constant(4),                 SSP00210
 ss_tri_rc_error       fixed(31)   constant(8),                 SSP00220
 ss_tri_rc_abend       fixed(31)   constant(12),                SSP00230
                                                                SSP00240
 /* ssTrie reason codes */                                      SSP00250
 ss_tri_re_success          fixed(31)   constant(0),            SSP00260
 ss_tri_re_bad_size         fixed(31)   constant(1700+1),       SSP00270
 ss_tri_re_trie_exists      fixed(31)   constant(1700+2),       SSP00280
 ss_tri_re_out_of_storage   fixed(31)   constant(1700+3),       SSP00290
 ss_tri_re_dscr_fail        fixed(31)   constant(1700+4),       SSP00300
 ss_tri_re_trie_not_found   fixed(31)   constant(1700+5),       SSP00310
 ss_tri_re_trie_busy        fixed(31)   constant(1700+6),       SSP00320
 ss_tri_re_bad_index_len    fixed(31)   constant(1700+7),       SSP00330
 ss_tri_re_bad_capacity     fixed(31)   constant(1700+8),       SSP00340
 ss_tri_re_out_of_ds_storage  fixed(31)   constant(1700+9);     SSP00350
                                                                SSP00360
                                                                SSP00370
/************************************************/               SSP00380
/* Entry points                                */               SSP00390
/************************************************/               SSP00400
                                                                SSP00410
Declare                                                         SSP00420
                                                                SSP00430
 /* ssTrieCreate */                                             SSP00440
 ssTrieCreate entry                                             SSP00450
 (                                                              SSP00460
  fixed(31),             /* return code   */                    SSP00470
  fixed(31),             /* reason code   */                    SSP00480
  char(8),               /* trie name     */                    SSP00490
  fixed(31),             /* DS size (pgs) */                    SSP00500
  char(8),               /* ASIT          */                    SSP00510
  fixed(31)              /* ALET          */                    SSP00520
 )                                                              SSP00530
 external as ('BKWYCR'),                                         SSP00540
                                                                SSP00550
 /* ssTrieDelete */                                             SSP00560
 ssTrieDelete entry                                             SSP00570
 (                                                              SSP00580
  fixed(31),             /* return code   */                    SSP00590
```

```
  fixed(31),                /* reason code    */          SSP00600
  char(8)                   /* trie name      */          SSP00610
  )                                                        SSP00620
  external as ('BKWYDE'),                                  SSP00630
                                                           SSP00640
  /* ssTrieRecordInsert */                                 SSP00650
  ssTrieRecordInsert entry                                 SSP00660
  (                                                        SSP00670
  fixed(31),                /* return code    */          SSP00680
  fixed(31),                /* reason code    */          SSP00690
  char(8),                  /* trie name      */          SSP00700
  fixed(31),                /* trie ALET      */          SSP00710
  fixed(31),                /* record number  */          SSP00720
  char(*),                  /* index buffer   */          SSP00730
  fixed(31)                 /* index length   */          SSP00740
  )                                                        SSP00750
  external as ('BKWYRI'),                                  SSP00760
                                                           SSP00770
  /* ssTrieRecordList */                                   SSP00780
  ssTrieRecordList entry                                   SSP00790
  (                                                        SSP00800
  fixed(31),                /* return code    */          SSP00810
  fixed(31),                /* reason code    */          SSP00820
  char(8),                  /* trie name      */          SSP00830
  fixed(31),                /* trie ALET      */          SSP00840
  char(*),                  /* index buffer   */          SSP00850
  fixed(31),                /* index length   */          SSP00860
  fixed(31),                /* recnum array   */          SSP00870
  fixed(31),                /* array capacity */          SSP00880
  fixed(31)                 /* recs found     */          SSP00890
  )                                                        SSP00900
  external as ('BKWYRL');                                  SSP00910
                                                           SSP00920
```

User ID Bindings (SSPLXUID COPY)

```
*COPY SSPLXUID                                            SSP00010
                                                          SSP00020
/**********************************************************************/ SSP00030
/*                                                      */ SSP00040
/*  NAME      -    Reusable Server Kernel   PL/X bindings */ SSP00050
/*                                                      */ SSP00060
/*  FUNCTION  -    Language bindings for userid services */ SSP00070
/*                                                      */ SSP00080
/*  COPYRIGHT -                               @VR2OZOZ */ SSP00090
/*     5684-112 (C) COPYRIGHT IBM CORP.1991, 1992    @VR2OZOZ */ SSP00100
/*     LICENSED MATERIALS - PROPERTY OF IBM         @VR2OZOZ */ SSP00110
/*     SEE COPYRIGHT INSTRUCTIONS, G120-2083        @VR2OZOZ */ SSP00120
/*     ALL RIGHTS RESERVED                          @VR2OZOZ */ SSP00130
/*                                                      */ SSP00140
/*  STATUS - VM/ESA Version 2 Release 4              @VR2OZOZ */ SSP00150
/*                                                      */ SSP00160
/*  CHANGE ACTIVITY  - New for VM/ESA Version 2 Release 4 */ SSP00170
/**********************************************************************/ SSP00180
                                                          SSP00190
/********************************************************/ SSP00200
/* CONSTANTS                                          */ SSP00210
/********************************************************/ SSP00220
                                                          SSP00230
Declare                                                   SSP00240
                                                          SSP00250
  /* config constants */                                 SSP00260
  ss_uid_index_width      fixed(31) constant(64),         SSP00270
                                                          SSP00280
  /* return and reason codes */                          SSP00290
  ss_uid_rc_success       fixed(31) constant(0),          SSP00300
  ss_uid_rc_warning       fixed(31) constant(4),          SSP00310
  ss_uid_rc_error         fixed(31) constant(8),          SSP00320
  ss_uid_rc_abend         fixed(31) constant(12),         SSP00330
                                                          SSP00340
  ss_uid_re_success       fixed(31) constant(0),          SSP00350
  ss_uid_re_not_found     fixed(31) constant(100+1);      SSP00360
                                                          SSP00370
/********************************************************/ SSP00380
/* STRUCTURES                                         */ SSP00390
/********************************************************/ SSP00400
                                                          SSP00410
/********************************************************/ SSP00420
/* FUNCTIONS                                          */ SSP00430
```

```
          /****************************************************/    SSP00440
                                                                    SSP00450
          Declare                                                   SSP00460
                                                                    SSP00470
           /* routine to map user IDs */                           SSP00480
           ssUseridMap entry                                       SSP00490
           (                                                        SSP00500
             fixed(31),                    /* return code   */     SSP00510
             fixed(31),                    /* reason code   */     SSP00520
             character(*),                 /* input conn    */     SSP00530
             fixed(31),                    /* its length    */     SSP00540
             character(*),                 /* input node    */     SSP00550
             fixed(31),                    /* its length    */     SSP00560
             character(*),                 /* input user    */     SSP00570
             fixed(31),                    /* its length    */     SSP00580
             character(ss_uid_index_width), /* output user  */     SSP00590
             fixed(31)                     /* its length    */     SSP00600
           )                                                        SSP00610
           external as ('BKWBMU');                                 SSP00620
                                                                    SSP00630
```

Worker Bindings (SSPLXWRK COPY)

```
*COPY SSPLXWRK                                                      SSP00010
                                                                    SSP00020
 /********************************************************************/   SSP00030
 /*                                                            */ SSP00040
 /*  NAME      -     Reusable Server Kernel   PL/X bindings    */ SSP00050
 /*                                                            */ SSP00060
 /*  FUNCTION  -     Language bindings for worker services     */ SSP00070
 /*                                                            */ SSP00080
 /*  COPYRIGHT -                                               */ SSP00090
 /*                                                            */ SSP00100
 /*     THIS MODULE IS "RESTRICTED MATERIALS OF IBM"           */ SSP00110
 /*     5654-030 (C) COPYRIGHT IBM CORP. - 1998, 1999          */ SSP00120
 /*     LICENSED MATERIALS - PROPERTY OF IBM                   */ SSP00130
 /*     ALL RIGHTS RESERVED.                                   */ SSP00140
 /*                                                            */ SSP00150
 /*  STATUS - VM/ESA Version 2, Release 4.0                    */ SSP00160
 /*                                                            */ SSP00170
 /*  CHANGE ACTIVITY  - New for VM/ESA Version X Release Y      */ SSP00180
 /*   @SI124VM - alternate userid support in worker API        */ SSP00190
 /********************************************************************/   SSP00200
                                                                    SSP00210
 /*********************************************************/   SSP00220
 /* CONSTANTS                                          */   SSP00230
 /*********************************************************/   SSP00240
                                                                    SSP00250
 Declare                                                       SSP00260
                                                                    SSP00270
  /* return and reason codes */                               SSP00280
  ss_wrk_rc_success          fixed(31)   constant(0),          SSP00290
  ss_wrk_rc_warning          fixed(31)   constant(4),          SSP00300
  ss_wrk_rc_error            fixed(31)   constant(8),          SSP00310
  ss_wrk_rc_abend            fixed(31)   constant(12),         SSP00320
                                                                    SSP00330
  ss_wrk_re_success          fixed(31)   constant(0),          SSP00340
  ss_wrk_re_out_of_storage   fixed(31)   constant(1600+1),     SSP00350
  ss_wrk_re_bad_count        fixed(31)   constant(1600+2),     SSP00360
  ss_wrk_re_bad_flag_name    fixed(31)   constant(1600+3),     SSP00370
  ss_wrk_re_bad_flag_value   fixed(31)   constant(1600+4),     SSP00380
  ss_wrk_re_no_class         fixed(31)   constant(1600+5),     SSP00390
  ss_wrk_re_no_subordinates  fixed(31)   constant(1600+6),     SSP00400
  ss_wrk_re_algtries_exceeded fixed(31)  constant(1600+7),     SSP00410
  ss_wrk_re_autolog_fail     fixed(31)   constant(1600+8),     SSP00420
  ss_wrk_re_timer_fail       fixed(31)   constant(1600+9),     SSP00430
  ss_wrk_re_iucvcon_fail     fixed(31)   constant(1600+10),    SSP00440
  ss_wrk_re_force_fail       fixed(31)   constant(1600+11),    SSP00450
  ss_wrk_re_force_timeout    fixed(31)   constant(1600+12),    SSP00460
  ss_wrk_re_oper_delete      fixed(31)   constant(1600+13),    SSP00470
                                                                    SSP00480
  /* option flag names */                                     SSP00490
  ss_wrk_ofn_prefer_empty    fixed(31)   constant(0),          SSP00500
  ss_wrk_ofn_retry_count     fixed(31)   constant(1),          SSP00510
  ss_wrk_ofn_alt_userid      fixed(31)   constant(2),    /*@SI124VM*/ SSP00520
  ss_wrk_ofn_alt_seclabel    fixed(31)   constant(3),    /*@SI124VM*/ SSP00530
                                                                    SSP00540
  /* option flag values */                                    SSP00550
  ss_wrk_ofv_no              fixed(31)   constant(0),          SSP00560
```

```
 ss_wrk_ofv_yes             fixed(31)   constant(1);                    SSP00570
                                                                        SSP00580
/**********************************************************/            SSP00590
/* STRUCTURES                                           */              SSP00600
/**********************************************************/            SSP00610
                                                                        SSP00620
/**********************************************************/            SSP00630
/* FUNCTIONS                                            */              SSP00640
/**********************************************************/            SSP00650
                                                                        SSP00660
Declare                                                                 SSP00670
                                                                        SSP00680
 /* allocate a worker machine */                                       SSP00690
 ssWorkerAllocate entry                                                 SSP00700
 (                                                                      SSP00710
  fixed(31),                    /* return code      */                  SSP00720
  fixed(31),                    /* reason code      */                  SSP00730
  pointer(31),                  /* instance C-block */                  SSP00740
  char(8),                      /* class name       */                  SSP00750
  fixed(31),                    /* option count     */                  SSP00760
  fixed(31),                    /* option names     */                  SSP00770
  fixed(31),                    /* option values    */                  SSP00780
  pointer(31),                  /* worker C-block   */                  SSP00790
  fixed(31)                     /* connection ID    */                  SSP00800
 )                                                                      SSP00810
 external as ('BKWCAL');                                                SSP00820
                                                                        SSP00830
```

Appendix J. What's Changed Since the Beta

The reusable server kernel was available for several months before it became generally available. There are some differences between the beta level and the GA level. The following table summarizes the differences and describes the actions you must take to convert your program to run on the GA level.

Table 58. Differences Between Beta and GA Levels

Topic	Beta	GA	Action
Name of your mainline	`VSSMAIN`	`RSKMAIN`	Edit and recompile or reassemble your mainline.
Profile file name	`PROFILE VMSS`	`PROFILE RSK`	Change the name of your profile.
Subcom name	`VMSS`	`RSK`	Change your EXECs to use `ADDRESS RSK`.
Entry point names	`VSSxxx`	`BKWxxx`	Recompile or reassemble your program.
Names of CMS- or CP-managed objects the server kernel creates (mutexes, semaphores, condition variables, queues, subpools, HNDIUCV exit names, data spaces, and so on)	Often started with SS or VSS	All start with BKW or DMS	Avoid prefixes BKW and DMS.
IPC message keys, event keys, timer userwords	Often started with SS or VSS	All start with BKW	Avoid prefix BKW.
Macro library containing SSASMxxx bindings	`VSSGPI MACLIB`	`DMSGPI MACLIB`	Change the control file you use for assemblies.
Macro library containing SSPLXxxx bindings	`VSSPLX MACLIB`	`DMSRP MACLIB`	Change the control file you use for compilations.
The reusable server kernel text library	`VSS TXTLIB`	`BKWLIB TXTLIB`	Change your GLOBAL TXTLIB command.
Supplementary text library shipped with the beta	`PSL TXTLIB`	`DMSPSLK TXTLIB`	Change your GLOBAL TXTLIB command.

Table 58. Differences Between Beta and GA Levels (continued)

Topic	Beta	GA	Action
Default names for authorization data files	CMS filetypes started with VSS	CMS filetypes start with RSK	Rename your files or adjust `PROFILE RSK`.
Default name for storage group configuration file	`DEFAULT VSSSGP A`	`DEFAULT RSKSGP A`	Rename your file or adjust `PROFILE RSK`.
Default name for user ID mapping file	`DEFAULT VSSUMAP *`	`DEFAULT RSKUMAP *`	Rename your file or adjust `PROFILE RSK`.
Exit name a worker control program should use when it issues `HNDIUCV SET`	`VSSWORK`	`RSKWORK`	
Default filetype for request files arriving for the SPOOL line driver	`VSSRQST`	`RSKRQST`	Change your client or `PROFILE RSK` appropriately.
Default filetype for response files generated the SPOOL line driver	`VSSRESP`	`RSKRESP`	Change your client or `PROFILE RSK` appropriately.
Message repository file	`VSSUME TEXT`	`BKWUME TEXT`	Change the `SET LANGUAGE` command your server issues when it starts.
Runtime environment manager module	`VSSRTE MODULE`	`BKWRTE MODULE`	The old module is incompatible and must be replaced with the new one.
Message numbers	`VSScccnnnns`	`BKWcccnnnns`	Probably nothing.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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
US

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

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM 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 provide 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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

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

This information may contain 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 actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

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

Programming Interface Information

This publication primarily documents intended Programming Interfaces that allow the customer to write programs to obtain the services of z/VM.

This publication also documents information that is NOT intended to be used as Programming Interfaces of z/VM. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

`NOT-PI`

<...NOT Programming Interface information...>

`NOT-PI end`

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on IBM Copyright and trademark information (https://www.ibm.com/legal/copytrade).

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 world-wide basis.

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

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.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

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:

- The section entitled **IBM Websites** at IBM Privacy Statement (https://www.ibm.com/privacy)
- Cookies and Similar Technologies (https://www.ibm.com/privacy#Cookies_and_Similar_Technologies)

Bibliography

This topic lists the publications in the z/VM library. For abstracts of the z/VM publications, see *z/VM: General Information*.

Where to Get z/VM Information

The current z/VM product documentation is available in IBM Documentation - z/VM (https://www.ibm.com/docs/en/zvm).

z/VM Base Library

Overview

- *z/VM: License Information*, GI13-4377
- *z/VM: General Information*, GC24-6286

Installation, Migration, and Service

- *z/VM: Installation Guide*, GC24-6292
- *z/VM: Migration Guide*, GC24-6294
- *z/VM: Service Guide*, GC24-6325
- *z/VM: VMSES/E Introduction and Reference*, GC24-6336

Planning and Administration

- *z/VM: CMS File Pool Planning, Administration, and Operation*, SC24-6261
- *z/VM: CMS Planning and Administration*, SC24-6264
- *z/VM: Connectivity*, SC24-6267
- *z/VM: CP Planning and Administration*, SC24-6271
- *z/VM: Getting Started with Linux on IBM Z*, SC24-6287
- *z/VM: Group Control System*, SC24-6289
- *z/VM: I/O Configuration*, SC24-6291
- *z/VM: Running Guest Operating Systems*, SC24-6321
- *z/VM: Saved Segments Planning and Administration*, SC24-6322
- *z/VM: Secure Configuration Guide*, SC24-6323

Customization and Tuning

- *z/VM: CP Exit Customization*, SC24-6269
- *z/VM: Performance*, SC24-6301

Operation and Use

- *z/VM: CMS Commands and Utilities Reference*, SC24-6260
- *z/VM: CMS Primer*, SC24-6265
- *z/VM: CMS User's Guide*, SC24-6266
- *z/VM: CP Commands and Utilities Reference*, SC24-6268

- *z/VM: System Operation*, SC24-6326
- *z/VM: Virtual Machine Operation*, SC24-6334
- *z/VM: XEDIT Commands and Macros Reference*, SC24-6337
- *z/VM: XEDIT User's Guide*, SC24-6338

Application Programming

- *z/VM: CMS Application Development Guide*, SC24-6256
- *z/VM: CMS Application Development Guide for Assembler*, SC24-6257
- *z/VM: CMS Application Multitasking*, SC24-6258
- *z/VM: CMS Callable Services Reference*, SC24-6259
- *z/VM: CMS Macros and Functions Reference*, SC24-6262
- *z/VM: CMS Pipelines User's Guide and Reference*, SC24-6252
- *z/VM: CP Programming Services*, SC24-6272
- *z/VM: CPI Communications User's Guide*, SC24-6273
- *z/VM: ESA/XC Principles of Operation*, SC24-6285
- *z/VM: Language Environment User's Guide*, SC24-6293
- *z/VM: OpenExtensions Advanced Application Programming Tools*, SC24-6295
- *z/VM: OpenExtensions Callable Services Reference*, SC24-6296
- *z/VM: OpenExtensions Commands Reference*, SC24-6297
- *z/VM: OpenExtensions POSIX Conformance Document*, GC24-6298
- *z/VM: OpenExtensions User's Guide*, SC24-6299
- *z/VM: Program Management Binder for CMS*, SC24-6304
- *z/VM: Reusable Server Kernel Programmer's Guide and Reference*, SC24-6313
- *z/VM: REXX/VM Reference*, SC24-6314
- *z/VM: REXX/VM User's Guide*, SC24-6315
- *z/VM: Systems Management Application Programming*, SC24-6327
- *z/VM: z/Architecture Extended Configuration (z/XC) Principles of Operation*, SC27-4940

Diagnosis

- *z/VM: CMS and REXX/VM Messages and Codes*, GC24-6255
- *z/VM: CP Messages and Codes*, GC24-6270
- *z/VM: Diagnosis Guide*, GC24-6280
- *z/VM: Dump Viewing Facility*, GC24-6284
- *z/VM: Other Components Messages and Codes*, GC24-6300
- *z/VM: VM Dump Tool*, GC24-6335

z/VM Facilities and Features

Data Facility Storage Management Subsystem for z/VM

- *z/VM: DFSMS/VM Customization*, SC24-6274
- *z/VM: DFSMS/VM Diagnosis Guide*, GC24-6275
- *z/VM: DFSMS/VM Messages and Codes*, GC24-6276
- *z/VM: DFSMS/VM Planning Guide*, SC24-6277

- *z/VM: DFSMS/VM Removable Media Services*, SC24-6278
- *z/VM: DFSMS/VM Storage Administration*, SC24-6279

Directory Maintenance Facility for z/VM

- *z/VM: Directory Maintenance Facility Commands Reference*, SC24-6281
- *z/VM: Directory Maintenance Facility Messages*, GC24-6282
- *z/VM: Directory Maintenance Facility Tailoring and Administration Guide*, SC24-6283

Open Systems Adapter

- Open Systems Adapter/Support Facility on the Hardware Management Console (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC14-7580-02.pdf), SC14-7580
- Open Systems Adapter-Express ICC 3215 Support (https://www.ibm.com/docs/en/zos/2.3.0?topic=osa-icc-3215-support), SA23-2247
- Open Systems Adapter Integrated Console Controller User's Guide (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/SC27-9003-02.pdf), SC27-9003
- Open Systems Adapter-Express Customer's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.3.0/pdf/ioa2z1f0.pdf), SA22-7935

Performance Toolkit for z/VM

- *z/VM: Performance Toolkit Guide*, SC24-6302
- *z/VM: Performance Toolkit Reference*, SC24-6303

The following publications contain sections that provide information about z/VM Performance Data Pump, which is licensed with Performance Toolkit for z/VM.

- *z/VM: Performance*, SC24-6301. See z/VM Performance Data Pump.
- *z/VM: Other Components Messages and Codes*, GC24-6300. See Data Pump Messages.

RACF® Security Server for z/VM

- *z/VM: RACF Security Server Auditor's Guide*, SC24-6305
- *z/VM: RACF Security Server Command Language Reference*, SC24-6306
- *z/VM: RACF Security Server Diagnosis Guide*, GC24-6307
- *z/VM: RACF Security Server General User's Guide*, SC24-6308
- *z/VM: RACF Security Server Macros and Interfaces*, SC24-6309
- *z/VM: RACF Security Server Messages and Codes*, GC24-6310
- *z/VM: RACF Security Server Security Administrator's Guide*, SC24-6311
- *z/VM: RACF Security Server System Programmer's Guide*, SC24-6312
- *z/VM: Security Server RACROUTE Macro Reference*, SC24-6324

Remote Spooling Communications Subsystem Networking for z/VM

- *z/VM: RSCS Networking Diagnosis*, GC24-6316
- *z/VM: RSCS Networking Exit Customization*, SC24-6317
- *z/VM: RSCS Networking Messages and Codes*, GC24-6318
- *z/VM: RSCS Networking Operation and Use*, SC24-6319
- *z/VM: RSCS Networking Planning and Configuration*, SC24-6320

TCP/IP for z/VM

- *z/VM: TCP/IP Diagnosis Guide*, GC24-6328
- *z/VM: TCP/IP LDAP Administration Guide*, SC24-6329
- *z/VM: TCP/IP Messages and Codes*, GC24-6330
- *z/VM: TCP/IP Planning and Customization*, SC24-6331
- *z/VM: TCP/IP Programmer's Reference*, SC24-6332
- *z/VM: TCP/IP User's Guide*, SC24-6333

Prerequisite Products

Device Support Facilities

- Device Support Facilities (ICKDSF): User's Guide and Reference (https://www.ibm.com/docs/en/SSLTBW_2.5.0/pdf/ickug00_v2r5.pdf), GC35-0033

Related Products

XL C++ for z/VM

- *XL C/C++ for z/VM: Runtime Library Reference*, SC09-7624
- *XL C/C++ for z/VM: User's Guide*, SC09-7625

z/OS

IBM Documentation - z/OS (https://www.ibm.com/docs/en/zos)

Index

IBM®

Product Number: 5741-A09

Printed in USA