VMS Software

VSI OpenVMS

VSI DECnet-Plus OSAK Programming Reference

VSI DECnet-Plus OSAK Programming Reference

VMS Software

Preface

This book contains reference material that you need when using the OSI Applications Kernel (OSAK) interface to create Open Systems Interconnection (OSI) applications on any supported operating system. Use this book with *VSI DECnet-Plus OSAK Programming*.

1. About VSI

VMS Software, Inc. (VSI) is an independent software company licensed by Hewlett Packard Enterprise to develop and support the OpenVMS operating system.

2. Intended Audience

The audience for this manual is OSI application programmers who require a basic understanding of the upper-layer standards implemented by the OSAK product.

3. Prerequisites

Before using the OSAK interface, you should ensure that you:

*   Have installed DECnet-Plus and the OSAK interface

    The *DECnet-Plus for OpenVMS Applications Installation and Advanced Configuration Guide* explains how to install DECnet-Plus and the OSAK interfaces.

*   Have *VSI DECnet-Plus OSAK Programming* available.

*   Understand the parts of the OSI standards that apply to the protocols your application uses. *VSI DECnet-Plus OSAK Programming* lists the relevant standards.

    This book (and *VSI DECnet-Plus OSAK Programming*) assume that you understand the terminology and concepts used in the relevant standards.

4. Related Documents

VSI DECnet-Plus OSAK Programming gives a list of the relevant international standards.

You may also need to refer to the *VSI DECnet-Plus Planning Guide*.

5. VSI Encourages Your Comments

You may send comments or suggestions regarding this manual or any VSI document by sending electronic mail to the following Internet address: `<docinfo@vmssoftware.com>`. Users who have VSI OpenVMS support contracts through VSI can contact `<support@vmssoftware.com>` for help with this product.

6. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at https://docs.vmssoftware.com.

7. Typographical Conventions

VMScluster systems are now referred to as OpenVMS Cluster systems. Unless otherwise specified, references to OpenVMS Cluster systems or clusters in this document are synonymous with VMScluster systems.

The contents of the display examples for some utility commands described in this manual may differ slightly from the actual output provided by these commands on your system. However, when the behavior of a command differs significantly between OpenVMS Alpha and Integrity servers, that behavior is described in text and rendered, as appropriate, in separate examples.

In this manual, every use of DECwindows and DECwindows Motif refers to DECwindows Motif for OpenVMS software.

The following conventions are also used in this manual:

Convention	Meaning
Ctrl/*x*	A sequence such as **Ctrl/** *x* indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 *x*	A sequence such as PF1 *x* indicates that you must first press and release the key labeled PF1 and then press and release another key or a pointing device button.
Return	In examples, a key name enclosed in a box indicates that you press a key on the keyboard. (In text, a key name is not enclosed in a box.)
…	A horizontal ellipsis in examples indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.
( )	In command format descriptions, parentheses indicate that you must enclose the options in parentheses if you choose more than one.
[ ]	In command format descriptions, brackets indicate optional choices. You can choose one or more items or no items. Do not type the brackets on the command line. However, you must include the brackets in the syntax for OpenVMS directory specifications and for a substring specification in an assignment statement.
[|]	In command format descriptions, vertical bars separate choices within brackets or braces. Within brackets, the choices are options; within braces, at least one choice is required. Do not type the vertical bars on the command line.
{ }	In command format descriptions, braces indicate required choices; you must choose at least one of the items listed. Do not type the braces on the command line.
bold text	This typeface represents the introduction of a new term. It also represents the name of an argument, an attribute, or a reason.
italic text	Italic text indicates important information, complete titles of manuals, or variables. Variables include information that varies in system output (Internal error *number*),

Convention	Meaning
	in command lines (/PRODUCER= *name*), and in command parameters in text (where *dd* represents the predefined code for the device type).
UPPERCASE TEXT	Uppercase text indicates a command, the name of a routine, the name of a file, or the abbreviation for a system privilege.
`Monospace type`	Monospace type indicates code examples and interactive screen displays. In the C programming language, monospace type in text identifies the following elements: keywords, the names of independently compiled external functions and files, syntax summaries, and references to variables or identifiers introduced in an example.
-	A hyphen at the end of a command format description, command line, or code line indicates that the command or statement continues on the following line.
numbers	All numbers in text are assumed to be decimal unless otherwise noted. Nondecimal radixes—binary, octal, or hexadecimal—are explicitly indicated.

8. Abbreviations

The following abbreviations are used in this book:

ACSE	Association Control Service Element
APDU	application protocol data unit
ASN.1	Abstract Syntax Notation One
BER	basic encoding rules
CLNS	Connectionless-Mode Network Service
CONS	Connection-Oriented Network Service
DCS	defined context set
ISO	International Organization for Standardization
NSAP	network service access point
OSAK	OSI Applications Kernel
OSI	Open Systems Interconnection
PCI	protocol control information
PDU	protocol data unit
PDV	presentation data value
PSEL	presentation selector
ROSE	Remote Operations Service Element
SPDU	session protocol data unit
SSEL	session selector
TCP/IP	Transmission Control Protocol/Internet Protocol
TLV	tag, length, and value
TSDU	transport service data unit
TSEL	transport selector

Chapter 1. OSAK Routines

This chapter contains the following information about the OSAK interface:

- The names of the include files, and where to find them

- A description of the OSAK parameter block

- A description of each OSAK data type

- A description of each OSAK routine

Communications software that conforms to the OSI standards follows a model of layers. Each layer provides a service to the layer immediately above it. The layer that provides the service is called the **provider**; the layer that uses the service is called the **user**. Note this use of the term 'user' in this book, in the OSI standards, and in other books that deal with the OSAK software; a 'user' is not a person.

1.1. Include Files

The include files for the OSAK interface are:

- osak_api.h

- osak_api_codes.h

- osak_api_messages.h

Their locations differ according to the operating system:

OpenVMS	SYS$COMMON:[SYSLIB]
ULTRIX	/usr/include
UNIX	/usr/include/osi

1.2. OSAK Parameter Block

This section describes the parameter block, *osak_parameter_block* data type, and the data types it includes.

Table 1.1 lists the parameters in the parameter block, describes them briefly, and shows their data types.

Table 1.1. OSAK API Parameters

Parameter	Brief Description	Data Type
abort_ppdu	Presentation provider abort identifier	osak_abort_ppdu
abort_reason[1]	Reason for abort	osak_abort_reason
acontext	Application context name	Address (osak_mem_descriptor)
acse_pci_eoc	End of contents count (ACSE)	Unsigned long integer
action_result	Acceptance or rejection of release request	Address (osak_action_result)
activity_id	Activity identifier	osak_mem_descriptor
activity_reason	Reason code	Address (osak_activity_reason)

Parameter	Brief Description	Data Type
alloc_param	User-defined parameter for use with *alloc_rtn* and *dealloc_rtn*	Unsigned long integer
alloc_rtn	Memory allocation routine	osak_rtn
api_version	OSAK API version to be used	Unsigned long integer
called_aei	Responder application entity invocation	Address (osak_aei)
calling_aei	Initiator application entity invocation	Address (osak_aei)
completion_param (OpenVMS systems only)	User-defined parameter for use with *completion_rtn*	Unsigned long integer
completion_rtn

(OpenVMS systems only) | Completion routine | osak_rtn |
data_length	Total data octets	Unsigned long integer
data_separation	Data separation flag	osak_data_separation
dealloc_rtn	Memory deallocation routine	osak_rtn
event_type	Type of event	osak_event
exception_reason	Reason for exception report	osak_exception_reason
func	Service identifier	Unsigned long integer
functional_units	Presentation and session functional units	Address (osak_fus)
initial_serial_number	Serial number of first synchronization point	Address (osak_sync_point)
initial_tokens	Initial token settings	Address (osak_token_setting)
local_abort	Origin of abort	Long integer
local_aei	Application-entity invocation of the calling process	Address (osak_aei)
local_data	Buffers holding redirected local user data	Address (osak_mem_descriptor)
more_flag	Data segmentation flag	Long integer
next_pb	Pointer to next parameter block	Address (osak_parameter_block)
old_activity_id	Interrupted activity identifier	osak_mem_descriptor
old_sconnection_id	Previous session connection	Address (osak_sconnection_id)
pb_length	Size of parameter block	Unsigned long integer
pcontext_list	Proposed DCS	Address (osak_pcontext_proposal)
pcontext_del_list	Proposed deletions from DCS	Address (osak_pcontext_deletion)
pcontext_id_list	Presentation context identifiers and associated transfer syntaxes	Address (osak_pcontext_id)
pcontext_del_res_list	Response to proposed deletions from DCS	Address (osak_pcontext_deletion_result)

Parameter	Brief Description	Data Type
pcontext_redirect_list	DCS on redirected association	Address (osak_pcontext)
pcontext_res_list	Response to proposed DCS	Address (osak_pcontext_proposal_result)
pdefault_context	Proposed default presentation context	Address (osak_default_context)
pdefault_context_res	Response to proposed default presentation context	Address (osak_default_context_result)
peer_data	User data (inbound)	Address (osak_buffer)
port_id	Port identifier	osak_port
pres_pci_eoc	End of contents count (presentation)	Unsigned long integer
process_id	Process identifier	Address (osak_process_id)
process_name	Process name	Address (osak_mem_descriptor)
protocol_versions	Protocol version numbers	Address (osak_protocol_versions)
rcv_data_list	Buffers holding redirected peer data	Address (osak_buffer)
redirect_state	State of protocol machine	osak_state
reject_reason	Reason for rejecting connection request	osak_reject_reason
release_reason	Reason for releasing association	osak_release_reason
release_resp_reason	Reason for rejecting release request	osak_release_resp_reason
request_tokens	Tokens requested from peer entity	Address (osak_token_setting)
responding_aei	Responding application entity invocation	Address (osak_aei)
resync_type	Type of resynchronization	osak_resync_type
sconnect_id	Session connection information	Address (osak_sconnect_id)
segmentation	Session segmentation use and size of TSDU	Address (osak_segmentation)
status_block	Status code	osak_status_block
sync_confirm	Confirmation flag for a minor synchronization point	osak_sync_confirm
sync_point	Synchronization point serial number	Address (osak_sync_point)
token_item	Token positions	Address (osak_token_setting)
tokens	Distribution of tokens	Address (osak_token_setting)
transport_template	Transport template identifier list	Address (osak_transport_templates)
tsdu_ptr	Pointer to list of user buffers	Address (osak_buffer)

Parameter	Brief Description	Data Type
user_context	Space for applications to store local information	Address
user_data	User data (outbound)	Address (osak_buffer)
workspace[2]	Parameter block workspace	None
ws_length	Length of workspace	Unsigned long integer

[1]Returned by OSAK. Cannot be specified by the application.

[2]The workspace is a section of memory at the end of the structure. It is not a field in the structure itself.

1.3. Data Type Definitions

This section describes the data types specific to the OSAK interface. The data types are described in alphabetical order. Where a data type consists of fields, these are presented in table form.

For more detailed information about parameters, see Section 1.4.

osak_abort_ppdu

Unsigned long integer

osak_abort_reason

Unsigned long integer

osak_acse_version

Field	Brief Description	Data Type
version1	ACSE version 1	Bit field mask

osak_action_result

Unsigned long integer

osak_activity_reason

Unsigned long integer

osak_aei

Field	Brief Description	Data Type
paddress	Presentation address	osak_paddress (see the section called "osak_paddress")
aetitle	Application-entity title	osak_aetitle (see the section called "osak_aetitle")
aeiid	Application-entity invocation identifier	osak_aeiid (see the section called "osak_aeiid")

osak_aeiid

Field	Brief Description	Data Type
apiid	Application-process invocation identifier, a TLV encoding of an ASN.1 integer	osak_mem_descriptor
aeiid	Application-entity invocation identifier, an ASN.1 integer TLV	osak_mem_descriptor

osak_aetitle

Field	Brief Description	Data Type
aptitle	Application-process title, an ASN.1 object identifier TLV or an encoded RDN	osak_mem_descriptor
ae_qualifier	Application-entity qualifier, an ASN.1 integer TLV	osak_mem_descriptor

osak_api_version

Unsigned long integer

osak_buffer

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_buffer)
buffer_ptr	Pointer to beginning of buffer	Unsigned octet
buffer_length	Length of buffer	Unsigned long integer
data_ptr	Start of user data	Unsigned octet
data_length	Length of user data	Unsigned long integer

osak_data_separation

Unsigned longword integer

osak_default_context

Field	Brief Description	Data Type
ts_name	An ASN.1 object identifier TLV describing a transfer syntax	osak_mem_descriptor
as_name	An ASN.1 object identifier TLV	osak_mem_descriptor

osak_default_context_result

Unsigned long integer

osak_event

Unsigned long integer

osak_exception_reason

Unsigned long integer

osak_fus

Field	Brief Description	Data Type
half_duplex	Half-duplex functional unit selector	Bit field mask
duplex	Duplex functional unit selector	Bit field mask
expedited	Expedited functional unit selector	Bit field mask
syncminor	Minor synchronization functional unit selector	Bit field mask
syncmajor	Major synchronization functional unit selector	Bit field mask
resynchronize	Resynchronize functional unit selector	Bit field mask
activities	Activities functional unit selector	Bit field mask
negotiated_release	Negotiated release functional unit selector	Bit field mask
capability_data	Capability data functional unit selector	Bit field mask
exceptions [1]	Exceptions functional unit selector	Bit field mask
data_separation	Data separation functional unit selection	Bit field mask
typed_data	Typed data functional unit selector	Bit field mask
context_management	Context management functional unit selector	Bit field mask

[1]Supported on OpenVMS and UNIX systems only.

osak_handle

Field	Brief Description	Data Type
id	Handle identifier	Unsigned long integer
request_mask	Request event mask	Unsigned octet
returned_mask	Returned event mask	Unsigned octet

osak_handle_count

Unsigned long integer

osak_mem_descriptor

Field	Brief Description	Data Type
size	Length of buffer in octets	Unsigned long integer
pointer	Reference to buffer	Address (unsigned octet)

osak_nsap

Field	Brief Description	Data Type
next	Next network service access point (NSAP)	Address (osak_nsap)
id	Address	osak_mem_descriptor

Field	Brief Description	Data Type
type	A constant defining the network protocol	Long integer

osak_paddress

Field	Brief Description	Data Type
psel	Presentation selector	osak_mem_descriptor
ssel	Session selector	osak_mem_descriptor
tsel	Transport selector	osak_mem_descriptor
nsap	Network service access point	osak_nsap (see the section called "osak_nsap")

osak_parameter_block

See Section 1.2.

osak_pcontext

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_pcontext)
pcontext_id	An ASN.1 integer TLV describing a presentation context identifier	osak_mem_descriptor
ts_name	An ASN.1 object identifier TLV describing a transfer syntax name	osak_mem_descriptor
as_name	An ASN.1 object identifier TLV describing an abstract syntax name	osak_mem_descriptor

osak_pcontext_deletion

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_pcontext_deletion)
pcontext_id	An ASN.1 integer TLV describing a presentation context identifier	osak_mem_descriptor

osak_pcontext_deletion_result

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_pcontext_deletion_result)
result	Response to proposal to delete a presentation context from the defined context set	Unsigned long integer

osak_pcontext_id

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_pcontext_id)

Field	Brief Description	Data Type
pcontext_id	An ASN.1 integer TLV describing a presentation context identifier	osak_mem_descriptor
ts_name	An ASN.1 object identifier TLV describing a transfer syntax name	osak_mem_descriptor

osak_pcontext_proposal

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_pcontext_proposal)
pcontext_id	An ASN.1 integer TLV	osak_mem_descriptor
ts_list	List of names of supported transfer syntaxes	Address (osak_ts_list)
as_name	An ASN.1 object identifier TLV	osak_mem_descriptor

osak_pcontext_proposal_result

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_pcontext_proposal_result)
ts_name	An ASN.1 object identifier TLV describing a transfer syntax	osak_mem_descriptor
result	Response to proposal	Unsigned long integer
reason	Provider reason	Unsigned long integer

osak_port

Address (unsigned octet)

osak_protocol_versions

Field	Brief Description	Data Type
acse_version	ACSE versions proposed	osak_acse_version
pversion	Presentation versions proposed	osak_pversion
sversion	Session versions proposed	osak_sversion

osak_process_id

Unsigned long integer

osak_pversion

Field	Brief Description	Data Type
version1	Presentation version 1	Bit field mask

osak_reject_reason

Unsigned long integer

osak_release_reason

Unsigned long integer

osak_release_resp_reason

Unsigned long integer

osak_resync_type

Unsigned long integer

osak_rtn

Unsigned long integer (*osak_rtn)()

osak_sconnect_id

Field	Brief Description	Data Type
ss_user_ref	Session service user reference	osak_mem_descriptor
common_ref	Common reference	osak_mem_descriptor
add_ref_info	Additional reference information	osak_mem_descriptor

osak_sconnection_id

Field	Brief Description	Data Type
called_ss_user_ref	Called session service user reference	osak_mem_descriptor
calling_ss_user_ref	Calling session service user reference	osak_mem_descriptor
common_ref	Common reference	osak_mem_descriptor
add_ref_info	Additional reference information	osak_mem_descriptor

osak_segmentation

Field	Brief Description	Data Type
init_resp	Segmentation in the direction from initiator to responder	Unsigned short integer
resp_init	Segmentation in the direction from responder to initiator	Unsigned short integer

osak_state

Field	Brief Description	Data Type
pm_state	State of the association	Unsigned octet

Field	Brief Description	Data Type
initiator	True if the peer entity requesting redirection is the initiator, false if the peer entity requesting redirection is the responder	Long integer

osak_status_block

Field	Brief Description	Data Type
osak_status_1	OSAK status code	Unsigned long integer
osak_status_2	Secondary OSAK status code	Unsigned long integer
transport_status_1	Generic transport provider status	Unsigned long integer
transport_status_2	Specific transport provider status	Unsigned long integer

osak_sversion

Field	Brief Description	Data Type
version1	Session version 1	Bit field mask
version2	Session version 2	Bit field mask

osak_sync_confirm

Long integer

osak_sync_point

Unsigned long integer

osak_transport_templates

Field	Brief Description	Data Type
next	Pointer to next template in list	Address (osak_template)
name	Name of transport template	osak_mem_descriptor

osak_time

Unsigned long integer

osak_token_setting

Field	Brief Description	Data Type
data	Data token selector	Bit field mask length 2
sync_minor	Synchronize minor token selector	Bit field mask length 2
major_activity	Major activity token selector	Bit field mask length 2

Field	Brief Description	Data Type
release	Release token selector	Bit field mask length 2

osak_ts_list

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_ts_list)
ts_name	A TLV for an ASN.1 object identifier describing a transfer syntax	osak_mem_descriptor

osak_transport_templates

Field	Brief Description	Data Type
next	Pointer to next template in the list	Address (osak_transport_templates)
name	Transport template name	osak_mem_descriptor

1.4. Routine Descriptions

This section contains a description of each OSAK routine. Sections 1.4.1 and 1.4.2 describe the arguments and parameters common to all OSAK outbound services.

1.4.1. Arguments Common to All Outbound Services

This section describes the *port* and *parameter_block* arguments. These descriptions apply to all OSAK outbound service routines.

port

Identifies the association on which this service call is being made. You should specify the port in all the outbound service calls that you make on an association.

parameter_block

The address of a parameter block. A parameter block is a structure that contains all possible parameters for all OSAK services. The OSAK interface uses only the relevant parameters in each service call, ignoring the rest. Section 1.2 describes the structure of a parameter block.

For each routine, some parameters are mandatory and some are optional. Optional parameters are enclosed in square brackets in the Syntax section of each routine description. These parameters are not optional across the interface; you must specify values for all optional and mandatory fields and explicitly set to null any optional parameters that you do not want to use. Some parameters have dependencies on others; the routine descriptions indicate these dependencies.

1.4.2. Parameters Common to All Outbound Services

This section contains descriptions of the parameters that are common to all the OSAK outbound services:

alloc_param

The address of a user-defined structure. You can use this structure with the allocation and deallocation routines you are supplying, according to the needs of your application.

To indicate that *alloc_param* is not in use, make it null.

alloc_rtn

The address of the entry address of a memory allocation routine. You should supply a non-null value for this parameter. The OSAK interface returns the address of the allocated memory if the call is successful, and zero if it is not.

You should supply a routine that meets the memory allocation requirements of your application. The OSAK interface uses this routine only for internal memory management, not for returning inbound parameter values to your application.

The allocation routine should have the following syntax:

```
unsigned char *alloc_rtn(size, alloc_param)
    unsigned int size;
    unsigned int alloc_param;
```

The *size* parameter is the number of octets of memory being requested.

A jacket routine is a user-written routine designed to set up the parameters for an existing routine. The user-written routine surrounds the call to the existing routine. For example, your allocation routine surrounds `lib$get_vm` (OpenVMS systems) or `malloc` (UNIX and ULTRIX systems).

The following code is an example of a jacket routine using the existing routine `malloc`:

```
unsigned char *alloc_rtn(size, alloc_param)
    unsigned int size;
    unsigned int alloc_param;
{
    return malloc(size);
}
```

The following code is an example of a jacket routine using the existing routine `lib$get_vm`:

```
unsigned char *alloc_rtn(size, alloc_param)
    unsigned int size;
    unsigned int alloc_param;
{
    integer status;
    unsigned char *ptr;
    status = lib$get_vm &size, &ptr, 0);
    if (status & 0x01)
        return ptr;
    else
        return 0;
}
```

If the memory allocation routine fails to allocate memory, it should return a null pointer.

completion_param (OpenVMS systems only)

The address of a user-defined structure. You can use any structure that you need with the completion routine you are supplying. For example, you can use a generic completion routine in several different service calls. You can use the *completion_param* parameter to specify which service has finished.

completion_rtn (OpenVMS systems only)

The entry point of a completion routine.

data_length

You can use this parameter when you are sending segmented data, to specify the total length of the data being sent.

If you specify this length, the OSAK interface does not have to wait for the sender to supply all the data that it wants to send. When the sender passes a data segment to the OSAK interface, the interface can send that segment immediately. This improves the throughput and memory utilization of your application.

This parameter does not apply to user information. When you are using the routines `osak_data_req` or `osak_typed_req`, the OSAK interface does not wait for the full amount of data to arrive from the requester, because in these services, OSAK does not encode the length of the data in the PCI.

dealloc_rtn

The address of the entry address of a memory deallocation routine. You should supply a non-null value for this parameter.

You should supply a routine that meets the memory deallocation requirements of your application. The OSAK interface uses this routine only for internal memory management, not for returning inbound parameter values to your application.

The deallocation routine should have the following syntax:

```
unsigned long int dealloc_rtn(size, ptr, alloc_param)
    unsigned long int size;
    unsigned char *ptr;
    unsigned long int alloc_param;
```

The *size* parameter is the number of octets of memory to be deallocated. The OSAK interface always deallocates the same amount of memory as it allocated using your allocation routine. Your deallocation routine can ignore the *size* parameter if it does not need the number of octets of memory.

The *ptr* parameter is a pointer to the memory to be deallocated.

The possible return values of `dealloc_rtn` are:

• Zero, indicating success

• Any other number, indicating failure

If the call to the deallocation routine fails, the OSAK interface writes the following values to the *status_block* parameter:

• OSAK_S_DEALLOCERR in the *osak_status_1* field

• The value returned by the deallocation routine in the *osak_status_2* field

A jacket routine is a user-written routine designed to set up the parameters for an existing routine. The user-written routine surrounds the call to the existing routine. For example, your deallocation routine surrounds `lib$free_vm` (OpenVMS systems) or `free` (UNIX and ULTRIX systems).

The following code is an example of a jacket routine using the existing routine `free`:

```
unsigned long int dealloc_rtn (size, ptr, alloc_param)
    unsigned long int size;
    unsigned char *ptr;
    unsigned long int alloc_param;
{
    extern void free();
    free (ptr);
    return 0;
}
```

The following code is an example of a jacket routine using the existing routine `lib$free_vm`:

```
unsigned long int dealloc_rtn (size, ptr, zone)
    unsigned long int size;
    unsigned char *ptr;
    unsigned long int *zone;
{
    unsigned long int status;
    status = lib$free_vm (&size, ptr, zone);
    return ((status & 1) & 0:status);
}
```

func

In this parameter, the OSAK interface returns a code identifying the service you are calling.

more_flag

If you are sending segmented user data, use this parameter to indicate whether there is more user data to follow. Send any further user data on a call or calls to `osak_send_more`.

Set the parameter to true if there are more segments of data to follow and to false if you are sending the final segment of data.

pb_length

This parameter specifies the size of your parameter block structure. It should not include the size of the workspace.

port_id

In this parameter, the OSAK interface returns the port with which the parameter block passed on a call is associated.

This parameter is relevant if you are using completion routines. When a completion routine starts to run, indicating that a service has been completed, you can collect that service's parameter block and user buffers from the OSAK interface. The only way to find out which port the parameter block is associated with is to examine the **port_id** parameter.

status_block

When a requested service finishes, the OSAK interface returns a status code in this parameter. If the result is OSAK_S_TRANSERR, the OSAK interface also returns a transport provider status code. Chapter 6 lists all the OSAK status codes.

user_context

The address of an area in which you can store local information that is relevant to your application, for example, parameter block context information.

user_data

The address of the head of a linked list of user buffers. The list consists of zero, one, or more buffers containing segments of encoded user data that you want to send across an association.

ws_length

This parameter specifies the size of the workspace you allocate as an extension to the parameter block. The workspace should be at least the minimum size defined by the OSAK interface as OSAK_C_MINIMUM_WS.

osak_abort_req

osak_abort_req — Aborts an association.

Syntax

`status = osak_abort_req (port, parameter_block)`

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
abort_reason	osak_abort_reason	read only
[abort_ppdu]	Unsigned long integer	read only
[pcontext_id_list]	osak_pcontext_id	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_abort_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

abort_reason

Specifies the reason the association is being aborted. If the reason you specify is OSAK_C_PP_ABORT_INVALID_VALUE, indicating that an incoming event contains an invalid presentation protocol data unit (PDU) parameter value, you should also specify the *abort_ppdu* parameter.

If you specify provider abort in the *abort_reason* parameter, the OSAK interface ignores all the parameters in the parameter block except for the *abort_ppdu* parameter. Section 10.2 gives the possible values this parameter can have.

abort_ppdu

Specifies the identifier of an incoming event that is the cause of a provider abort because it contains an invalid presentation PDU parameter value. Section 10.1 gives the possible values this parameter can have.

You need to specify this parameter only if the value in the *abort_reason* parameter is OSAK_C_PP_ABORT_INVALID_VALUE.

pcontext_id_list

The address of the head of a linked list of structures, each one specifying a presentation context identifier and the identifier of its associated transfer syntax.

The list should include the presentation context for ACSE and any presentation contexts for which there is user data encoded in the *user_data* parameter.

This parameter must only be used (and is mandatory) if both the following conditions are true:

- The abort is an ACSE user abort

- Session version 2 is being used

user_data

You should use this parameter only if the following two conditions are true:

- The abort is an ACSE user abort

- Session version 2 is being used

Description

The *abort_reason* parameter indicates whether the abort originates from the provider or the user of the service. If the abort is a user abort and you want to send user data on the call, you can segment the user data between the call to `osak_abort_req` and calls to `osak_send_more`.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.

OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid in the current state of the association.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_close_port
osak_release_req

osak_accept_rsp

osak_accept_rsp — Accepts an association attempt.

Syntax

`status = osak_accept_rsp (port, parameter_block)`

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[acontext]	osak_mem_descriptor	read only
[responding_aei]	osak_aei	read only
[sconnect_id]	osak_sconnect_id	read only
[segmentation]	osak_segmentation	read only
[initial_serial_number]	osak_sync_point	read only
[initial_tokens]	osak_token_setting	read only
[request_tokens]	osak_token_setting	read only
[functional_units]	osak_fus	read only

Parameters Used	Data Type	Access
pcontext_res_list	osak_pcontext_proposal_result	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_accept_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

acontext

The address of a structure you can use to specify the address of an ASN.1 object identifier TLV for the application context name. If you do not assign a value to this parameter, the OSAK interface supplies the value that was received in the A-ASSOCIATE indication to which this call is a response.

responding_aei

The address of a structure you can use to specify the title of the responding application entity. See Section 1.3 for a description of the data type.

You can set any of the fields of any of the substructures to null, though you should not change the transport selector. The *nsap* field is ignored in this routine.

The session selector should be no longer than 16 octets.

sconnect_id

The address of a structure containing three substructures, each one specifying a session connection reference parameter:

- *ss_user_ref* – maximum size 64 octets

- *common_ref* – maximum size 64 octets

- *add_ref_info* – maximum size 4 octets

If you omit this parameter or make any of the fields null, the OSAK interface does not send the associated session reference parameters.

segmentation The address of a structure you can use to specify in which direction data should be segmented. The structure contains two fields:

- *init_resp*

- *resp_init*

A value other than zero in the *init_resp* field indicates that segmentation is to be used on data passing from the initiator to the responder. The value specifies the maximum TSDU size.

A value other than zero in the *resp_init* field indicates that segmentation is to be used on data passing from the responder to the initiator. The value specifies the maximum TSDU size.

The maximum value allowed in either field is 65,535. The value may not be greater than that proposed by the initiator.

You can use segmentation in both directions, in only one direction, or in neither direction. If this parameter is not specified, OSAK accepts whatever the initiator proposes.

initial_serial_number

The address of the serial number of the initial synchronization point on the association.

You can assign a value to this parameter if both the following conditions are true:

- The major synchronize, the minor synchronize, or the resynchronize functional unit is selected.

- The activity management functional unit is not selected.

If you do not assign a value to this parameter, the OSAK interface supplies the value that was received in the A-ASSOCIATE indication to which this call is a response.

initial_tokens

The address of a structure you can use to specify the initial token settings for the association.

If the A-ASSOCIATE indication does not specify that the responder should choose the token setting, the values in the structure should be the same as those in the A-ASSOCIATE indication. If you specify different values, the OSAK interface aborts the association.

If you make this parameter null when the A-ASSOCIATE indication specifies that the responder should choose the token setting, the OSAK interface gives all the available tokens to the initiator. The available tokens are those tokens of which the corresponding functional units are selected in the A-ASSOCIATE indication. Section 10.15 gives the possible values this parameter can have.

request_tokens

The address of a structure you can use to specify the tokens that the responder requires from the initiator. The OSAK interface ignores this parameter if no tokens are in use.

functional_units

The address of a structure you can use to specify the functional units required at the ACSE, presentation, and session levels. If you use this parameter, you should specify functional units for all levels. If you omit this parameter, the OSAK interface uses the functional units selected in the event A-ASSOCIATE indication.

To specify data separation, select the data separation and minor synchronize functional units, and do not select the activity management functional unit. The OSAK interface returns OSAK_S_INVFUS if you do not specify the correct combination of functional units.

pcontext_res_list

The address of the head of a linked list of structures, each one of which gives the response to one proposed presentation context.

This parameter is mandatory because a presentation context for the ACSE abstract syntax should always be proposed on an A-ASSOCIATE request, and should always be accepted on the corresponding A-ASSOCIATE-accept response. Section 10.13 gives the possible values the parameter can have.

In the linked list, the following rules apply:

- The *next* field can contain the value zero, indicating the end of the list

- The *ts_name* should contain one of the transfer syntax names proposed for this abstract syntax. This field is only necessary if you accept the abstract syntax.

- The *result* field specifies whether you accept or reject the transfer syntax. Section 10.13 gives the possible values the parameter can have.

- The field specifying the reason for rejecting an abstract syntax is ignored if the *result* field is not a provider reject. Section 10.7 gives the possible values the parameter can have.

Description

Call this routine after receiving an A-ASSOCIATE indication to accept the association request.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVFUS	The functional units are invalid.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_NOSYNCPNT	The synchronization point serial number is missing.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_associate_req

osak_open_responder
osak_reject_rsp

osak_act_discard_req

osak_act_discard_req — Terminates an activity and cancels its effects.

Syntax

status = osak_act_discard_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[activity_reason]	osak_activity_reason	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_act_discard_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameter Used

activity_reason

The address of a value specifying the reason for discarding the activity. If you make the address null, no reason is specified. Section 10.4 gives the possible values this parameter can have.

Description

You can use this service only if the activity management functional unit is selected.

If you are using session version 1, there is no user data on this service and therefore no segmentation is allowed and the *more_flag* parameter must be set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_end_req
osak_act_discard_rsp

osak_act_discard_rsp

osak_act_discard_rsp — Responds to a request to discard an activity.

Syntax

```
status = osak_act_discard_rsp (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only

Parameters Used	Data Type	Access
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_act_discard_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameter Used

token_item

The address of a structure you can use to specify the tokens that the accepter wants from the requester. The structure consists of four fields corresponding to the four tokens.

In each field, the only values allowed are zero and one:

- Zero means that the requester does not want this token from the accepter.

- One means that the requester wants this token from the accepter.

Description

You can use this service only if the activity management functional unit is selected.

Call the routine after receiving a P-ACTIVITY-DISCARD indication.

If you are using session version 1, there is no user data on this service and therefore no segmentation is allowed and the *more_flag* parameter must be set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.

OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_discard_req

osak_act_end_req

osak_act_end_req — Terminates an activity and saves its effects.

Syntax

`status = osak_act_end_req (port, parameter_block)`

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[sync_point]	osak_sync_point	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_act_end_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

sync_point

The address of the serial number of the major synchronization point current when the activity ends. The OSAK interface sets this value.

token_item

The address of a structure you can use to specify the tokens that the requester is passing to the accepter. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the requester is not passing this token to the accepter.

- One means that the requester is passing this token to the accepter.

Description

You can use this service only if the activity management functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_discard_req
osak_act_end_rsp

osak_act_end_rsp

osak_act_end_rsp — Responds to a request to end an activity.

Syntax

```
status = osak_act_end_rsp (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only

Parameters Used	Data Type	Access
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_act_end_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameter Used

token_item

The address of a structure you can use to specify the tokens that the requester wants from the accepter. The structure consists of four fields corresponding to the four tokens.

In each field, the only values allowed are zero and one:

• Zero means that the requester does not want this token from the accepter.

• One means that the requester wants this token from the accepter.

Description

Call this routine after receiving a P-ACTIVITY-END indication.

You can use this service only if the activity management functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.

OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_end_req

osak_act_interrupt_req

osak_act_interrupt_req — Interrupts a lower-priority activity on an association.

Syntax

status = osak_act_interrupt_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[activity_reason]	osak_activity_reason	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_act_interrupt_req (port, parameter_block)
osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameter Used

activity_reason The address of a value specifying the reason for interrupting the activity. If you make the address null, no reason is specified. Section 10.4 lists the possible values of this parameter.

Description

You can use this service only if the activity management functional unit is selected.

You should determine the relative priority of activities in your application, according to the needs and purpose of the application.

If you are using session version 1, there is no user data on this service; therefore, no segmentation is allowed and the *more_flag* parameter must be set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_resume_req
osak_act_interrupt_rsp

osak_act_interrupt_rsp

osak_act_interrupt_rsp — Responds to a request to interrupt an activity.

Syntax

```
status = osak_act_interrupt_rsp (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only

Parameters Used	Data Type	Access
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_act_interrupt_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameter Used

token_item

The address of a structure you can use to specify the tokens that the accepter wants from the requester. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the requester does not want this token from the accepter.

- One means that the requester wants this token from the accepter.

Description

You can use this service only if the activity management functional unit is selected.

Call this routine after receiving a P-ACTIVITY-INTERRUPT indication.

If you are using session version 1, there is no user data on this service; therefore, no segmentation is allowed and the *more_flag* parameter must be set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.

OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_interrupt_req

osak_act_resume_req

osak_act_resume_req — Requests the resumption of an interrupted activity.

Syntax

status = osak_act_resume_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
activity_id	osak_mem_descriptor	read only
old_activity_id	osak_mem_descriptor	read only
sync_point	osak_sync_point	read only
[old_sconnection_id]	osak_sconnection_id	read only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

osak_act_resume_req (port, parameter_block)

osak_port port;

```
struct osak_parameter_block *parameter_block;
```

Parameters Used

activity_id A structure in which you specify the identifier of the activity you want to resume. The identifier can be any string, with a maximum length of six characters. **old_activity_id** A structure in which you specify the identifier of the interrupted activity. The identifier can be any string, with a maximum length of six characters. **sync_point** The address of the serial number of the synchronization point at which you wish to resume the interrupted activity. **old_sconnection_id** The address of a structure you can use to specify session connection information for the interrupted activity. There are size restrictions on the values in each field, as follows:

Field	Maximum Size
called_ss_user_id	64 octets
calling_ss_user_id	64 octets
common_ref	64 octets
add_ref_info	4 octets

A null value in any of the fields of the structure implies omission of the parameter. If you omit the parameter, no session connection information is transferred. **token_item** The address of a structure you can use to specify the tokens that the requester is passing to the accepter. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the requester is not passing this token to the accepter.

- One means that the requester is passing this token to the accepter.

Description

You can use this service only if the activity management functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVID	The activity identifier is too long.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.

OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_interrupt_req

osak_act_start_req

osak_act_start_req — Starts an activity within an association.

Syntax

status = osak_act_start_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
activity_id	osak_mem_descriptor	read only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_act_start_req (port, parameter_block,)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

activity_id

In this structure, you specify an identifier for the activity you are starting. The identifier can be any string and can have a maximum length of six characters.

token_item

The address of a structure you can use to specify the tokens that the requester is passing to the accepter. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the requester is not passing this token to the accepter.

- One means that the requester is passing this token to the accepter.

Description

You can use this service only if the activity management functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

osak_alter_req

osak_alter_req — Requests alterations to the defined context set.

Syntax

```
status = osak_alter_req (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[pcontext_list][1]	osak_pcontext_proposal	read only

Parameters Used	Data Type	Access
[pcontext_del_list][1]	osak_pcontext_deletion	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

[1]You should specify one or both of these parameters; you cannot call the routine without specifying at least one parameter.

C Binding

```
osak_alter_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

pcontext_list

The address of the head of a linked list of structures, each of which proposes a presentation context. In each structure, the following rules apply:

- The *next* field can be null, indicating the end of the list. Otherwise, this field points to the next structure in the list.

- The *pcontext_id* field should contain a descriptor of an ASN.1 integer TLV string specifying the presentation context identifier.

- The *tslist* field should contain the address of a valid transfer syntax name that contains at least one element. Each element in the list is a structure of the type osak_ts_list and specifies an ASN.1 object identifier TLV encoding for the transfer syntax that you are proposing.

- The *as_name* field should contain a descriptor for the ASN.1 object identifier TLV encoding of the abstract syntax that you are proposing.

pcontext_del_list

The address of the head of a linked list of structures, each of which proposes the deletion of one presentation context from the defined context set. In each structure, the following rules apply:

- The *next* field can contain the value zero, indicating the end of the list. Otherwise, this field points to the next structure in the list.

- The *pcontext_id* field should contain a descriptor of an ASN.1 integer TLV string specifying the presentation context identifier.

Description

You can use this service only if the context management functional unit is selected.

The *pcontext_list* parameter contains a set of presentation contexts proposed for addition to the defined context set (DCS). The *pcontext_del_list* parameter contains a set of presentation contexts proposed for deletion from the DCS. You should use one or both of these parameters when you call `osak_alter_req`.

The OSAK interface does not allow you to delete the presentation context for ACSE abstract syntax.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_alter_rsp

osak_alter_rsp

osak_alter_rsp — Responds to a request for alterations to the defined context set.

Syntax

status = osak_alter_rsp (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[pcontext_res_list]	osak_pcontext_proposal_result	read only
[pcontext_del_res_list]	osak_pcontext_deletion_result	read only

Parameters Used	Data Type	Access
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_alter_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

pcontext_res_list

The address of the head of a linked list of structures. Each structure specifies the response to the corresponding entry in the list of proposed additions to the defined context set (DCS) sent on the P-ALTER-CONTEXT indication to which you are responding. Section 10.13 gives the possible values this parameter can have.

pcontext_del_res_list

The address of the head of a linked list of structures. Each structure specifies the response to the corresponding entry in the list of proposed deletions from the DCS sent on the P-ALTER-CONTEXT indication to which you are responding.

Description

You can use this service only if the context management functional unit is selected.

Call this routine after receiving a P-ALTER-CONTEXT indication. Use the routine to accept or reject the additions and deletions to the defined context set on an association proposed in the P-ALTER-CONTEXT indication.

The OSAK interface does not allow you to accept the deletion of the ACSE abstract syntax.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.

OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_alter_req

osak_associate_req

osak_associate_req — Establishes an association.

Syntax

status = osak_associate_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
acontext	osak_mem_descriptor	read only
called_aei	osak_aei	read only
[calling_aei]	osak_aei	read only
[transport_template]	osak_transport_templates	read only
[protocol_versions]	osak_protocol_versions	read only
[sconnect_id]	osak_sconnect_id	read only
[segmentation]	osak_segmentation	read only
[initial_serial_number][1]	osak_sync_point	read only
[initial_tokens]	osak_token_setting	read only
[functional_units]	osak_fus	read only
pcontext_list	osak_pcontext_proposal	read only
[pdefault_context]	osak_default_context	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only

Parameters Used	Data Type	Access
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

[1]This parameter is mandatory in some situations and optional in others; see the description in the Parameters Used section.

C Binding

```
osak_associate_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

acontext

The address of a structure in which you should specify the address of the encoded value of an ASN.1 object identifier for the application context name. This parameter is mandatory, but it can be null. The OSAK interface does not supply a default.

called_aei

The address of a structure that you should use to specify the peer entity with which you want to set up an association. See Section 1.3 for a description of the data type. If you specify an invalid selector in any field, the OSAK interface returns the status code OSAK_S_INVAEI.

The session selector should be no longer than 16 octets. The transport selector can be null.

Both the network address and the network protocol should be specified in the *nsap* field of the *p-address* structure. The *nsap* field can contain a list of network addresses and network protocols. The list can include both TCP/IP addresses and OSI addresses.

calling_aei

The address of a structure in which you can specify the title of your application entity. See Section 1.3 for a description of the data type.

transport_template

The address of a structure you can use to specify a list of transport templates that gives information about the transport requirements of an application. If you do not specify a template, the OSAK interface uses a default template based on the network protocol type. For CLNS and CONS, the default template is *default*. For RFC 1006, the default template is *osit$rfc1006* on OpenVMS and the pseudo-template 1006 on UNIX. Refer to your network management documentation for further information on the OSI transport module.

protocol_versions

The address of a structure you can use to specify which protocol versions are required on the association. The structure has three fields:

- *acse_version*

- *pversion*

- *sversion*

If any of the fields contains the value zero, the OSAK interface uses the default version number for that protocol. If the parameter is null, the OSAK interface uses the default version numbers for all three protocols. Table 1.2 shows the defaults.

Table 1.2. osak_associate_req: Default Protocol Version Numbers

Protocol	Default Version Number
ACSE	1
Presentation	1
Session	1 and 2

The two session default values are not mutually exclusive. **sconnect_id** The address of a structure you can use to specify the session connection reference parameters. The structure contains three substructures:

- *ss_user_ref*

- *common_ref*

- *add_ref_info*

Any of the fields can be null. **segmentation** The address of a structure you can use to specify in which direction data is to be segmented. The structure contains two fields:

- *init_resp*

- *resp_init*

A value other than zero in the *init_resp* field indicates that segmentation is to be used on data passing from the initiator to the responder. The value specifies the maximum TSDU size.

A value other than zero in the *resp_init* field indicates that segmentation is to be used on data passing from the responder to the initiator. The value specifies the maximum TSDU size.

The maximum value allowed in either field is 65,535.

You can use segmentation in both directions, in only one direction, or in neither direction. If this parameter is null, OSAK uses the default of unlimited TSDU size for both directions.

initial_serial_number

The address of the serial number of the initial synchronization point on the association.

This parameter is mandatory if both the following conditions are true:

- The major synchronize, the minor synchronize, or the resynchronize functional unit is selected.

- The activity management functional unit is not selected.

initial_tokens

The address of a structure you can use to specify either the initial token settings for the association, or to specify that the responder should choose the settings.

If this parameter is null, the OSAK interface uses a default setting; all the available tokens are assigned to the initiator. Section 10.15 gives the possible values this parameter can have.

functional_units

The address of a structure you can use to specify the presentation and session functional units that you require. All the fields in the structure are of the type bit field mask. If you make the address null, the OSAK interface uses the following default set of session functional units, in addition to the kernel functional unit at each layer:

- Activity management functional unit

- Capability functional unit

- Exceptions functional unit

- Half-duplex functional unit

- Minor synchronize functional unit

If you do not make the address null, but you set all the functional unit bit fields to zero, the OSAK interface assumes that you are not selecting any functional units, and returns status OSAK_S_INVFUS. You cannot set up an association without any functional units. You should specify at least one of duplex and half-duplex.

If you make the value of the presentation functional unit bit zero, the OSAK interface assumes that no presentation functional unit except the kernel functional unit is required.

OpenVMS

To specify data separation, select the data separation and minor synchronize functional units, but not the activity management functional unit.

pcontext_list

The address of the head of a linked list of structures, each of which specifies a proposed presentation context.

The context list should include a proposal for the ACSE abstract syntax. The OSAK interface checks that the ACSE syntax is correctly proposed. If it is not correctly proposed, the OSAK interface returns status OSAK_S_INVPCTXT.

You should propose at least two presentation contexts, for example, the ACSE abstract syntax and one other. ISO Standard 8824 explains how you can use ASN.1 to define your own abstract syntax.

In each structure in the linked list, the following rules apply:

- The *next* field can contain the value zero, indicating the end of the list.

- The *pcontext_id* field should contain a descriptor of an ASN.1 integer TLV string specifying the presentation context identifier. The integer value should be odd.

- The *ts_list* field should contain the address of a valid transfer syntax name that contains at least one element. Each element in the list is a structure of the type *osak_ts_list* and specifies an ASN.1 object identifier TLV encoding for the transfer syntax that you are proposing.

- The *as_name* field should contain a descriptor for the ASN.1 object identifier TLV encoding of the abstract syntax that you are proposing.

pdefault_context

The address of a structure that you can use to propose a default context for a connection. If this parameter is not null, a transfer syntax name and an abstract syntax name should be specified in the *ts_list* field and the *as_name* field, respectively. The transfer syntax name and the abstract syntax name should be ASN.1 object identifier encodings.

Description

Call this routine after calling `osak_open_initiator`, using the port identifier returned by `osak_open_initiator`.

You can send a first segment of data with `osak_associate_req` and send the rest on multiple calls to `osak_send_more`. The first segment should include all the session, presentation, and ACSE PCI, and it can include part of the user data. Alternatively, you can send all the PCI and all the user data with `osak_associate_req`.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVAEI	The application entity invocation is invalid.
OSAK_S_INVDEFCTXT	The default context response is invalid.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVFUS	The functional units are invalid.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_INVTEMPLATE	Transport template is unknown.
OSAK_S_NOSYNCPNT	The synchronization point serial number is missing.
OSAK_S_NOCTXTNAME	The application context name is missing.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_accept_rsp

osak_open_initiator
osak_open_responder
osak_reject_rsp

osak_async_close

osak_async_close — Closes down a specified port from AST level and reclaims memory controlled by OSAK. This is available only on OpenVMS systems.

Syntax

status = osak_async_close (port, parameter_block, destructive_flag)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only
destructive_flag	Unsigned octet	read only

Parameters Used	Data Type	Access
completion_rtn	osak_rtn	read only
completion_param	Longword	read only
next_pb	osak_parameter_block	write only
tsdu_ptr	osak_buffer	write only

C Binding

```
osak_async_close (port, parameter_block, destructive_flag)

osak_port port;
struct osak_parameter_block *parameter_block;
unsigned char destructive_flag;
```

Arguments

destructive_flag

A flag you can use to indicate how you want the OSAK interface to close the port. The flag can have either of the following values:

• OSAK_C_DESTRUCTIVE

• OSAK_C_NON_DESTRUCTIVE

If you set the value to OSAK_C_DESTRUCTIVE, OSAK closes the port and disconnects the transport connection no matter what state the association is in.

If you set the value to OSAK_C_NON_DESTRUCTIVE, OSAK closes the port only when the association has been terminated. If the association is still active, the OSAK interface will return OSAK_S_INVFUNC.

Parameters Used

tsdu_ptr

The address of the head of a linked list of user buffers. The OSAK interface returns the unused buffers that the application passed in calls to `osak_give_buffers`. The OSAK interface makes this parameter null if there are no buffers to return.

next_pb

The address of the head of a linked list of parameter blocks. The OSAK interface returns the parameter blocks that have been passed in outbound calls during the association and have not already been collected.

Description

Call this routine after you terminate an association by aborting or releasing it or after you redirect an association. This should be used only if your application needs to perform this function at asynchronous system trap (AST) level, otherwise you should use `osak_close_port`.

To close a port for a peer entity that receives an A-RELEASE indication, you should do the following:

*   Call `osak_release_rsp`

*   Set up a timer and wait for the arrival of the transport event indicating that the transport connection is disconnected. The event you should wait for is OSAK_C_TDIS. The recommended waiting time is 30 seconds.

*   If the transport event arrives before the timer expires, call `osak_async_close` with the *destructive_flag* parameter set to OSAK_C_NON_DESTRUCTIVE. If the transport event does not arrive before the timer expires, call `osak_async_port` with the *destructive_flag* parameter set to OSAK_C_DESTRUCTIVE.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.

See Also

osak_abort_req
osak_close_port
osak_open_redirect
osak_redirect
osak_release_req
osak_release_rsp

osak_capability_req

osak_capability_req — Transfers capability data over an association.

_____

Syntax

status = osak_capability_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
user_data	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_capability_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Description

You can use capability data only under the following conditions:

- The activity management and capability functional units should be selected.

- You must have the major activity token.

- You must have the data token and minor-synchronize token if they are available.

- There should be no activity in progress on the association.

- At least one byte of data must be sent if the *more_flag* parameter is set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.

OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.
OSAK_S_NODATA	No data has been specified in the call.

See Also

osak_capability_rsp

osak_capability_rsp

osak_capability_rsp — Responds to a request to transfer capability data.

Syntax

status = osak_capability_rsp (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_capability_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

token_item

The address of a structure you can use to specify the tokens that the accepter wants from the requester. The structure consists of four fields corresponding to the four tokens.

In each field, the only values allowed are zero and one:

- Zero means that the accepter does not want this token from the requester.

- One means that the accepter wants this token from the requester.

Description

Call this routine after receiving a P-CAPABILITY-DATA indication.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_capability_req

osak_close_port

osak_close_port — Closes down a specified port and reclaims memory controlled by OSAK.

Syntax

```
status = osak_close_port (port, rcv_buffer_list, parameter_block, destructive_flag
```

Argument	Data Type	Access
port	osak_port	read only
rcv_buffer_list	osak_buffer	write only

Argument	Data Type	Access
parameter_block	osak_parameter_block	write only
destructive_flag	unsigned octet	read only

C Binding

```
osak_close_port (port, rcv_buffer_list, parameter_block, destructive_flag)

osak_port port;
struct osak_buffer **rcv_buffer_list;
struct osak_parameter_block **parameter_block;
unsigned char destructive_flag;
```

Arguments

rcv_buffer_list

The address of the head of a linked list of user buffers. The OSAK interface returns the unused buffers that the application passed in calls to `osak_give_buffers`. The OSAK interface makes this parameter null if there are no buffers to return.

parameter_block

The address of the head of a linked list of parameter blocks. The OSAK interface returns the parameter blocks that have been passed in outbound calls during the association and have not already been collected.

destructive_flag

A flag you can use to indicate how you want the OSAK interface to close the port. The flag can have either of the following values:

• OSAK_C_DESTRUCTIVE

• OSAK_C_NON_DESTRUCTIVE

If you set the value to OSAK_C_DESTRUCTIVE, OSAK closes the port and disconnects the transport connection no matter what state the association is in.

If you set the value to OSAK_C_NON_DESTRUCTIVE, OSAK closes the port only when the association has been terminated. If the association is still active, the OSAK interface will return OSAK_S_INVFUNC.

Description

Call this routine after you terminate an association by aborting or releasing it or after you redirect an association.

To close a port for a peer entity that receives an A-RELEASE indication, you should do the following:

• Call `osak_release_rsp`.

• Set up a timer and wait for the arrival of the transport event indicating that the transport connection is disconnected. The event you should wait for is OSAK_C_TDIS. The recommended waiting time is 30 seconds.

- If the transport event arrives before the timer expires, call `osak_close_port` with the *destructive_flag* parameter set to OSAK_C_NON_DESTRUCTIVE. If the transport event does not arrive before the timer expires, call `osak_close_port` with the *destructive_flag* parameter set to OSAK_C_DESTRUCTIVE.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.

See Also

osak_abort_req
osak_async_close
osak_open_redirect
osak_redirect
osak_release_req
osak_release_rsp

osak_collect_pb

osak_collect_pb — Checks for the completion of outbound services.

Syntax

status = osak_collect_pb (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	write only

C Binding

```
osak_collect_pb (port, parameter_block)

osak_port port;
struct osak_parameter_block **parameter_block;
```

Arguments

port

The port from which you want to collect any available parameter blocks.

parameter_block

The address of the head of a linked list of parameter blocks. The OSAK interface returns any free parameter blocks that it is holding.

If there are no free parameter blocks, the OSAK interface returns a null address in this parameter.

Description

The routine checks for the completion of outbound services on the specified port. The OSAK interface returns the addresses of any parameter blocks and user buffers that you passed in outbound calls and that are now free for you to reuse.

You can examine the *func* parameter to determine the service on which the parameter block was used. If you are using the *user_context* parameter to keep track of parameter blocks and user buffers, you can determine the call on which the parameter block was used. The service codes are defined in the OSAK include file *osak_api_codes.h*.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_NORMAL	The routine has finished without error and parameter blocks have been retrieved.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_NOEVENT	No parameter blocks to return to the user.

osak_control_give_req

osak_control_give_req — Relinquishes ownership of all available tokens.

Syntax

```
status = osak_control_give_req (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_control_give_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Description

You can use this service only if the activity management functional unit is selected. There should be no activity in progress on the association when you use the service. You must have the major activity token and all other available tokens.

If you are using session version 1, there is no user data on this service and therefore no segmentation is allowed and the *more_flag* parameter must be set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_token_please_req
osak_token_give_req

osak_data_req

osak_data_req — Transfers user information over an association.

Syntax

```
status = osak_data_req (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_data_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameter Used

token_item

The address of a structure you can use to specify the tokens that the requester is passing to the accepter. The structure consists of four fields corresponding to the four tokens.

In each field, the only values allowed are zero and one:

- Zero means that the requester is not passing this token to the accepter.

- One means that the requester is passing this token to the accepter.

Description

You can use this service to send user information under normal circumstances. At least one byte of data must be sent in this call if the *more_flag* parameter is set to false. You must have the data token if the half-duplex functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	the OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.

OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_TRANSERR	There is an error in the transport provider.
OSAK_S_NODATA	No data has been specified in the call.

See Also

osak_send_more

osak_exception_req

osak_exception_req — Signals error conditions that are not serious enough to cause termination of an association.

Syntax

`status = osak_exception_req (port, parameter_block)`

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
exception_reason	osak_exception_reason	read only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_exception_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

exception_reason

Use this parameter to specify the reason for the exception report. Section 10.5 lists the possible values of this parameter.

token_item

The address of a structure you can use to specify the tokens that the requester wants from the accepter. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the requester does not want this token from the accepter.

- One means that the requester wants this token from the accepter.

Description

You can use this service only if the half-duplex functional unit is selected. You must have the data token.

If used with the activity management service, the exception-reporting service is only permitted while an activity is in progress.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

osak_expedited_req

osak_expedited_req — Transfers expedited data over an association.

Syntax

```
status = osak_expedited_req (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
user_data	osak_buffer	read only

C Binding

```
osak_expedited_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Description

You can use this service only if the expedited functional unit is selected.

No segmentation of user data is allowed. The maximum amount of user data you can send on the service is 14 octets.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

osak_get_event

osak_get_event — Receives an event from a specified association.

Syntax

status = osak_get_event (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
tsdu_ptr	osak_buffer	write only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
event_type	osak_event	write only
[peer_data]	osak_buffer	write only
[acse_pci_eoc]	Unsigned long integer	write only
[pres_pci_eoc]	Unsigned long integer	write only
[user_context]	Address	read only
more_flag	Long integer	write only
[data_length]	Unsigned longword	write only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_get_event (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

tsdu_ptr

The address of the head of a user buffer list. The buffers are those you passed to OSAK in calls to `osak_give_buffers`.

To free the buffers when they are returned by `osak_get_event`, you should follow the *tsdu_ptr* pointer, not the *peer_data* pointer.

event_type

In this parameter, the OSAK interface returns the type of a newly arrived event. Table 1.3 lists the values that can occur in the *event_type* parameter, and their corresponding event types.

If you call `osak_get_event` before all the user data has been received but when at least one buffer has been filled, the OSAK interface may segment the event. The *more_flag* paramter will be set to true and the *event_type* will indicate which event was received. On subsequent calls, until the complete event is received, the *event_type* will be OSAK_C_CONTINUE. The *more_flag* parameter will be true on all calls except the last.

When the return value of a call to `osak_get_event` is OSAK_S_NOEVENT, the value in the *event_type* parameter is OSAK_C_NOEVENT. If you are using a completion routine, you should check for the value OSAK_C_NOEVENT in the *event_type* parameter (OpenVMS systems only).

peer_data

The address of a linked list of zero, one, or more user buffers containing segments of encoded user data received from the remote peer entity.

This parameter points to the position in the list of buffers where the user data starts. The parameter *tsdu_ptr* points to the head of the list of buffers.

acse_pci_eoc

Indicates the number of end-of-contents octets in the data arriving from the remote peer entity.

To arrive at the value, the OSAK interface counts the number of indefinite length encodings in the ACSE PCI for which it cannot find end-of-contents octets. You should look for this number of end-of-contents octets in the ACSE PCI. If you do not find this number, you should issue a presentation provider abort.

An end-of-contents octet consists of two zero octets.

pres_pci_eoc

Indicates the number of end-of-contents octets in the data arriving from the remote peer entity.

To arrive at the value, the OSAK interface counts the number of indefinite length encodings in the presentation PCI for which it cannot find end-of-contents octets. You should look for this number of end-of-contents octets in the presentation PCI. If you do not find this number, you should issue a provider abort.

An end-of-contents octet consists of two zero octets.

Description

To receive an event, you should first pass user buffers to OSAK. Use the `osak_give_buffers` routine to do this. The OSAK interface uses the buffers to receive and store incoming data units. Chapter 2 describes each of the incoming events that can occur.

When an event arrives, the OSAK interface extracts the user data and the PCI from the data units and writes them into the parameters supplied in the call to `osak_get_event`. With the exception of the *peer_data* parameter, the parameters supplied in the call to `osak_get_event` point to values contained in the PCI. The *peer_data* parameter points to the user data. *VSI DECnet-Plus OSAK Programming* describes the structure of user buffers for a call to `osak_get_event`, and how to use the buffers.

Note

The routine `osak_get_event` can write values in almost every parameter in the parameter block. If you call this routine with a parameter block that already contains values, any of these values can be overwritten.

If a buffer contains only PCI, its *data_ptr* field is a null pointer. If a buffer contains user data, or a mixture of PCI and user data, its *data_ptr* field points to the beginning of the user data.

If the *more_flag* parameter on an event is set to true, this means that the incoming user data is segmented. Make additional calls to osak_get_event to receive all the user data, until the *more_flag* parameter is set to false.

If you have not supplied sufficient buffers to receive the complete events, the OSAK interface returns OSAK_S_NOBUFFERS. If the event is segmented, this can occur on the first or any subsequent osak_get_event call. This is not a fatal error; if you post additional buffers, the OSAK interface will continue to receive the event and no data will be lost.

During an osak_get_event call, the OSAK interface can set parameters in the parameter block to point to the values received. The actual values are either in the TSDU or in the workspace provided. You should not delete these pointers, but you may set them to zero.

When an event arrives, the OSAK interface returns a value identifying that event in the *event_type* parameter. Table 1.3 lists these values and the event types that they represent.

Table 1.3. OSAK Event Types

Value in event_type Parameter	**Event Type Indicated**
OSAK_C_ABORT_IND	ABORT indication
OSAK_C_ACCEPT_CNF	A-ASSOCIATE accept confirm
OSAK_C_ACT_DISCARD_CNF	P-ACTIVITY-DISCARD confirm
OSAK_C_ACT_DISCARD_IND	P-ACTIVITY-DISCARD indication
OSAK_C_ACT_END_CNF	P-ACTIVITY-END confirm
OSAK_C_ACT_END_IND	P-ACTIVITY-END indication
OSAK_C_ACT_INTERRUPT_CNF	P-ACTIVITY-INTERRUPT confirm
OSAK_C_ACT_INTERRUPT_IND	P-ACTIVITY-INTERRUPT indication
OSAK_C_ACT_RESUME_IND	P-ACTIVITY-RESUME indication
OSAK_C_ACT_START_IND	P-ACTIVITY-START indication
OSAK_C_ALTER_CONTEXT_IND	P-ALTER-CONTEXT indication
OSAK_C_ALTER_CONTEXT_CNF	P-ALTER-CONTEXT confirm
OSAK_C_ASSOC_IND	A-ASSOCIATE indication
OSAK_C_CAPABILITY_CNF	P-CAPABILITY-DATA confirm
OSAK_C_CAPABILITY_IND	P-CAPABILITY-DATA indication
OSAK_C_CONTINUE	Continuation event when segmentation is in use
OSAK_C_CONTROL_GIVE_IND	P-CONTROL-GIVE indication
OSAK_C_DATA_IND	P-DATA indication
OSAK_C_EXCEPTION_IND	P-U-EXCEPTION indication
OSAK_C_EXPEDITED_IND	P-EXPEDITED-DATA indication
OSAK_C_EXCEPTION_IND	P-P-EXCEPTION indication
OSAK_C_NOEVENT	Returned when the status code of the call to osak_get_event is OSAK_S_NOEVENT
OSAK_C_PLEASE_IND	P-TOKEN-PLEASE indication
OSAK_C_REDIRECT_IND	Redirect indication
OSAK_C_REJECT_CNF	A-ASSOCIATE reject confirm

Value in event_type Parameter	Event Type Indicated
OSAK_C_RELEASE_CNF	P-RELEASE confirm
OSAK_C_RELEASE_IND	P-RELEASE indication
OSAK_C_RESYNC_CNF	P-RESYNCHRONIZE confirm
OSAK_C_RESYNC_IND	P-RESYNCHRONIZE indication
OSAK_C_SYNC_MAJOR_CNF	P-SYNC-MAJOR confirm
OSAK_C_SYNC_MAJOR_IND	P-SYNC-MAJOR indication
OSAK_C_SYNC_MINOR_CNF	P-SYNC-MINOR confirm
OSAK_C_SYNC_MINOR_IND	P-SYNC-MINOR indication
OSAK_C_TDISCONNECT	Transport disconnect indication
OSAK_C_TOKEN_GIVE_IND	P-TOKEN-GIVE indication
OSAK_C_TYPED_DATA_IND	P-TYPED-DATA indication

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_NOEVENT	No event has occurred.
OSAK_S_QUEUED	The OSAK interface has queued the request (returned only if the call includes a completion routine).
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INCPCI	The PCI is not complete.
OSAK_S_INSFWS	There is not enough workspace in the parameter block.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_NOBUFFERS	There are not enough user data buffers.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_give_buffers
osak_select

osak_get_handle

osak_get_handle — Returns the transport connection handle for an association.

Syntax

```
status = osak_get_handle (port, handle)
```

Argument	Data Type	Access
port	osak_port	read only
handle	Longword	write only

C Binding

```
osak_get_handle (port, handle)

osak_port port;
longword *handle;
```

Argument

handle

In this argument, the OSAK interface returns the address of the external handle for the association specified in the *port* argument.

On OpenVMS systems, *handle* is an event flag number (EFN). On UNIX and ULTRIX systems, *handle* is a file descriptor.

Description

OpenVMS

This is a dummy routine. In the *handle* argument, the routine returns the port identifier that you pass in the *port* argument.

UNIX

The routine returns the file descriptor returned by the transport interface that you use to set up a transport connection. You can use this file descriptor in a call to `osak_select` as an alternative to using the *port* identifier returned by `osak_open_initiator`, `osak_open_responder`, or `osak_open_redirect`.

If you are acting as responder, note that the file descriptor returned by the OSAK interface before a transport connection is not the same as that returned after a connection is established. This is because OSAK uses one descriptor for listening for inbound connections and another for accepting the connections. For this reason, unless you need to handle the file descriptor directly, VSI recommends you use the *port* identifier. This also allows the OSAK interface to perform additional checks.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_NOTRANSPORT	No transport connection set up yet.

See Also

osak_select

osak_give_buffers

osak_give_buffers — Passes a list of user data buffers to the OSAK interface for receiving and storing events.

Syntax

status = osak_give_buffers (port, rcv_buffer_list)

Argument	Data Type	Access
port	osak_port	read only
rcv_buffer_list	osak_buffer	read only

C Binding

```
osak_give_buffers (port, rcv_buffer_list,)

osak_port port;
struct osak_buffer *rcv_buffer_list;
```

Argument

rcv_buffer_list

The address of the head of a linked list of user buffers. The OSAK interface uses the buffers to receive and store incoming events.

Description

This routine supplies buffers that the OSAK interface passes to the transport interface. The transport interface fills the buffers with segments of incoming TSDUs, and passes them back to the OSAK interface. The size of each buffer must not be less that the number of bytes defined by OSAK_C_MINIMUM_RCV_BUFFER.

Unused buffers are returned in the osak_close_port call.

The description of the routine osak_get_event gives further information on the structure of these buffers.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_INVPARAM	The size of one or more of the buffers is less than the minimum.
OSAK_S_INVPORT	The port identifier is invalid.

See Also

osak_get_event
osak_close_port

osak_major_req

osak_major_req — Requests the setting of a major synchronization point.

Syntax

status = osak_major_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
sync_point	osak_sync_point	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_major_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

sync_point

The address of the current major synchronization point. OSAK increments the serial number of the major synchronization point and writes the result to this parameter. OSAK does this when the major synchronization service completes, so you should not use the value of this parameter if the call returns with status OSAK_S_QUEUED.

token_item

The address of a structure you can use to specify the tokens that the requester is passing to the accepter. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the requester is not passing this token to the accepter.

- One means that the requester is passing this token to the accepter.

Description

You can use this service only if the major synchronize functional unit is selected. You must have the major activity token. You must also have the minor synchronize token and data token if they are available.

If the activity management functional unit is selected, the service can only be initiated within an activity.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_act_end_req
osak_major_rsp
osak_minor_req
osak_resync_req

osak_major_rsp

osak_major_rsp — Responds to a request to set a major synchronization point.

Syntax

```
status = osak_major_rsp (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_major_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

token_item

The address of a structure you can use to specify the tokens that the accepter wants from the requester. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

• Zero means that the accepter does not want this token from the requester.

• One means that the accepter wants this token from the requester.

Description

Call this routine after receiving a P-SYNC-MAJOR indication.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.

OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_major_req

osak_minor_req

osak_minor_req — Requests the setting of a minor synchronization point.

Syntax

status = osak_minor_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
sync_point	osak_sync_point	write only
sync_confirm	osak_sync_confirm	read only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_minor_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

sync_point

The address of the current minor synchronization point. OSAK increments the serial number of the minor synchronization point and writes the result to this parameter. OSAK does this when the minor synchronization service completes, so you should not use the value of this parameter if the call returns with status OSAK_S_QUEUED.

sync_confirm

If you set the value of this parameter to true, the requester is asking for explicit confirmation from the accepter that the synchronization point has been set. If you set the value to false, the accepter can send explicit confirmation, but it does not have to do so.

data_separation (OpenVMS only)

Indicates whether data separation is required. Set this parameter to true if you select the data separation functional unit. If you do not select the data separation session functional unit, the OSAK interface ignores this parameter.

token_item

The address of a structure you can use to specify the tokens that the requester is passing to the accepter. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the requester is not passing this token to the accepter

- One means that the requester is passing this token to the accepter.

Description

You can use this service only if the minor synchronize functional unit is selected. You must have the minor activity token and data token if they are available.

If the activity management functional unit is selected, this service can only be initiated within an activity.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.

OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_major_req
osak_minor_rsp
osak_resync_req

osak_minor_rsp

osak_minor_rsp — Responds to a request to set a minor synchronization point.

Syntax

status = osak_minor_rsp (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
sync_point	osak_sync_point	read only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_minor_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

sync_point

The address of the value of the synchronization point that the accepter is acknowledging. This value should be greater than the value of the last acknowledged synchronization point, but less than or equal to the value of the most recently requested synchronization point.

token_item

The address of a structure you can use to specify the tokens that the accepter wants from the requester. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

• Zero means that the accepter does not want this token from the requester.

• One means that the accepter wants this token from the requester.

Description

Call this routine after receiving a P-SYNC-MINOR indication.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_minor_req

osak_open_initiator

osak_open_initiator — Allocates a port for use in a subsequent request to establish an association.

Syntax

```
status = osak_open_initiator (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	write only

Argument	Data Type	Access
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
api_version	Unsigned long integer	read only
[local_aei]	osak_aei	read only
alloc_rtn	osak_rtn	read only
dealloc_rtn	osak_rtn	read only
[alloc_param]	Unsigned integer	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_open_initiator (port, parameter_block)

osak_port *port;
struct osak_parameter_block *parameter_block;
```

Argument

port

In this parameter, the OSAK interface returns an identifier for the access point between itself and the initiator.

Parameters Used

local_aei

The address of a structure specifying the initiator's application-entity invocation. See Section 1.3 for a description of the data type.

Any of the fields of this structure can be null.

api_version

The version of the OSAK interface being used.

You should specify the following constant in this parameter: OSAK_C_API_VERSION_3. If you specify any other value, the OSAK interface returns the status OSAK_S_INVAPIVERSION.

Description

This routine returns a port identifier that you then use in a call to `osak_associate_req`.

You should supply the memory allocation and deallocation routines for the OSAK interface to use.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFWS	There is not enough workspace in the parameter block.
OSAK_S_INVTEMPLATE	Transport template is unknown.

See Also

osak_accept_rsp
osak_associate_req
osak_open_responder
osak_reject_rsp

osak_open_redirect

osak_open_redirect — Allocates a port for use on a redirected association. For a passive application (OpenVMS systems only), opens a responder.

Syntax

status = osak_open_redirect (port, parameter_block)

Argument	Data Type	Access
port	osak_port	write only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
api_version	Unsigned long integer	read only
alloc_rtn	osak_rtn	read only
dealloc_rtn	osak_rtn	read only
[alloc_param]	unsigned integer	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

osak_open_redirect (port, parameter_block)

```
osak_port *port;
struct osak_parameter_block *parameter_block;)
```

Description

This routine allocates a new port for a redirected association. Provided the routine returns a success status, you can make further routine calls using this port.

OpenVMS

> For a passive application, use this routine with the *local_aei* parameter set to open a responder. Use the routine with the *local_aei* parameter set to null for an active application waiting for a redirect.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_INSFWS	There is not enough workspace in the parameter block.
OSAK_S_INVAEI	The application entity invocation is invalid.
OSAK_S_NOSERVER	There is no response from OSAK server.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_close_port
osak_redirect

osak_open_responder

osak_open_responder — Opens a port for use by a responder.

Syntax

status = osak_open_responder (port, parameter_block)

Argument	Data Type	Access
port	osak_port	write only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[transport_template]	osak_transport_templates	read only

Parameters Used	Data Type	Access
api_version	Unsigned long integer	read only
local_aei (UNIX and ULTRIX systems)	osak_aei	read only
[local_aei] (OpenVMS systems)	osak_aei	read only
[protocol_versions]	osak_protocol_versions	read only
alloc_rtn	osak_rtn	read only
dealloc_rtn	osak_rtn	read only
[alloc_param]	Unsigned integer	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_open_responder (port, parameter_block)

osak_port *port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

transport_template

UNIX

The address of a structure you can use to specify a transport template that gives information about the transport requirements of an application.

For inbound connections, specify an OSI transport template. One template only is supported. The OSAK software accepts the network types CLNS, CONS, and ANY. Note that the same port cannot wait for connections over both TCP/IP and OSI transport. To do this, you must open more than one responder.

If you do not specify a template, the OSAK interface uses a default template called *default*. Refer to your network management documentation for further information on the OSI transport module.

OpenVMS

This is ignored on OpenVMS systems, but can be included if necessary for compatibility.

api_version

The version of the OSAK interface being used.

Specify the following constant in this parameter: OSAK_C_API_VERSION_3. If you specify any other value, the OSAK interface returns the status OSAK_S_INVAPIVERSION.

local_aei

The address of a structure specifying the responder's application entity invocation. See Section 1.3 for a description of the data type.

If you are using the OSAK interface on an OpenVMS system and you omit this parameter, the OSAK interface supplies default null values for all the fields.

protocol_versions

The address of a structure that you can use to specify which protocol versions are required on the association. The structure has three fields:

- *acse_version*

- *pversion*

- *sversion*

If any of the fields contains the value zero, the OSAK interface uses the default version number for that protocol. If the parameter is null, the OSAK interface uses the default version numbers for all three protocols. Table 1.4 shows the defaults.

Table 1.4. osak_open_responder: Default Protocol Version Numbers

Protocol	**Default Version Number**
ACSE	1
Presentation	1
Session	1 and 2

The two session default values are not mutually exclusive.

OpenVMS

On OpenVMS systems, if OSAKserver receives an A-ASSOCIATE indication specifying incompatible protocol versions, OSAKserver rejects the association attempt.

Description

This routine allocates a port identifier that the OSAK interface uses in all service requests made on that port. The port is ready to receive association indications for a particular application-entity invocation. This is the first routine that a responder should call.

The routine also declares the responder's application-entity invocation.

The OSAK interface registers this information with OSAKserver (OpenVMS systems only).

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_INSFWS	There is not enough workspace in the parameter block.
OSAK_S_INVAEI	The application entity invocation is invalid.

OSAK_S_NOSERVER	There is no response from OSAKserver.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_accept_rsp
osak_associate_req
osak_open_initiator
osak_reject_rsp

osak_redirect

osak_redirect — Requests the redirection of an association from one process to another on the same system.

Syntax

status = osak_redirect (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[process_id]	osak_process_id	read only
[process_name]	osak_mem_descriptor	read only
[pcontext_list]	osak_pcontext_proposal	read only
[pcontext_redirect_list]	osak_pcontext	read only
[pdefault_context]	osak_default_context	read only
redirect_state	osak_state	read only
[calling_aei]	osak_aei	read only
[called_aei]	osak_aei	read only
acontext	osak_mem_descriptor	read only
[protocol_versions]	osak_protocol_versions	read only
[sconnect_id]	osak_sconnect_id	read only
[segmentation]	osak_segmentation	read only
[sync_point]	osak_sync_point	read only
[tokens]	osak_token_setting	read only
[activity_id]	osak_mem_descriptor	read only
functional_units	osak_fus	read only

Parameters Used	Data Type	Access
[alloc_param]	Unsigned integer	read only
[rcv_data_list]	osak_buffer	read only
[local_data]	osak_mem_descriptor	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_redirect (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

process_id

The address of the longword process identifier of the destination process. If you do not use this parameter, you should use the *process_name* parameter.

process_name

The address of a structure specifying the name of the destination process. If you do not use this parameter, you should use the *process_id* parameter.

pcontext_list

The address of the head of a linked list of structures, each one of which specifies one member of the DCS for the association. Set this parameter only if the process from which the association is to be redirected has not established the association when you call osak_redirect.

pcontext_redirect_list

The address of the head of a linked list of structures, each one of which specifies one member of the DCS for the association. Set this parameter only if the process from which the association is to be redirected has established the association when you call osak_redirect.

pdefault_context

The address of the structure that specifies the proposed default context for the redirected association.

redirect_state

A structure you should use to specify the state of the association when the requester makes the redirect request. This parameter consists of two fields:

- *initiator*

Set this field to true if the local entity is the initiator of the association, and to false if the local entity is the responder to the association.

- *pm_state*

  Set this field to the state of the association when the redirect call is made. Section 10.6 lists the possible values of this parameter.

calling_aei

The address of a structure you can use to specify the presentation address and title of your application entity. See Section 1.3 for a description of the data type.

called_aei

The address of a structure you can use to specify the presentation address and title of the peer entity with which you want to make a connection. See Section 1.3 for a description of the data type.

acontext

The address of a structure you should use to specify the address of the ASN.1 object identifier TLV for the application context name.

protocol_versions

The address of a structure you can use to specify which protocol versions are required on the association. The structure has three fields:

- *acse_version*

- *pversion*

- *sversion*

If any of the fields contains the value zero, the OSAK interface uses the default version number for that protocol. If the parameter is null, the OSAK interface uses the default version numbers for all three protocols. Table 1.5 shows the defaults.

Table 1.5. osak_redirect: Default Protocol Version Numbers

Protocol	Default Version Number
ACSE	1
Presentation	1
Session	1 and 2

The two session default values are not mutually exclusive.

sconnect_id

The address of a structure you can use to specify the session connection identifier. Make the address null if there is no session connection identifier.

segmentation

The address of a structure you can use to specify the direction data is to be segmented. The structure contains two fields:

- *init_resp*

- *resp_init*

A value other than zero in the *init_resp* field indicates that segmentation is to be used on data passing from the initiator to the responder. The value specifies the maximum TSDU size.

A value other than zero in the *resp_init* field indicates that segmentation is to be used on data passing from the responder to the initiator. The value specifies the maximum TSDU size.

The maximum value allowed in either field is 65,535.

You can use segmentation in both directions, in only one direction, or in neither direction.

sync_point

The address of the current synchronization point serial number.

tokens

The address of a structure you can use to specify the existing distribution of tokens. The value of the *state* parameter determines the interpretation of this parameter because it indicates whether the local entity is the initiator of or the responder to the association.

activity_id

A structure you can use to specify the identifier of any activity in progress on the association.

If there is no activity in progress, make this parameter null.

functional_units

The address of a structure you can use to specify the session and presentation functional units that are in use on the association.

rcv_data_list

The address of the linked list of user buffers that contains the user data for the service in progress. This parameter is relevant only if the requester makes the redirection request when the process is in one of the following states:

- The process has received an A-ASSOCIATE indication, but has not responded to it.

- The process has received an association indication with incomplete user data or no user data.

Refer to the Description section for details.

local_data

The address of a structure, the *descriptor* field of which holds the address of a buffer containing data that the requester wants to send to the destination process.

Description

An association should be in one of the following states before you can redirect it:

- The process has received an A-ASSOCIATE indication, but has not responded to it.

- The process has received an A-ASSOCIATE indication with incomplete user data or no user data.

- The process has established an association and is transferring data.

The routine `osak_redirect` passes a specified association to a specified destination process on the local system. If the association is successfully redirected, indicated by status codes OSAK_S_NORMAL, OSAK_S_FREE or OSAK_S_QUEUED, the port used by the original process is invalid. Call `osak_close_port` to close it down. Any other status code indicates that the association has not been successfully redirected, and the port used by the original process is still valid.

The destination process should call `osak_open_redirect`, which returns a new port identifier.

When you call `osak_redirect`, the OSAK interface should not hold any unused buffers passed from your application. If the interface holds any unused buffers, the routine returns the failure status OSAK_S_READPOSTED.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVAEI	The application entity invocation is invalid.
OSAK_S_INVDEFCTXT	The default context response is invalid.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVFUS	The functional units are invalid.
OSAK_S_INVID	The activity identifier is too long.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_NOPROCINFO	There is no process identifier and no process name.
OSAK_S_NOSYNCPNT	The synchronization point serial number is missing.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_close_port
osak_open_redirect

osak_reject_rsp

osak_reject_rsp — Rejects an association request.

Syntax

`status = osak_reject_rsp (port, parameter_block)`

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
reject_reason	osak_reject_reason	read only
[acontext]	osak_mem_descriptor	read only
[responding_aei]	osak_aei	read only
[sconnect_id]	osak_sconnect_id	read only
[functional_units]	osak_fus	read only
pcontext_res_list	osak_pcontext_proposal_result	read only
[pdefault_context_res]	osak_default_context_result	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_reject_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

reject_reason

The address of the reason for rejecting an association request. Section 10.9 lists the possible values of this parameter.

acontext

The address of a structure you should use to specify the address of the encoded value of an ASN.1 object identifier TLV for the application context name. If you do not assign a value to this parameter, the OSAK interface supplies the value that was received in the A-ASSOCIATE indication to which this

call is a response. If the association rejection originates from the service provider, the OSAK interface ignores this parameter.

responding_aei

The address of a structure you can use to specify the presentation selector of the responding application entity. See Section 1.3 for a description of the data type.

- The *nsap* field is ignored in this routine.

- The session selector should be no longer than 16 octets.

- The substructures *aetitle* and *aeiid* can both be null.

sconnect_id

The address of a structure you can use to specify session connection reference parameters. The structure contains three fields, any of which can be null. See Section 1.3 for a description of the data type.

functional_units

The address of a structure you can use to specify which session and presentation functional units are selected for use. If you use this parameter, you should specify both session and presentation functional units. If you do not set this parameter, the OSAK interface uses the selected functional units from the A-ASSOCIATE indication.

pcontext_res_list

The address of the head of a linked list of structures, each of which specifies the response to one entry in the list of proposed members of the defined context set (DCS) received in the A-ASSOCIATE indication. There should be a one-to-one correspondence between the responses and the proposed members of the DCS. The OSAK interface checks only that there is the same number of structures in each list.

The responder should accept the ACSE context. If the responder does not do this, the OSAK interface returns OSAK_S_INVPCTXT.

In the linked list, the following rules apply:

- The *next* field can contain the value zero, indicating the end of the list.

- The *ts_name* should contain one of the transfer syntax names proposed by the initiator for this abstract syntax. This field is only necessary if you accept the abstract syntax.

- The *result* field specifies whether you accept or reject the context. Section 10.13 lists possible values of the parameter.

- The field specifying the reason for rejecting an abstract syntax is ignored if the *result* field is not a provider reject. Section 10.7 lists possible values of the parameter.

pdefault_context_res

The address of a structure that specifies the response to the proposed default presentation context. If the initiator of an association proposes a default context on the P-CONNECT indication, the responder should supply a value other than null for the *pdefault_context_res* parameter. If the initiator does not propose a default context on the P-CONNECTION indication, the responder should make the *pdefault_context_res* parameter null.

Description

Call this routine after receiving an A-ASSOCIATE indication to reject the association attempt.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVDEFCTXT	The default context response is invalid.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVFUS	The functional units are invalid.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_accept_rsp
osak_associate_req
osak_open_initiator
osak_open_responder

osak_release_req

osak_release_req — Requests the orderly release of an association.

Syntax

```
status = osak_release_req (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only

Parameters Used	Data Type	Access
func	Unsigned long integer	write only
status_block	osak_status_block	write only
release_reason	osak_release_reason	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_release_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

release_reason

Specifies the reason for releasing the association. Section 10.10 lists the possible values of this parameter.

Description

The only functional unit required for this service is the kernel functional unit. The caller of this service must own all the available tokens that are in use.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.

OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_abort_req
osak_release_rsp
osak_close_port

osak_release_rsp

osak_release_rsp — Responds to a request for orderly release of an association.

Syntax

status = osak_release_rsp (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[action_result]	osak_action_result	read only
[release_resp_reason]	osak_release_resp_reason	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_release_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

action_result

The address of a value specifying acceptance or rejection of the request to release the association. Omission of this parameter means that the accepter agrees to the release request. You can use this parameter only if the negotiated release functional unit is selected.

release_resp_reason

Specifies the reason for rejecting the release request. Section 10.11 lists possible values of this parameter.

Description

Call this routine after receiving an A-RELEASE indication.

A responder can refuse a release request only if the negotiated release functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVACTION	The *action_result* parameter is invalid.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_release_req
osak_close_port

osak_resync_req

osak_resync_req — Requests the resynchronization of an association to a specified synchronization point.

Syntax

```
status = osak_resync_req (port, parameter_block)
```

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only

Parameters Used	Data Type	Access
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
resync_type	osak_resync_type	read only
[sync_point]	osak_sync_point	read only
[tokens]	osak_token_setting	read only
[pcontext_id_list]	osak_pcontext_id	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_resync_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

resync_type

Specifies the type of resynchronization that you require. The three types of resynchronization are:

- *abandon* (resync_type=OSAK_C_RESYNC_ABANDON)

  Resynchronizes to a synchronization point the serial number that is higher than the serial numbers of synchronization points in use on the existing association.

- *restart* (resync_type=PSAK_C_RESYNC_RESTART)

  Resynchronizes to a synchronization point set since the last acknowledged major synchronization point.

- *set* (resync_type=OSAK_C_RESYNC_SET)

  Resynchronizes to any valid synchronization point serial number.

sync_point

Specifies the synchronization point from which resynchronization is to start. You should not specify a value for this parameter when the *resync_type* is *abandon*; the OSAK interface supplies the value. If you specify a value when the resynchronization type is *abandon*, the OSAK interface returns the status code OSAK_S_INVSYNCPNT.

tokens

The address of a structure you can use to specify the token setting that should apply after resynchronization.

The structure consists of four fields corresponding to the four possible tokens. In each field, you can specify one of the following:

- The token is assigned to the initiator.

- The token is assigned to the responder.

- The token is assigned according to the responder's choice.

Section 10.15 lists the possible values for this parameter.

If no tokens are available on the association, but you specify the *token_item* parameter, the OSAK interface returns error status OSAK_S_INVTOKEN. If you make the parameter null, OSAK uses a default token setting. The default setting is that the peer entity that requests resynchronization has all the tokens after resynchronization.

pcontext_id_list

The address of the head of a linked list of structures specifying the DCS after resynchronization. If you omit this parameter, the DCS is empty after resynchronization. The following rules apply to each structure:

- The *next* field should be set to null in the last structure in the list.

- The *pcontextid* field should be a descriptor for an ASN.1 integer TLV encoding for a presentation context identifier.

- The *tsname* field should be a descriptor for an ASN.1 object identifier TLV encoding for a transfer syntax name.

Description

You can use this service only if the resynchronize functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVRESYNCTYPE	The resynchronization type is invalid.

OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_major_req
osak_minor_req
osak_resync_rsp

osak_resync_rsp

osak_resync_rsp — Responds to a request for resynchronization of an association.

Syntax

status = osak_resync_rsp (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[sync_point][1]	osak_sync_point	read only
[tokens][1]	osak_token_setting	read only
[pcontext_id_list][1]	osak_pcontext_id	read only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

[1]This parameter is mandatory in some situations and optional in others; see the description in the Parameters Used section.

C Binding

```
osak_resync_rsp (port, parameter_block)
```

```
osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

sync_point

Specifies the serial number of the synchronization point from which resynchronization is to start.

You should specify a value for this parameter if the type of resynchronization requested on the P-RESYNCHRONIZE indication to which you are responding is *set*.

If the type of resynchronization requested is *abandon* or *restart*, you do not need to specify a value for the *sync_point* parameter. However, if you specify a value, this should be the value received on the P-RESYNCHRONIZE indication.

tokens

The address of a structure that specifies the distribution of tokens after resynchronization. Make the address null if there are no tokens available or if the settings specified in the P-RESYNCHRONIZE indication are to be used. If the P-RESYNCHRONIZE indication specifies that the accepter should decide the distribution of tokens, this parameter is mandatory.

pcontext_id_list

The address of the head of a linked list of structures specifying the DCS after resynchronization. If you set this parameter to null when the context management functional unit is selected, the DCS is empty after resynchronization.

You should only use this parameter if the context management functional unit is selected. The parameter should contain at least the presentation context for the ACSE abstract syntax.

The following rules apply to each structure:

- The *next* field should be set to null in the last structure in the list.

- The *pcontext_id* field should be a descriptor for an ASN.1 integer TLV encoding for a presentation context identifier.

- The *ts_name* field should be a descriptor for an ASN.1 object identifier TLV encoding for a transfer syntax name.

token_item

The address of a structure that specifies the tokens that the accepter wants from the requester. The structure consists of four fields corresponding to the four tokens. In each field, the only values allowed are zero and one:

- Zero means that the accepter does not want this token from the requester.

- One means that the accepter wants this token from the requester.

Description

Call this routine after receiving a P-RESYNCHRONIZE indication.

You can use this service only if the resynchronize functional unit is selected.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPCTXT	The presentation context list is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_resync_req

osak_select

osak_select — Inspects ports for events waiting to be received or arriving within a specified time.

Syntax

status = osak_select (port_count, port_list, time_out)

Argument	Data Type	Access
port_count	osak_handle_count	read only
port_list	osak_handle	modify
time_out	osak_time	read only

C Binding

```
osak_select (port_count, port_list, time_out)

osak_handle_count port_count;
osak_handle *port_list;
osak_time *time_out;
```

Arguments

port_count

The number of ports in the port list.

port_list

The address of an array in which you specify the identifiers of the ports you want to inspect for events. You can also specify the types of events about which you want the OSAK interface to notify you. The array has three fields:

- *id* specifies one or more ports that the OSAK interface should inspect

- *request_event_mask* specifies the type of event for which the OSAK interface should inspect the ports

- *returned_event_mask* lists the events that occur on the specified ports before the OSAK interface returns control to your application

A port is defined to be one of the following:

- An OSI association. The `osak_open_initiator`, `osak_open_redirect` and `osak_open_responder` calls return the identifier of the association in the *port* parameter.

- An identifier specific to the operating system. The identifier refers to the source on which the OSAK interface makes selections. On OpenVMS systems, the source is a port or an event flag number (EFN). On UNIX and ULTRIX systems, the source is a file descriptor.

time_out

The address of a value specifying the maximum time, in seconds, that you want the OSAK interface to wait for an event if one is not present on a specified port. A value of zero indicates no waiting. A null pointer indicates an indefinite wait. The maximum permitted size for this argument is 1 day (86,400 seconds). If you want your application to wait longer than 1 day for an event to arrive, you can do either of two things:

- Set this argument to null.

- Make several calls to `osak_select` with timers lasting less than a day. When the timer expires, re-issue the `osak_select` call.

Description

The routine examines the ports listed in the *port_list* argument for the following:

- Events waiting to be received

- Events occurring within a specified time

The call finishes either when an event arrives on one of the ports listed in the *port_list* argument or when the *time_out* argument expires.

UNIX

Although the `osak_select` call maps directly on to the UNIX or ULTRIX `select(2)` system call, the semantics of the write bit are different. The OSAK interface uses the write bit to indicate that a write event has finished. The UNIX and ULTRIX operating systems use the write bit to indicate that writing is possible.

If the OSAK interface finds an OSI port identifier, it applies OSAK semantics. If the OSAK interface finds an UNIX or ULTRIX file descriptor, it maps that file descriptor to the UNIX or ULTRIX `select(2)` system call, which applies UNIX or ULTRIX semantics.

To use the UNIX or ULTRIX `select(2)` system call, first call `osak_get_handle`. This returns a file descriptor you can pass to `select(2)`.

To use the `osak_select` routine, pass either a file descriptor or a port identifier.

OpenVMS

You can pass an event flag number (EFN) to `osak_select` as a port identifier. If you do this, you should use an EFN from cluster 1. The OSAK interface passes the EFN to the OpenVMS system call SYS$WFLOR().

Each port has two associated event masks:

- The request-event mask

- The returned-event mask

If you want the OSAK interface to notify you when an inbound event from the transport provider arrives, set the read bit in the *request_event_mask* field of the *port_list* parameter. When an inbound event arrives, the OSAK interface returns OSAK_S_NORMAL.

If you want the OSAK interface to notify you when an outbound event completes, set the write bit in the *returned_event_mask* field of the *port_list* parameter. When an outbound event completes, the OSAK interface returns OSAK_S_NORMAL.

Table 1.6 shows the meanings of values in the 2-bit masks.

Table 1.6. Definitions of Request-Event Mask and Returned-Event Mask

Request-Event Bit Number	Returned-Event Bit Number	Meaning
0	0	OSAK_C_READEVENT
1	1	OSAK_C_WRITEEVENT

You can specify a maximum time for the OSAK interface to wait for an event to arrive. If no event arrives within that time, the OSAK interface returns the status OSAK_S_NOEVENT and clears the returned-event bit mask. If an event arrives, the OSAK interface returns the status OSAK_S_NORMAL. You should inspect the returned-event bit mask to find the port on which the event arrived.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_NOEVENT	No event has occurred.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INVPARAM	There is an invalid parameter, or no `request_event_mask` is specified in the `port_list` argument.
OSAK_S_INVPORT	The port identifier is invalid.

| OSAK_S_TRANSERR | There is an error in the transport provider. |

See Also

osak_get_event
osak_give_buffers

osak_send_more

osak_send_more — Sends a further segment of user data.

Syntax

status = osak_send_more (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
user_data	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
data_length	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_send_more (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Description

The OSAK interface allows you to segment the user data you are sending on any service. Call
osak_send_more as many times as necessary to send user data to complete a service. The
completion of each call to osak_send_more indicates that the transport provider has processed a
segment of data. This does not guarantee that the segment has been transferred to the peer entity.

You do not have to send any user data on the original service call. If you set the *more_flag* parameter to
true on the original service call, you can send all the user data on calls to osak_send_more.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

osak_token_give_req

osak_token_give_req — Relinquishes ownership of some or all of the available tokens.

Syntax

status = osak_token_give_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_token_give_req (port, parameter_block,)

osak_port port;
```

```
struct osak_parameter_block *parameter_block;
```

Parameters Used

token_item

The address of a structure you can use to specify the tokens that the requester is passing to the accepter. The structure consists of four fields corresponding to the four tokens.

In each field, the only values allowed are zero and one:

- Zero means that the requester is not passing this token to the accepter.

- One means that the requester is passing this token to the accepter.

Description

If you are using session version 1, there is no user data on this service and therefore no segmentation is allowed and the *more_flag* parameter must be set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_control_give_req
osak_token_please_req

osak_token_please_req

osak_token_please_req — Requests the peer entity to relinquish ownership of some or all of the available tokens.

Syntax

status = osak_token_please_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_token_please_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

token_item

The address of a structure you can use to specify the tokens that the requester wants from the accepter.
The structure consists of four fields corresponding to the four tokens. In each field, the only values
allowed are zero and one:

- Zero means that the requester does not want this token from the accepter.

- One means that the requester wants this token from the accepter.

Description

The token you want must be available and owned by the other user before you request ownership of the
token.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.

OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVTOKEN	The token setting is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

osak_control_give_req
osak_token_give_req

osak_typed_req

osak_typed_req — Transfers typed data over an association.

Syntax

status = osak_typed_req (port, parameter_block)

Argument	Data Type	Access
port	osak_port	read only
parameter_block	osak_parameter_block	read only

Parameters Used	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[user_data]	osak_buffer	read only
[user_context]	Address	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
port_id	osak_port	write only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
osak_typed_req (port, parameter_block)

osak_port port;
```

```
struct osak_parameter_block *parameter_block;
```

Description

The typed data service is useful when you select the half-duplex functional unit. You can use the service to send user information when a peer entity needs to send data and does not hold the data token. At least one byte of data must be sent if the *more_flag* parameter is set to false.

Return Value

A value indicating the status of the routine. Possible values are:

OSAK_S_FREE	The OSAK interface has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK interface has queued the request.
OSAK_S_BADPARAM	There is an invalid parameter.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_TRANSERR	There is an error in the transport provider.

Chapter 2. OSAK Events

This chapter lists the OSAK events in alphabetical order. Refer to the description of the `osak_get_event` call in Chapter 1 for how to use this routine to receive events. Chapter 1 also describes all the OSAK parameters and their data types. You should ignore parameters that are not included in the event specification. The following parameters are common to all events:

acse_pci_eoc

Indicates how many end-of-contents octets there should be in the data arriving from the remote peer entity to meet the requirements of the ACSE PCI encoding.

To arrive at the value, the OSAK interface counts the number of indefinite length encodings in the ACSE PCI for which it cannot find end-of-contents octets. You should look for this number of end-of-contents octets in the ACSE PCI. If you do not find this number, you should issue a presentation provider abort.

An end-of-contents octet consists of two zero octets.

event_type

The type of the event that `osak_get_event` receives. Table 1.3 shows the event indicated by each possible value of this parameter.

more_flag

Indicates whether there is more user data to follow. The value of this parameter is true if there are more data units to follow and false if there are no more data units to follow.

peer_data

The address of a linked list of user buffers containing user data transferred from the remote peer entity. The user data does not necessarily start at the beginning of the first buffer in the list. The *tsdu_ptr* parameter points to the head of the list.

For a user abort, the *peer_data* parameter points to the start of the user information in the incoming protocol data unit (PDU). This parameter is not used for a redirect indication.

pres_pci_eoc

Indicates how many end-of-contents octets there should be in the data arriving from the remote peer entity to meet the requirements of the presentation PCI encoding.

To arrive at the value, the OSAK interface counts the number of indefinite length encodings in the presentation PCI for which it cannot find end-of-contents octets. You should look for this number of end-of-contents octets in the presentation PCI. If you do not find this number, you should issue a provider abort.

An end-of-contents octet consists of two zero octets.

The *acse_pci_eoc* and *pres_pci_eoc* parameters are cumulative. You must check for a number of end-of-context octet sequal to the sum of *acse_pci_eoc* and *pres_pci_eoc*.

status_block

When `osak_get_event` finishes, the OSAK interface writes a status code to this parameter. If the status code is OSAK_S_TRANSERR, the OSAK interface also returns a transport provider status.

tsdu_ptr

The address of the head of a list of buffers of the type *osak_buffer*. The buffers are those that you passed to the OSAK interface in calls to `osak_give_buffers`.

To free the buffers when they are returned by `osak_get_event`, you should follow the *tsdu_ptr* pointer, not the *peer_data* pointer.

ABORT indication

ABORT indication — Indicates a user or a provider abort.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
abort_reason	osak_abort_reason
[abort_ppdu]	osak_abort_ppdu
[pcontext_id_list]	osak_pcontext_id
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer
local_abort	Long integer

Parameters Returned

abort_reason

This parameter explains why the association in being aborted. Section 10.2 lists the possible values.

abort_ppdu

If the value in the *abort_reason* parameter is OSAK_C_PP_ABORT_INVALID_VALUE, OSAK_C_PP_ABORT_UNREC_PARAM, or OSAK_C_PP_ABORT_UNEXP_PARAM, indicating a presentation provider abort, the OSAK interface uses this parameter to return the identifier of the event that contained an invalid value.

pcontext_id_list

The address of the head of a linked list of structures, each one specifying one presentation context (a presentation context identifier and its associated transfer syntax) for which there is user data encoded in the user data parameter.

This parameter is null if the event is a presentation provider abort.

local_abort

This parameter is true if the OSAK interface generated the ABORT indication locally and false if the ABORT indication originated from the remote peer entity.

Description

An application entity receives this event when one of the following circumstances occurs:

*   The remote peer entity issues a call to `osak_abort_req`.

*   The remote protocol machine issues a provider abort.

*   The local protocol machine issues a provider abort.

This event indicates that the remote peer entity, the local protocol machine, or the remote protocol machine is terminating the association. You should delete all the data structures for the association.

If this is an ACSE user abort, your application should make multiple calls to `osak_get_event` to receive all the userdata arriving on the ABORT indication, if any. When all the user data has been received, close the OSAK port.

A-ASSOCIATE-ACCEPT confirm

A-ASSOCIATE-ACCEPT confirm — An application entity receives this event when the remote peer entity calls `osak_accept_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
acontext	osak_mem_descriptor
[responding_aei]	osak_aei
[protocol_versions]	osak_protocol_versions
[sconnect_id]	osak_sconnect_id
[segmentation]	osak_segmentation
[initial_serial_number]	osak_sync_point
[initial_tokens]	osak_token_setting
[request_tokens]	osak_token_setting
functional_units	osak_fus
pcontext_res_list	osak_pcontext_proposal_result
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer

Parameters Returned	Data Type
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

acontext

The address of an encoded object identifier representing the application context name.

responding_aei

The address of information about the application entity that is responding to a request to set up an association. The information can include the selectors, the application-entity title, and the application-entity identifier.

protocol_versions

The address of the identifiers of the protocol versions in use on the association.

sconnect_id

The address of encoded session connection information. If the incoming data units do not contain session connection information, the OSAK interface returns a null address.

segmentation

The address of session segmentation data that specifies:

- Whether session segmentation is in use

- If session segmentation is in use, the maximum size permitted for a TSDU

If the incoming data units do not contain session segmentation information, the OSAK interface returns a null address.

initial_serial_number

The address of the initial synchronization point serial number on this association. If the incoming data units do not contain a value for the initial synchronization point serial number, the OSAK interface returns a null address.

initial_tokens

The address of a structure indicating the initial token settings for the association. If the incoming data units do not contain values for the initial token settings, the OSAK interface returns a null address.

request_tokens

The address of the token identifiers that the calling application entity is requesting from its peer. If the incoming data units do not contain values for requested tokens, the OSAK interface returns a null address.

functional_units

The address of the presentation and session functional units proposed for this association. If the incoming data units contain no functional units, the OSAK interface returns the default values:

• Half-duplex functional unit

• Minor synchronization functional unit

• Activity functional unit

• Capability functional unit

• Exceptions functional unit

pcontext_res_list

The address of the head of a linked list of structures, each of which gives the response to the corresponding context proposed in the A-ASSOCIATE request. If the response is acceptance, the OSAK interface returns, in the *ts_name* field of this parameter, the ASN.1 encoding for the identifier of the transfer syntax being used. Section 10.13 lists the values that the fields of this parameter may have.

Description

This event indicates a positive response to a request to establish an association. It means that the association has been established and you can start transferring data.

A-ASSOCIATE-REJECT confirm

A-ASSOCIATE-REJECT confirm — An application entity receives this event when the remote peer entity calls `osak_reject_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
reject_reason	osak_reject_reason
acontext	osak_mem_descriptor
[responding_aei]	osak_aei
[sconnect_id]	osak_sconnect_id
[functional_units]	osak_fus
[pcontext_res_list]	osak_pcontext_proposal_result
[pdefault_context_res]	osak_default_context_result
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

reject_reason

This parameter explains why the association request is being rejected. Section 10.9 lists the values this parameter may have.

If the value returned indicates that the rejection is due to temporary congestion, the initiator can try again to establish an association.

acontext

The address of an encoded object identifier representing the application context name.

responding_aei

The address of information about the application entity that is responding to a request to set up an association. The information may include the selectors, the application-entity title, and the application-entity identifier.

sconnect_id

The address of encoded session connection information. If the incoming data units do not include any session connection information, the OSAK interface returns a null address.

functional_units

The address of the presentation and session functional units. If the incoming data units do not include these values, the OSAK interface returns the default values:

- Half-duplex functional unit

- Minor synchronize functional unit

- Activity functional unit

- Capability functional unit

- Exceptions functional unit

pcontext_res_list

The address of the head of a linked list of structures, each of which gives the response to the corresponding context proposed in the A-ASSOCIATE request. If the response is acceptance, the OSAK interface returns, in the *ts_name* field of this parameter, the ASN.1 encoding for the identifier of the transfer syntax being used. Section 10.13 lists the values that the fields of this parameter may have.

pdefault_context_res

The address of the response to the proposed default context for this association. If incoming data units do not include the response, the OSAK interface returns a null address.

Description

This event indicates a negative response to a request to establish an association. Check the reason for the refusal given in the parameter *reject_reason*. If the reason is a temporary problem, for example congestion or lack of resources, you can try to establish the association again.

A-ASSOCIATE indication

A-ASSOCIATE indication — An application entity receives this event when the remote peer entity calls `osak_associate_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
acontext	osak_mem_descriptor
called_aei	osak_aei
[calling_aei]	osak_aei
protocol_versions	osak_protocol_versions
[sconnect_id]	osak_sconnect_id
[segmentation]	osak_segmentation
[initial_serial_number]	osak_sync_point
[initial_tokens]	osak_token_setting
functional_units	osak_fus
pcontext_list	osak_pcontext_proposal
[pdefault_context]	osak_default_context
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

acontext

The address of an encoded object identifier representing the application context name.

called_aei

The address of information about the called application entity. The information may include the selectors, the application-entity title, and the application-entity identifier. If the incoming data units do not include this information, the OSAK interface returns a null address.

calling_aei

The address of information about the calling application entity. The information may include the selectors, the application-entity title, and the application-entity identifier. If the incoming data units do not contain this information, the OSAK interface returns a null address.

protocol_versions

The address of the identifiers of the protocol versions proposed for use on the association. If the incoming data units do not specify any proposed protocol version identifiers, the OSAK interface returns default values. Table 2.1 shows the default versions.

Table 2.1. A-ASSOCIATE indication: Default Protocol Version Numbers

Protocol	Possible Version Numbers
ACSE	1
Presentation	1
Session	1 or 2

sconnect_id

The address of encoded session connection information. If the incoming data units do not include any session connection information, the OSAK interface returns a null address.

segmentation

The address of session segmentation information that specifies:

- Whether session segmentation is in use

- If session segmentation is in use, the maximum size permitted for a TSDU

If the incoming data units do not include session segmentation information, the OSAK interface returns a null address.

initial_serial_number

The address of the initial synchronization point serial number of the association. If the incoming data units do not specify a value for the initial synchronization point serial number, the OSAK interface returns a null address.

initial_tokens

The address of the initial token settings for the association. If the incoming data units do not specify values for the initial token settings, the OSAK interface returns a null address.

functional_units

The address of the presentation and session functional units accepted for this association. If the incoming data units do not include these values, the OSAK interface returns the default values:

- Half-duplex functional unit

- Minor synchronize functional unit

- Activity functional unit

- Capability functional unit

- Exceptions functional unit

pcontext_list

The address of the head of a linked list of structures, each of which contains the following information about one of the presentation contexts that the initiator is proposing:

- The presentation context identifier

- A reference to the head of a linked list of transfer syntax names

- The abstract syntax name

pdefault_context

The address of the names of the transfer syntax and the abstract syntax that make up the default context proposed for this association. If the incoming data units do not specify any default context, the OSAK interface returns a null address.

Description

This event indicates a request to establish an association. Accept the association request by calling `osak_accept_rsp`; reject the association by calling `osak_reject_rsp`.

A-RELEASE confirm

A-RELEASE confirm — An application entity receives this event when the remote peer entity calls `osak_release_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
action_result	osak_action_result
[release_resp_reason]	osak_release_resp_reason
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

action_result

The address of a value indicating acceptance or rejection of the release request. If the incoming data units do not specify a value for this parameter, the OSAK interface returns a null address, which means that the release request is accepted.

release_resp_reason

The reason for rejection of the release request. Section 10.11 lists the values this parameter may have.

Description

This event responds to a request to terminate an association. By the time the event arrives, the association has been terminated. To reclaim parameter blocks and user buffers that were used on the association, you can call `osak_close_port`, or use a completion routine (OpenVMS systems only).

A-RELEASE indication

A-RELEASE indication — An application entity receives this event when the remote peer entity calls `osak_release_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
release_reason	osak_release_reason
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

release_reason

A value indicating the reason for releasing the association. If the incoming data units do not include a value for this parameter, the OSAK interface returns the value zero, which indicates a normal release. Section 10.10 lists the values this parameter may have.

Description

This event indicates a request to release an association. Respond to the event in one of the following ways:

- If the negotiated release functional unit is not in use on the association, call `osak_release_rsp` to accept the release request.

- If the negotiated release functional unit is in use on the association, call `osak_release_rsp` to accept or reject the release request.

P-ACTIVITY-DISCARD confirm

P-ACTIVITY-DISCARD confirm — An application entity receives this event when the remote peer entity calls `osak_act_discard_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event responds to a request to discard an activity and confirms that the activity has been discarded.

P-ACTIVITY-DISCARD indication

P-ACTIVITY-DISCARD indication — An application entity receives this event when the remote peer entity calls `osak_act_discard_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
reason	osak_activity_reason
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

activity_reason

The address of the reason for discarding an activity. If the incoming event does not include a value for this parameter, the OSAK interface returns a null address. Section 10.4 lists the values this parameter may have.

Description

This event indicates a request to discard the current activity. Respond by calling `osak_act_discard_rsp`.

P-ACTIVITY-END confirm

P-ACTIVITY-END confirm — An application entity receives this event when the remote peer entity calls `osak_act_end_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event responds to a request to terminate the current activity and confirms that the activity has been terminated.

P-ACTIVITY-END indication

P-ACTIVITY-END indication — An application entity receives this event when the remote peer entity calls `osak_act_end_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[sync_point]	osak_sync_point
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

sync_point

The address of the major synchronization point serial number current when the end of the activity was requested.

Description

This event indicates a request to terminate the current activity. Respond by calling
`osak_act_end_rsp`.

P-ACTIVITY-INTERRUPT confirm

P-ACTIVITY-INTERRUPT confirm — An application entity receives this event when the remote peer
entity calls `osak_act_interrupt_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event responds to a request to interrupt the current activity.

P-ACTIVITY-INTERRUPT indication

P-ACTIVITY-INTERRUPT indication — An application entity receives this event when the remote peer
entity calls `osak_act_interrupt_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[activity_reason]	osak_activity_reason
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

activity_reason

The address of the reason for the interruption of the activity. If the incoming event does not include a value for this parameter, the OSAK interface returns a null address. Section 10.4 lists the values this parameter may have.

Description

This event indicates a request to interrupt the current activity. Respond by calling `osak_act_interrupt_rsp`.

P-ACTIVITY-RESUME indication

P-ACTIVITY-RESUME indication — An application entity receives this event when the remote peer entity calls `osak_act_resume_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
activity_id	osak_mem_descriptor
old_activity_id	osak_mem_descriptor
sync_point	osak_sync_point
[old_sconnection_id]	osak_sconnection_id
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

activity_id

The identifier of the resumed activity.

old_activity_id

The identifier of the interrupted activity.

sync_point

The address of the synchronization point serial number at which to resume the activity.

old_sconnection_id

The address of the session connection identification information from the session connection over which the interrupted activity occurred.

Description

This event indicates a request to resume a previously interrupted activity. No response to the request is necessary because resumption of an activity is not a confirmed service.

P-ACTIVITY-START indication

P-ACTIVITY-START indication — An application entity receives this event when the remote peer entity calls `osak_act_start_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
activity_id	osak_mem_descriptor
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

activity_id

The identifier of the new activity.

Description

This event indicates a request to start a new activity. No response to the request is necessary because starting an activity is not a confirmed service.

P-ALTER-CONTEXT confirm

P-ALTER-CONTEXT confirm — An application entity receives this event when the remote peer entity calls `osak_alter_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[pcontext_res_list]	osak_pcontext_proposal_result

Parameters Returned	Data Type
[pcontext_del_res_list]	osak_pcontext_deletion_result
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

pcontext_res_list

The address of the head of a linked list of structures, each of which gives the response to the corresponding context proposed in the P-ALTER-CONTEXT request. If the response is acceptance, the OSAK interface returns, in the *ts_name* field of this parameter, the ASN.1 encoding for the identifier of the transfer syntax being used. Section 10.13 lists the range of values that the fields of this parameter may have.

pcontext_del_res_list

The address of the head of a linked list of structures, each of which gives the response to the proposed deletion of a context from the DCS.

Description

This event gives the response to a proposed alteration in the DCS existing on an association.

P-ALTER-CONTEXT indication

P-ALTER-CONTEXT indication — An application entity receives this event when the remote peer entity calls `osak_alter_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[pcontext_list]	osak_pcontext_proposal
[pcontext_del_list]	osak_pcontext_deletion
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

pcontext_list

The address of the head of a linked list of structures, in each of which you can find the following information about one of the presentation contexts proposed for addition to the DCS:

- The presentation context identifier

- The address of the head of a linked list of transfer syntax names

- The abstract syntax name

pcontext_del_list

The address of the head of a linked list of structures, in each of which you can find the presentation context identifier of one presentation context proposed for deletion from the DCS.

Description

This event indicates a proposed alteration in the DCS existing on an association. Respond by calling `osak_alter_rsp`.

P-CAPABILITY-DATA confirm

P-CAPABILITY-DATA confirm — An application entity receives this event when the remote peer entity calls `osak_capability_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event confirms that capability data sent by the remote peer entity has been received.

P-CAPABILITY-DATA indication

P-CAPABILITY-DATA indication — An application entity receives this event when the remote peer entity calls `osak_capability_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event

Parameters Returned	Data Type
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event indicates that capability data is being sent. Respond by calling `osak_capability_rsp`.

P-CONTROL-GIVE indication

P-CONTROL-GIVE indication — An application entity receives this event when the remote peer entity calls `osak_control_give_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event indicates that the remote peer entity relinquishes ownership of all the tokens available on an association. No response is necessary because relinquishing tokens is not a confirmed service.

P-DATA indication

P-DATA indication — An application entity receives this event when the remote peer entity calls `osak_data_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block

Parameters Returned	Data Type
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event indicates that normal data is being transferred. No response to the request is necessary because sending normal data is not a confirmed service.

P-EXPEDITED-DATA indication

P-EXPEDITED-DATA indication — An application entity receives this event when the remote peer entity calls osak_expedited_req.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event indicates the sending of expedited data over an association. No response is necessary because sending expedited data is not a confirmed service.

P-P-EXCEPTION-REPORT indication

P-P-EXCEPTION-REPORT indication — An application entity receives this event when the service provider signals an exception.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block

Parameters Returned	Data Type
tsdu_ptr	osak_buffer
exception_reason	osak_exception_reason
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

Parameters Returned

exception_reason

A value indicating the reason for the exception report. Section 10.5 lists the values this parameter may have.

Description

This event indicates a user error has occurred that is not severe enough to cause an association to be aborted. The *exception_reason* parameter contains a value indicating the reason for the error. You should examine the parameter and take corrective action as appropriate.

P-U-EXCEPTION-REPORT indication

P-U-EXCEPTION-REPORT indication — An application entity receives this event when the remote peer entity calls `osak_exception_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
exception_reason	osak_exception_reason
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

exception_reason

A value indicating the reason for the exception report. Section 10.5 lists the values this parameter may have.

Description

This event indicates an error has occurred that is not severe enough to cause an association to be aborted. The error originates from the service user. The *exception_reason* parameter contains a value indicating the reason for the error. You should examine the parameter and take corrective action as appropriate.

P-RESYNCHRONIZE confirm

P-RESYNCHRONIZE confirm — An application entity receives this event when the remote peer entity calls `osak_resync_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
sync_point	osak_sync_point
[tokens]	osak_token_setting
[token_item]	osak_token_setting
[pcontext_id_list]	osak_pcontext_id
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

sync_point

The address of the synchronization point serial number specified in the resynchronization request.

tokens

The address of the existing token assignments. To interpret the assignment, examine the *redirect_state* parameter to determine whether the local entity is acting as initiator or responder.

token_item

The address of information about the assignment of tokens on completion of the resynchronization service. If the incoming data units do not include values for the assignment of tokens, the OSAK interface returns a null address.

pcontext_id_list

The address of the head of a linked list of structures, each one confirming one presentation context in the DCS resulting from the resynchronization.

Description

This event confirms resynchronization from a specified synchronization point.

P-RESYNCHRONIZE indication

P-RESYNCHRONIZE indication — An application entity receives this event when the remote peer entity calls `osak_resync_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
resync_type	osak_resync_type
sync_point	osak_sync_point
[token_item]	osak_token_setting
[pcontext_id_list]	osak_pcontext_id
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

resync_type

A value indicating the type of resynchronization. Section 10.14 lists the values this parameter may have.

sync_point

The address of the synchronization point serial number from which resynchronization should start.

token_item

The address of a structure indicating the tokens that the remote application entity is requesting. If the incoming data units do not include this value, the OSAK interface returns zero. Zero indicates that all the tokens are available to the application entity that is requesting resynchronization.

Section 10.15 lists the values this parameter may have.

pcontext_id_list

The address of the head of a linked list of structures, each one indicating one presentation context in the DCS resulting from the resynchronization. If this parameter is null and the context management functional unit is selected, the DCS is empty after resynchronization. If the context management functional unit is not selected, ignore this parameter.

Description

This event indicates that a request to resynchronize an association from a specified synchronization point. Respond by calling `osak_resync_rsp`.

P-SYNC-MAJOR confirm

P-SYNC-MAJOR confirm — An application entity receives this event when the remote peer entity calls `osak_major_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event confirms the setting of a major synchronization point.

P-SYNC-MAJOR indication

P-SYNC-MAJOR indication — An application entity receives this event when the remote peer entity calls `osak_major_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
sync_point	osak_sync_point
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

sync_point

The address of the major synchronization point serial number.

Description

This event indicates a request to set a major synchronization point. Respond by calling `osak_major_rsp`.

P-SYNC-MINOR confirm

P-SYNC-MINOR confirm — An application entity receives this event when the remote peer entity calls `osak_minor_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
sync_point	osak_sync_point
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

sync_point

The address of the minor synchronization point serial number.

data_separation

Indicates whether data separation is required. The value is true if the remote peer entity requires data separation.

Description

This event confirms the setting of a minor synchronization point.

P-SYNC-MINOR indication

P-SYNC-MINOR indication — An application entity receives this event when the remote peer entity calls `osak_minor_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event

Parameters Returned	Data Type
status_block	osak_status_block
tsdu_ptr	osak_buffer
sync_point	osak_sync_point
sync_confirm	osak_sync_confirm
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

sync_point

The address of the minor synchronization point serial number.

sync_confirm

The value of this parameter is true if the remote peer entity requested confirmation of the synchronization point and false if the remote application entity did not request a confirmation of the synchronization point.

data_separation

Indicates whether data separation is required. The value is true if the remote peer entity requires data separation.

Description

This event indicates a request to set a minor synchronization point. You can respond by calling `osak_minor_rsp`, but you do not have to do so. If you do not do so, you imply agreement with the parameters set by the remote peer entity.

P-TOKEN-GIVE indication

P-TOKEN-GIVE indication — An application entity receives this event when the remote peer entity calls `osak_token_give_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
token_item	osak_token_setting
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer

Parameters Returned	Data Type
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

token_item

The address of a structure that indicates the tokens that the remote application entity is passing to its peer.

Description

This event indicates that its sender relinquishes ownership of all or some of the tokens available on an association. The *token_item* parameter specifies which tokens are being relinquished. No response is necessary because relinquishing tokens is not a confirmed service.

P-TOKEN-PLEASE indication

P-TOKEN-PLEASE indication — An application entity receives this event when the remote peer entity calls `osak_token_please_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
token_item	osak_token_setting
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

token_item

The address of a structure indicating the tokens that the remote application entity is requesting from its peer.

Description

This event indicates that the remote peer entity is requesting the recipient of the event to relinquish some or all of the tokens available on the association. The *token_item* parameter specifies which tokens are being requested. No response to the request is necessary because this is not a confirmed service.

P-TYPED-DATA indication

P-TYPED-DATA indication — An application entity receives this event when the remote peer entity calls `osak_typed_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Description

This event indicates that typed data is being sent over an association. No response is necessary because sending typed data is not a confirmed service.

REDIRECT indication

REDIRECT indication — An application entity receives this event when the remote peer entity calls `osak_redirect.`

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
redirect_state	osak_state
[pcontext_list]	osak_pcontext
[pcontext_redirect_list]	osak_pcontext_proposal
[pdefault_context]	osak_default_context
[calling_aei]	osak_aei
[called_aei]	osak_aei
[acontext]	osak_mem_descriptor
[protocol_versions]	osak_protocol_versions
[sconnect_id]	osak_sconnect_id
[segmentation]	osak_segmentation
[sync_point]	osak_sync_point

Parameters Returned	Data Type
[tokens]	osak_token_setting
[activity_id]	osak_mem_descriptor
[functional_units]	osak_fus
[rcv_data_list]	osak_buffer
[local_data]	osak_mem_descriptor
[acse_pci_eoc]	Unsigned long integer
[pres_pci_eoc]	Unsigned long integer
[more_flag]	Long integer

Parameters Returned

redirect_state

A structure defining the state of the association. The values in the structure indicate whether the local entity is acting as an initiator of or a responder to the redirection request.

pcontext_list

The address of a list of structures, each one of which is a member of the DCS that existed when the association was originally set up. The OSAK interface returns this parameter only if the process that redirects the association has not established the association when it makes the request to redirect.

pcontext_redirect_list

The address of a list of structures, each one of which is a member of the DCS that existed when the association was originally set up. The OSAK interface returns this parameter only if the process that redirects the association has established the association when it makes the request to redirect.

pdefault_context

The address of the presentation context identifier and the transfer syntax identifier of the proposed or negotiated default context.

calling_aei

The address of a structure containing information about the calling application entity. The information may include the selectors, the application-entity title, and the application-entity identifier. If the incoming data units do not include this information, the OSAK interface returns a null address.

called_aei

The address of a structure containing information about the called application entity. The information may include the selectors, the application-entity title, and the application-entity identifier. If the incoming data units do not include this information, the OSAK interface returns a null address.

acontext

The address of an encoded object identifier representing the application context name.

protocol_versions

The address of a structure indicating the protocol versions proposed for use on the redirected connection. If the incoming data units do not include any proposed protocol version identifiers, the OSAK interface returns default values. Table 2.2 shows the default versions.

Table 2.2. REDIRECT indication: Default Protocol VersionNumbers

Protocol	Possible Version Numbers
ACSE	1
Presentation	1
Session	1 or 2

sconnect_id

The address of encoded session connection information. If the incoming data units do not include any session connection information, the OSAK interface returns a null address.

segmentation

The address of a structure containing session segmentation data that specifies the following:

- Whether session segmentation is in use

- If session segmentation is in use, the maximum size permitted for a TSDU

If the incoming data units do not include session segmentation information, the OSAK interface returns a null address.

sync_point

The address of the current synchronization point serial number.

tokens

The address of a structure indicating the existing token assignments. To interpret the assignment, examine the *redirect_state* parameter to determine whether the local entity is acting as initiator or responder.

activity_id

The identifier of the current activity.

functional_units

The address of a structure indicating the functional units in use for the Session and Presentation layers. If the incoming data units do not include these values, the OSAK interface returns default values:

- Half-duplex functional unit

- Minor synchronize functional unit

- Activity functional unit

- Capability functional unit

- Exceptions functional unit

rcv_data_list

The OSAK interface returns the address of a list of the buffers holding user data for the service in progress. If there are insufficient buffers to receive all the user data, the OSAK interface returns true in the *more_flag* parameter. On a redirect, the *rcv_data_list* parameter is used instead of the *peer_data* parameter.

local_data

The address of a structure that holds the address of a buffer containing data sent by the redirecting application entity.

Description

This event indicates that an association has been redirected from another local process.

The sequence of calls that you should make after the arrival of a REDIRECT indication depends on two things:

- The state of the association when the requester called `osak_redirect`

- Whether all the user data fits into the buffer supplied in the most recent call to `osak_give_buffers`

The description of the routine `osak_redirect` in Chapter 1 describes the possible states of an association. Table 2.3 shows the sequence of calls you should make for each state. In this table, each state is represented by its constant value. Section 10.6 lists the meanings of these constants.

Table 2.3. Sequence of Calls After the Arrival of a REDIRECT Indication

State	Sequence of Calls
OSAK_C_ASSOC_IND	Call `osak_get_event` to receive the REDIRECT indication. Call `osak_accept_rsp` or `osak_reject_rsp` to accept or reject the REDIRECT indication.
OSAK_C_PARTIAL_ASSOC_IND	Make multiple calls to `osak_get_event` to receive all the user data from the REDIRECT indication. Call `osak_accept_rsp` to accept or `osak_reject_rsp` to reject the REDIRECT indication.
OSAK_C_DATA_TRANSFER	Continue exchange of data between peer entities.

Chapter 3. ROSE Routines

This chapter contains the following details about the ROSE interface:

- A list of the include files for different operating systems (Section 3.1)

- A description of each constructed data type used in the ROSE interface (Section 3.2)

- A description of the arguments common to all ROSE calls (Section 3.3)

- A description of each ROSE routine (Section 3.4)

3.1. Include Files

The include files for the ROSE interface have the same name for each operating system: `osak_rose_codes.h`. Their locations for the different operating systems are:

OpenVMS	SYS$COMMON:[SYSLIB]
UNIX	/usr/include/osi
ULTRIX	/usr/include

3.2. Data Type Definitions

This section describes the data types used in the ROSE interface.

3.2.1. osak_buffer

Field	Brief Description	Data Type
next	Pointer to next element in list	Address (osak_buffer)
buffer_ptr	Pointer to beginning of buffer	Unsigned octet
buffer_length	Length of buffer	Unsigned long integer
data_ptr	Start of user data	Unsigned octet
data_length	Length of user data	Unsigned long integer

3.2.2. osak_mem_descriptor

Field	Brief Description	Data Type
size	Length of buffer in octets	Unsigned long integer
pointer	Reference to buffer	Address (unsigned octet)

3.2.3. osak_ro_problem

Address (unsigned octet)

3.2.4. osak_ro_reason

Address (unsigned octet)

3.2.5. osak_rose_pb

Parameter	Description	Data Type
pb_length	Length of ROSE parameter block including working space	Unsigned long integer
ws_length	Length of working space	Unsigned long integer
arg_length	Length of ROSE PDU that contains error, result, or argument parameter (user data)	Unsigned long integer
invoke_id	Address of ROSE invocation identifier	osak_mem_descriptor
linked_id	Address of linked ROSE invocation identifier	osak_mem_descriptor
local_value	Address locally defined operation or error value	osak_mem_descriptor
global_value	Address of globally defined operation or error value	osak_mem_descriptor
reason	Address of reject reason code	osak_ro_reason
problem	Address of problem code	osak_ro_problem
buffer	Address of buffer containing outcome of a ROSE request	osak_buffer
pdu_type	Type of PDU being encoded or decoded	Unsigned long integer
osak_rose_status	ROSE status code	osak_status_block

3.2.6. osak_status_block

Field	Brief Description	Data Type
osak_status1	OSAK status code	Unsigned long integer
osak_status2	Secondary OSAK status code	Unsigned long integer
transport_status1	Generic transport provider status	Unsigned long integer
transport_status2	Specific transport provider status	Unsigned long integer

3.3. Common Arguments

The *port* and *rose_pb* arguments are common to all the ROSE routines.

port

The identifier of the association over which you are making the ROSE call. You need to supply this identifier so the ROSE interface can use your OSAK memory allocation and deallocation routines if necessary. Refer to the description of the *buffer* parameter for further information.

rose_pb

The address of a ROSE parameter block. A ROSE parameter block is a structure that contains all possible parameters for all ROSE services. The ROSE interface uses only the relevant parameters in each service call, ignoring the rest. Section 3.2.5 describes the structure of a ROSE parameter block.

3.4. ROSE Routine Descriptions

This section contains a description of each ROSE routine.

Optional parameters are shown in square brackets ([]) in the Syntax section of each description.

osak_ro_invoke

osak_ro_invoke — Encodes ROSE PCI that requests a remote peer entity to perform an operation.

Syntax

`status = osak_ro_invoke (port, rose_pb)`

Argument	Data Type	Access
port	osak_port	read only
rose_pb	osak_rose_pb	read only

Parameters	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
[arg_length]	Unsigned long integer	read only
pdu_type	Unsigned longword	read only
invoke_id	osak_mem_descriptor	read only
[linked_id]	osak_mem_descriptor	read only
local_value	osak_mem_descriptor	read only
global_value	osak_mem_descriptor	read only
[buffer]	osak_buffer	read only
[osak_rose_status]	osak_status_block	write only

C Binding

```
osak_ro_invoke(port, rose_pb)

osak_port port;
struct osak_rose_pb *rose_pb;
```

Parameters Used

pb_length

The length of the ROSE parameter block, including the length of the working space.

ws_length

The length of the working space contained in the ROSE parameter block. The length of the working space should be 8 octets.

arg_length

The length of a ROSE PDU containing an error, result, or argument parameter. You pass the error, result, or argument as userdata in the *buffer* parameter. Set this parameter to zero if you are not sending an argument, result, or error parameter.

The ROSE interface encodes the value of the *arg_length* parameter as part of the ROSE PCI.

pdu_type

Specifies the type of APDU you are sending. Set this parameter to ROSE_C_INVOKE.

The ROSE interface uses the value in this parameter to check that you have included all the mandatory parameters for the routine you are calling.

invoke_id

The address of the identifier of this call to the Invoke function. You should assign an Invoke identifier that distinguishes this instance of the Invoke function from any other instance of the function.

When you call the Invoke function, you can reuse an Invoke identifier only if you have received a response to the Invoke request that previously used that identifier. If you have not received a response, you should not reuse the Invoke identifier.

You should set the *pointer* field of this parameter to the address of an ASN.1 encoded integer (TLV), and the *size* field to the length of the encoding.

linked_id

The address of the identifier of some other instance of the Invoke function to which you want this instance to be linked.

You should set the *pointer* field of this parameter to the address of an ASN.1 encoded integer (TLV), and the *size* field to the length of the encoding.

local_value

The address of a locally defined operation code.

Set the *pointer* field to the address of an ASN.1 encoded integer representing an operation code and the *size* field to the length of the encoding.

Note

The *local_value* and *global_value* parameters are mutually exclusive.

If you use the *local_value* parameter, you should set the *pointer* field of the *global_value* parameter to null and the *size* field to zero.

---

global_value

The address of a globally defined operation code.

Set the *pointer* field to the address of an ASN.1 encoded object identifier representing an operation code and the *size* field to the length of the encoding.

Note

The parameters *global_value* and *local_value* are mutually exclusive.

If you use the *global_value* parameter, you should set the *pointer* field of the *local_value* parameter to null and the *size* field to zero.

buffer

The address of a buffer structure you can use to pass user information required by your application. This parameter is optional. You should set the parameter to null if you are not passing a user buffer to the ROSE interface.

If you supply a user buffer, but the buffer is not big enough for the encoded ROSE PCI, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSEPCI in the new buffer and chains it to the front of the buffer you supplied.

If you do not supply a user buffer, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSE PCI in the new buffer. The *buffer_ptr* and *data_ptr* parameters both point to the same location.

The ROSE interface modifies your user buffer to accommodate the ROSEPCI. Therefore, you should save a copy of any buffer you pass in this routine call, unless you are certain you do not need to reference the unchanged buffer again.

osak_rose_status

Returns a status code specific to ROSE in the *osak_status2* field.

Description

This routine encodes ROSE PCI that requests a remote peer entity to perform an operation.

An instance of this routine is distinguished from all other instances by its Invoke identifier.

You do not need to wait for the server to perform one operation before you request another operation. You can request the server to perform an unlimited number of operations at a given time, within the limits of the system resources available to the server.

Return Value

A value indicating the status of the routine. Possible status values are:

OSAK_S_NORMAL	Routine has finished without error.
OSAK_S_BADPARAM	At least one mandatory parameter is missing.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVPARAM	At least one parameter is incorrectly specified.

osak_ro_result

osak_ro_result — Encodes ROSE PCI that reports the success of a requested operation.

Syntax

```
status = osak_ro_result (port, rose_pb)
```

Argument	Data Type	Access
port	osak_port	read only
rose_pb	osak_rose_pb	read only

Parameters	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
[arg_length]	Unsigned long integer	read only
pdu_type	Unsigned longword	write only
invoke_id	osak_mem_descriptor	read only
local_value	osak_mem_descriptor	read only
global_value	osak_mem_descriptor	read only
[buffer]	osak_buffer	read only
[osak_rose_status]	osak_status_block	write only

C Binding

```
osak_ro_result(port, rose_pb)

osak_port port;
struct osak_rose_pb *rose_pb;
```

Parameters Used

pb_length

The length of the ROSE parameter block, including the length of the working space.

ws_length

The length of the working space contained in the ROSE parameter block. The length of the working space should be 8 octets.

arg_length

The length of a ROSE PDU containing an error, result, or argument parameter. You pass the error, result, or argument as userdata in the *buffer* parameter. The ROSE interface encodes the value of the *arg_length* parameter as part of the ROSE PCI. Set this parameter to zero if you are not sending an argument, result, or error parameter.

pdu_type

Specifies the type of APDU that you are sending. Set this parameter to ROSE_C_RESULT.

The ROSE interface uses the value in this parameter to check that you have included all the mandatory parameters for the routine you are calling.

invoke_id

The Invoke identifier of the instance of the Invoke function to which you are responding.

local_value

The address of a locally defined operation code.

Set the *pointer* field to the address of an ASN.1 encoded integer representing an operation code and the *size* field to the length of the encoded integer.

Note

The *local_value* and *global_value* parameters are mutually exclusive.

If you use the *local_value* parameter, you should set the *pointer* field of the *global_value* parameter to null and the *size* field to zero.

---

global_value

The address of a globally defined operation code.

Set the *pointer* field to the address of an ASN.1 encoded object identifier representing an operation code and the *size* field to the length of the encoded object identifier.

Note

The *global_value* and *local_value* parameters are mutually exclusive.

If you use the *global_value* parameter, you should set the *pointer* field of the *local_value* parameter to null and the *size* field to zero.

---

buffer

The address of a buffer structure that you can use to pass user information required by your application, for example, the results of the successful operation. This parameter is optional. You should set the parameter to null if you are not passing a user buffer to the ROSE interface.

If you supply a user buffer, but the buffer is not big enough for the encoded ROSE PCI, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSE PCI in the new buffer and chains it to the front of the buffer you supplied.

If you do not supply a user buffer, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSE PCI in the new buffer. The *buffer_ptr* and *data_ptr* parameters both point to the same location.

The ROSE interface modifies your user buffer to accommodate the ROSE PCI. Therefore, you should save a copy of any buffer you pass in this routine call, unless you are certain that you do not need to reference the unchanged buffer again.

---

osak_rose_status

Returns a status code specific to ROSE in the *osak_status2* field.

Description

This routine encodes ROSE PCI that reports the success of an operation to the client that requested the operation.

Return Value

A value indicating the status of the routine. Possible status values are:

OSAK_S_NORMAL	Routine has finished without error.
OSAK_S_BADPARAM	At least one mandatory parameter is missing.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVPARAM	At least one parameter is incorrectly specified.

osak_ro_error

osak_ro_error — Encodes ROSE PCI that reports the failure of a requested operation.

Syntax

```
status = osak_ro_error (port, rose_pb)
```

Argument	Data Type	Access
port	osak_port	read only
rose_pb	osak_rose_pb	read only

Parameters	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
[arg_length]	Unsigned long integer	read only
pdu_type	Unsigned longword	read only
invoke_id	osak_mem_descriptor	read only
local_value	osak_mem_descriptor	read only
global_value	osak_mem_descriptor	read only
[buffer]	osak_buffer	read only
[osak_rose_status]	osak_status_block	write only

C Binding

```
osak_ro_error(port, rose_pb)

osak_port port;
```

```
struct osak_rose_pb *rose_pb;
```

Parameters Used

pb_length

The length of the ROSE parameter block, including the length of the working space.

ws_length

The length of the working space contained in the ROSE parameter block. The length of the working space should be 8 octets.

arg_length

The length of a ROSE PDU containing an error, result, or argument parameter. You pass the error, result, or argument as userdata in the *buffer* parameter. The ROSE interface encodes the value of the *arg_length* parameter as part of the ROSE PCI. Set this parameter to zero if you are not sending an argument, result, or error parameter.

pdu_type

Specifies the type of APDU that you are sending. Set this parameter to ROSE_C_ERROR.

The ROSE interface uses the value in this parameter to check that you have included all the mandatory parameters for the routine you are calling.

invoke_id

The Invoke identifier of the instance of the Invoke function to which you are responding.

local_value

The address of a locally defined operation code.

Set the *pointer* field to the address of an ASN.1 encoded integer representing an operation code and the *size* field to the length of the encoded integer.

Note

The *local_value* and *global_value* parameters are mutually exclusive.

If you use the *local_value* parameter, you should set the *pointer* field of the *global_value* parameter to null and the *size* field to zero.

global_value

The address of a globally defined operation code.

Set the *pointer* field to the address of an ASN.1 encoded object identifier representing an operation code and the *size* field to the length of the encoded object identifier.

Note

The *global_value* and *local_value* parameters are mutually exclusive.

If you use the *global_value* parameter, you should set the *pointer* field of the *local_value* parameter to null and the *size* field to zero.

buffer

The address of a buffer structure that you can use to pass user information required by your application. For example, you can pass information explaining why the requested operation was not successful. This parameter is optional. You should set the parameter to null if you are not passing a user buffer to the ROSE interface.

If you supply a user buffer, but the buffer is not big enough for the encoded ROSE PCI, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSE PCI in the new buffer and chains it to the front of the buffer you supplied.

If you do not supply a user buffer, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSE PCI in the new buffer. The *buffer_ptr* and *data_ptr* parameters both point to the same location.

The ROSE interface modifies your user buffer to accommodate the ROSE PCI. Therefore, you should save a copy of any buffer you pass in this routine call, unless you are certain that you do not need to reference the unchanged buffer again.

osak_rose_status

Returns a status code specific to ROSE in the *osak_status2* field.

Description

This routine encodes ROSE PCI that reports the failure of an operation to the client that requested the operation.

Return Value

A value indicating the status of the routine. Possible status values are:

OSAK_S_NORMAL	Routine has finished without error.
OSAK_S_BADPARAM	At least one mandatory parameter is missing.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVPARAM	At least one parameter is incorrectly specified.

osak_ro_reject_u

osak_ro_reject_u — Encodes ROSE PCI that rejects an operation request from a peer entity.

Syntax

```
status = osak_ro_reject (port, rose_pb)
```

Argument	Data Type	Access
port	osak_port	read only

Argument	Data Type	Access
rose_pb	osak_rose_pb	read only

Parameters	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
pdu_type	Unsigned longword	write only
invoke_id	osak_mem_descriptor	read only
reason	osak_ro_reason	read only
problem	osak_ro_problem	read only
[buffer]	osak_buffer	read only
[osak_rose_status]	osak_status_block	write only

C Binding

```
osak_ro_reject(port, rose_pb)

osak_port port;
struct osak_rose_pb *rose_pb;
```

Parameters Used

pb_length

The length of the ROSE parameter block, including the length of the working space.

ws_length

The length of the working space contained in the ROSE parameter block. The length of the working space should be 8 octets.

pdu_type

Specifies the type of APDU that you are sending. Set this parameter to ROSE_C_REJECT.

The ROSE interface uses the value in this parameter to check that you have included all the mandatory parameters for the routine you are calling.

invoke_id

The Invoke identifier of the instance of the Invoke function to which you are responding.

reason

The address of a reason code. Use this parameter to tell the client why you are rejecting an APDU.

Use one of the following constants as the reason code. You can assign whatever meaning you choose to these constants.

RORJ_C_INVOKE_PROB
RORJ_C_RET_RES_PROB

RORJ_C_RET_ERR_PROB

problem

The address of a problem code that tells the client more about why you are rejecting the APDU.

For each reason code, there is a set of problem codes, shown in Table 3.1. You can assign whatever meaning you choose to these problem codes.

Table 3.1. Problem Codes

Reason Code	Corresponding Problem Codes
RORJ_C_INVOKE_PROB	RORJ_C_DUPLIC_INV
	RORJ_C_UNREC_OPER
	RORJ_C_MISTYPE_ARG
	RORJ_C_RES_LIMIT
	RORJ_C_INIT_REJ
	RORJ_C_UNREC_LINK
	RORJ_C_RESP_UNEXP
	RORJ_C_UNEXP_CHILD_OPER
RORJ_C_RET_RES_PROB	RORJ_C_UNREC_INV
	RORJ_C_RES_RESP_UNEXP
	RORJ_C_MISTYPE_RES
RORJ_C_RET_ERR_PROB	RORJ_C_UNREC_INV
	RORJ_C_ERR_RESP_UNEXP
	RORJ_C_UNREC_ERR
	RORJ_C_UNEXP_ERR
	RORJ_C_MISTYPE_PAR

buffer

The address of a buffer structure you can use to pass user information required by your application. You should set this parameter to null if you are not passing a user buffer to the ROSE interface.

If you supply a user buffer, but the buffer is not big enough for the encoded ROSE PCI, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSE PCI in the new buffer and chains it to the front of the buffer you supplied.

If you do not supply a user buffer, the ROSE interface uses your OSAK memory allocation routine to create a new user buffer. The interface places the ROSE PCI in the new buffer. The *buffer_ptr* and *data_ptr* parameters both point to the same location.

The ROSE interface modifies your user buffer to accommodate the ROSE PCI. Therefore, you should save a copy of any buffer you pass in this routine call, unless you are certain you do not need to reference the unchanged buffer again.

osak_rose_status

A status code specific to ROSE, returned in the *osak_status2* field.

Description

This routine reports the rejection of a request to perform an operation. You should use this routine if an incoming APDU carrying an Invoke indication is in some way incorrect or badly structured, so that it does not make sense to your application.

The Reject function disrupts all other ROSE functions.

Return Value

A value indicating the status of the routine. Possible status values are:

OSAK_S_NORMAL	Routine has finished without error.
OSAK_S_BADPARAM	At least one mandatory parameter is missing.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVPARAM	At least one parameter is incorrectly specified.

osak_ro_decode

osak_ro_decode — Decodes ROSE PCI.

Syntax

status = osak_ro_decode (port, rose_pb)

Argument	Data Type	Access
port	osak_port	read only
rose_pb	osak_rose_pb	read only

Parameters	Data Type	Access
pb_length	Unsigned long integer	read only
ws_length	Unsigned long integer	read only
pdu_type	Unsigned longword	write only
invoke_id	osak_mem_descriptor	read only
linked_id	osak_mem_descriptor	read only
local_value	osak_mem_descriptor	read only
global_value	osak_mem_descriptor	read only
reason	osak_ro_reason	read only
problem	osak_ro_problem	read only
buffer	osak_buffer	read only
osak_rose_status	osak_status_block	write only

C Binding

```
osak_ro_decode(port, rose_pb)
```

```
osak_port port;
struct osak_rose_pb *rose_pb;
```

Parameters Used

pb_length

The length of the ROSE parameter block, including the length of the working space.

ws_length

The length of the working space contained in the ROSE parameter block. The length of the working space should be 8 octets.

pdu_type

The type of APDU being decoded. The value is one of the following:

- ROSE_C_INVOKE

- ROSE_C_RESULT

- ROSE_C_ERROR

- ROSE_C_REJECT_U

  This value indicates a rejection detected by the user.

- ROSE_C_REJECT_P

  This value indicates a rejection detected by the provider.

invoke_id

The address of an Invoke identifier. You should set the *pointer* field of this descriptor to null, and the *size* field to zero.

If the ROSE interface finds an Invoke identifier in the incoming APDU, it returns the address of that identifier, and a value specifying its size.

linked_id

The address of a Linked identifier. You should set the *pointer* field of this descriptor to null, and the *size* field to zero.

If the ROSE interface finds a Linked identifier in the incoming APDU, it returns the address of that identifier, and a value specifying its size.

local_value

The address of an ASN.1 encoded integer representing an operation code.

You should set the *pointer* field of this descriptor to null, and the *size* field to zero. If the ROSE interface finds a local value in the incoming APDU, it returns the address of that local value, and a value specifying its size.

global_value

The address of an ASN.1 encoded object identifier representing an operation code.

You should set the *pointer* field of this descriptor to null, and the *size* field to zero. If the ROSE interface finds a global value in the incoming APDU, it returns the address of that global value, and a value specifying its size.

reason

The address of a reason code. You should set this parameter to null.

If the ROSE interface finds a reason code in the incoming APDU, it returns the address of that code. The code has one of the following values:

Value	Meaning
RORJ_C_RETURN_PARAMETER	Problem detected by the provider
RORJ_C_GEN_PROB	Problem detected by the provider
RORJ_C_INVOKE_PROB	Meaning assigned by your application
RORJ_C_RET_RES_PROB	Meaning assigned by your application
RORJ_C_RET_ERR_PROB	Meaning assigned by your application

problem

The address of a problem code. You should set this parameter to null.

If the ROSE interface finds a problem code in the incoming APDU, it returns the address of that problem code. The problem code has one of the following values:

Reason	Corresponding Problem Codes
Returned parameter	No problem code
General problem	RORJ_C_UNREC_APDU
	RORJ_C_MISTYPE_APDU
	RORJ_C_BAD_STRCT_APDU
Invoke problem	RORJ_RO_DUPLIC_INV
	RORJ_C_UNREC_OPER
	RORJ_C_MISTYPE_ARG
	RORJ_C_RES_LIMIT
	RORJ_C_INIT_REJ
	RORJ_C_UNREC_LINK
	RORJ_C_RESP_UNEXP
	RORJ_C_UNEXP_CHILD_OPER
Return result problem	RORJ_C_UNREC_INV
	RORJ_C_RES_RESP_UNEXP
	RORJ_C_MISTYPE_RES
Return error problem	RORJ_C_UNREC_INV
	RORJ_C_ERR_RESP_UNEXP
	RORJ_C_UNREC_ERR

Reason	Corresponding Problem Codes
	RORJ_C_UNEXP_ERR
	RORJ_C_MISTYPE_PAR

Each of these values has the meaning assigned to it by your application.

buffer

The address of a buffer structure that holds the ROSE PCI and any user data that the ROSE interface finds in the incoming APDU.

If the ROSE interface successfully decodes the ROSE PCI, the *data_length* and *data_ptr* parameters are both set to point to the start of the undecoded ROSE user data.

When a provider reject occurs, the reason code is RORJ_C_RETURN_PARAM. This means that the ROSE provider has detected an error, and has set the buffer pointer to the address of the ROSE primitive that is in error.

osak_rose_status

A status code specific to ROSE in the *osak_status2* field.

Description

This routine decodes incoming ROSE PCI. The routine does not examine and remove the PDU part of the APDU. Your application should do this.

The routine does not check that the *invoke_id* parameter in the APDU is unique. Your application should check this.

Return Value

A value indicating the status of the routine. Possible status values are:

OSAK_S_NORMAL	Routine has finished without error.
OSAK_S_BADPARAM	At least one mandatory parameter is missing.
OSAK_S_INSFMEM	There is not enough dynamic memory.
OSAK_S_INVPARAM	At least one parameter is incorrectly specified.
OSAK_S_NOBUFFER	No user buffer supplied.

Chapter 4. Trace Emitter Routines

This chapter describes the OSAK trace emitter routines you can use to enable the OSAK trace emitter. Use these routines to output a trace record to a trace binary file.

osak_trace_dcs_verify

osak_trace_dcs_verify — Checks the DCS and default context maintained by the OSAK trace utility.

Syntax

status = osak_trace_dcs_verify (port, dcs, dflt_ctxt)

Argument	Data Type	Access
port	osak_port	read only
dcs	osak_dcs_pcontext	read only
dflt_ctxt	osak_default_context	read only

C Binding

```
osak_trace_dcs_verify (port, dcs, dflt_ctxt )

osak_port port;
struct osak_dcs_pcontext *dcs;
struct osak_default_context *dflt_ctxt;
```

Arguments

port

Port on which the connection you are tracing is established.

dcs

Address of a linked list of presentation contexts. The OSAKtrace utility uses this list of presentation contexts to verify the DCS.

Set this parameter to null if you call the `osak_trace_dcs_verify` routine after a session connection has been established.

dflt_ctxt

Address of the default context.

Set this parameter to null if you call the `osak_trace_dcs_verify` routine after a session connection has been established.

Description

This function writes a DCS verification record to the trace binary file for the specified connection. When decoding the trace binary file, the OSAK trace utility checks that it is using correct values for the

DCS and the default context on a connection. The utility checks the values it holds for the DCS and the default context against the values you supply in this routine call. This checking is done off-line when you activate the OSAK trace utility.

The OSAK trace utility compares the DCS verification record with the DCS contents and with the default presentation context. If the verification record matches the DCS and the default presentation context, the OSAK trace utility outputs the DCS contents and the default context.

If the verification record does not match the DCS and the default presentation context, the OSAK trace utility outputs both sets of values.

Return Value

OSAK_S_NORMAL	Normal execution of routine completed.
OSAK_S_BADPARAM	DCS is empty and the OSAK state machine is active or DCS is not empty and the OSAK state machine is idle.
OSAK_S_FILEERR	Error occurred in writing to the trace binary file.
OSAK_S_INSFMEM	Memory allocation error.
OSAK_S_INVFUNC	You have not opened a trace file, or you have not enabled tracing, for the connection.
OSAK_S_INVPORT	You have specified an invalid port.
OSAK_S_UNSPECERR	Error in some unspecified part of the system, for example an error in the *gettimeofday()* function.

osak_trace_close

osak_trace_close — Closes a trace binary file.

Syntax

status = osak_trace_close (port)

Argument	Data Type	Access
port	osak_port	read only

C Binding

```
osak_trace_close (port)

osak_port port;
```

Argument

port

Port on which the connection you are tracing is established.

Description

This routine closes the trace binary file for the connection on the port you specify. If you do not call the `osak_trace_stop` routine, this routine (`osak_trace_close`) calls `osak_trace_stop` to write a trace stop record to the binary file, and then closes the binary file.

A call to this routine should be paired with a call to `osak_trace_open`.

The routine is valid only in the context of an established connection. In your application program, it should come after a call to `osak_open_initiator`, `osak_open_responder`, or `osak_open_redirect`, so that the *port* parameter is valid.

Return Value

OSAK_S_NORMAL	Normal execution of routine completed.
OSAK_S_FILEERR	Unable to close trace binary file.
OSAK_S_INVFUNC	You have already closed the trace file on this connection.
OSAK_S_INVPORT	You have specified an invalid port.

See Also

- osak_trace_open

osak_trace_open

osak_trace_open — Opens a trace binary file.

Syntax

`status = osak_trace_open = (port, in_filename, out_filename)`

Argument	Data Type	Access
port	osak_port	read only
in_filename	osak_mem_descriptor	read only
[out_filename]	osak_mem_descriptor	modify

C Binding

```
osak_trace_open (port, in_filename, out_filename)

osak_port port;
osak_mem_descriptor *in_filename;
osak_mem_descriptor *out_filename;
```

Arguments

port

Port on which the connection you want to trace is established.

in_filename

Address of the name of the trace binary file. The file name can consist of any number of ASCII characters.

out_filename

Address of the actual name of the file containing the trace.

Description

This routine opens a trace binary file for the connection on the port you specify. The OSAK interface uses the file name you supply in the *in_filename* parameter, and makes the name unique. The OSAK interface does this by adding some digits derived from the time stamp and the process identifier. This protects the trace binary file from being accidentally overwritten. The resultant file name is returned in the *out_filename* parameter. If the *out_filename* parameter is specified, it must point to a buffer at least 20 bytes larger than buffer used for *in_filename*. The interface returns the actual length of the resultant file name in the size field in the *out_filename* parameter. If the *out_filename* parameter is null, the name specified by the *in_filename* parameter is used without modification. If you set the *out_filename* to null, you must ensure that only one port writes to the file. If more than one port tries to write to the file, the results are unpredictable.

The call to this routine should be paired with a call to `osak_trace_close`.

The call is valid only in the context of an established connection. In your application program, it should come after a call to `osak_open_initiator`, `osak_open_responder`, or `osak_open_redirect`, so that the *port* parameter is valid.

Returned Value

OSAK_S_NORMAL	Normal execution of routine completed.
OSAK_S_BADPARAM	DCS is empty and the OSAK state machine is active or DCS is not empty and the OSAK state machine is idle.
OSAK_S_FILEERR	Error occurred in opening the trace binary file.
OSAK_S_INSFMEM	Memory allocation error.
OSAK_S_INVFUNC	A trace file is already open for this connection.
OSAK_S_INVPORT	You have specified an invalid port.

See Also

• osak_trace_close

osak_trace_start

osak_trace_start — Starts tracing for the specified port.

Syntax

```
status = osak_trace_start (port, dcs, dflt_ctxt)
```

Argument	Data Type	Access
port	osak_port	read only
dcs	osak_dcs_pcontext	read only
dflt_ctxt	osak_default_context	read only

C Binding

```
osak_trace_start (port, dcs, dflt_ctxt )

osak_port port;
struct osak_dcs_pcontext *dcs;
struct osak_default_context *dflt_ctxt;
```

Arguments

port

Port on which the connection you want to trace is established.

dcs

Address of a linked list of presentation contexts. These presentation contexts are the members of the DCS.

When you call `osak_trace_start`, you should observe the following rules for setting this parameter:

- Set the parameter to null if you call `osak_trace_start` immediately after `osak_open_initiator`, `osak_open_responder`, or `osak_open_redirect`, that is, if you call it before issuing any calls to `osak_associate_req`, `osak_select`, or `osak_give_buffers`.

- Specify a non-null value if you call `osak_trace_start` after calling `osak_associate_req` or `osak_get_event`.

dflt_ctxt

Address of the default presentation context. If you call `osak_trace_start` after any exchange of protocol on a port, you should observe the same rules for setting this parameter as for setting the *dcs* parameter.

Description

This routine enables trace records to be written to the trace binary file that you have set up for a connection. The routine also sends a set DCS record to the trace binary file in order to initialize the DCS contents and the default context.

The routine should come after a call to `osak_trace_open`, and before a call to `osak_trace_close` in your application program.

If the association was already established, or is in the process of being established when you call `osak_trace_start`, the OSAK trace utility cannot build its own table of information about the DCS and the default context. Therefore, if you call `osak_trace_start` after opening a port,

you should pass the DCS and default context information to the trace utility in the *dcs* and *dflt_ctxt* parameters.

Returned Value

OSAK_S_NORMAL	Normal execution of routine completed.
OSAK_S_BADPARAM	DCS is empty and the OSAK state machine is active or DCS is not empty and the OSAK state machine is idle.
OSAK_S_FILEERR	Error occurred in writing to the trace binary file.
OSAK_S_INSFMEM	Memory allocation error.
OSAK_S_INVFUNC	There is no open trace file, or you have already enabled tracing, for this connection.
OSAK_S_INVPORT	You have specified an invalid port.
OSAK_S_UNSPECERR	Error in some unspecified part of the system, for example an error in the *gettimeofday()* function.

See Also

- osak_trace_stop

osak_trace_stop

osak_trace_stop — Stops the trace utility.

Syntax

status = osak_trace_stop = (port)

Argument	Data Type	Access
port	osak_port	read only

C Binding

```
osak_trace_stop (port)

osak_port port;
```

Argument

port

Port on which you wish to stop tracing.

Description

This routine stops tracing on the port you specify, and sends a trace stop record to the trace binary file.

The routine is invalid if it is not preceded by a call to `osak_trace_start` in your application program.

Return Value

OSAK_S_NORMAL	Normal execution of routine completed.
OSAK_S_FILEERR	Error occurred in writing to the trace binary file.
OSAK_S_INVFUNC	You have not opened a trace file, or you have not enabled tracing, for the connection.
OSAK_S_INVPORT	You have specified an invalid port.
OSAK_S_UNSPECERR	Error in some unspecified part of the system, for example an error in the *gettimeofday()* function.

See Also

• osak_trace_start

Chapter 5. How OSAK Calls Map to Protocol Messages

Table 5.1 shows the mapping between OSI Applications Kernel (OSAK) calls and Association Control Service Element (ACSE), presentation, and session protocol messages. The calls are arranged in alphabetical order.

Table 5.1. Mappings Between OSAK Routines and Protocol Messages

Call	ACSE	Presentation	Session
osak_abort_req	ABRT	ARU/ARP	AB
osak_accept_rsp	AARE+ve	CPA	AC
osak_act_discard_req	-	-	AD
osak_act_discard_rsp	-	-	ADA
osak_act_end_req	-	-	AE
osak_act_end_rsp	-	-	AEA
osak_act_interrupt_req	-	-	AI
osak_act_interrupt_rsp	-	-	AIA
osak_act_resume_req	-	-	AR
osak_act_start_req	-	-	AS
osak_alter_req	-	AC	TD
osak_alter_rsp	-	ACA	TD
osak_associate_req	AARQ	CP	CN
osak_capability_req	-	-	CD
osak_capability_rsp	-	-	CDA
osak_close_port	-	-	-
osak_control_give_req	-	-	GTC
osak_data_req	-	-	DT
osak_exception_req	-	-	ED
osak_expedited_req	-	-	EX
osak_get_event	-	-	-
osak_get_handle	-	-	-
osak_give_buffers	-	-	-
osak_minor_req	-	-	MIP
osak_minor_rsp	-	-	MIA
osak_major_req	-	-	MAP
osak_major_rsp	-	-	MAA
osak_open_initiator	-	-	-
osak_open_redirect	-	-	-
osak_open_responder	-	-	-

Call	ACSE	Presentation	Session
osak_collect_pb	-	-	-
osak_redirect	-	-	-
osak_reject_rsp	AARE-ve	CPR	RF
osak_release_req	RLRQ	-	FN
osak_release_rsp	RLRE	-	DN/NF
osak_resync_req		RS	RS
osak_resync_rsp	-	RSA	RA
osak_select	-	-	-
osak_send_more			SPDU
osak_token_give_req	-	-	GT
osak_token_please_req	-	-	PT
osak_typed_req	-	-	TD

Chapter 6. Checking OSAK Status Codes

This chapter lists the status codes returned by the OSAK routines. The status codes are divided into the following categories:

- Success (Section 6.1)

- Informational (Section 6.2)

- Error (Section 6.3)

Subsidiary status codes occur only in the *osak_status2* field of the *status_block* parameter.

All the status codes except OSAK_S_INVPORT can be returned either as return values or in the *status_block* parameter. The status code OSAK_S_INVPORT can only be returned as a return value.

To find the status of a call, follow these guidelines:

- In the following circumstances, you need to check only the return value of the call:

  - You are sending unsegmented user data, or you are sending a final segment of user data in a call to `osak_send_more` with the *more_flag* parameter set to false.

  - You are not using completion routines (OpenVMS systems only).

  - The return value of the call is not OSAK_S_QUEUED or OSAK_S_FREE.

- In the following circumstances, you need to check the return value of the call and the value in the *status_block* parameter:

  - You are sending segmented data.

  - You are using completion routines (OpenVMS systems only).

  - The return value of the call is OSAK_S_QUEUED or OSAK_S_FREE.

Note

You cannot check the value in the *status_block* parameter until the OSAK interface returns the ownership of the parameter block to the application.

6.1. Success Status Codes

OSAK_S_NORMAL, the routine has finished without error

Usually indicates that the OSAK interface has delivered the call to the local transport provider and has returned ownership of the parameter block and user data buffers to your application. However, the status code has specific meanings in certain cases:

- A call to `osak_close_port` returns this status code when the OSAK interface has closed the port specified in the call.

- A call to `osak_get_event` returns this status code when an event has arrived on the association specified in the call.

- A call to `osak_get_handle` returns this status code when the OSAK interface has identified the channel or channels specified in the call.

- A call to `osak_give_buffers` returns this status code when user data buffers have been passed to OSAK.

- A call to `osak_open_initiator`, `osak_open_responder`, or `osak_redirect` returns this status code when the OSAK interface has opened a port.

- A call to `osak_collect_pb` returns this status code when a parameter block used on an outbound service is being returned.

- A call to `osak_select` returns this status code when an event is present on the channel or channels specified in the call.

6.2. Informational Status Codes

OSAK_S_FREE, OSAK has queued the request and there are free parameter blocks

A previous request has been completed and there are one or more parameter blocks or user data buffers awaiting collection. If you want to reuse these parameter blocks and user data buffers, call `osak_collect_pb`.

The code means the same as OSAK_S_QUEUED, with the added indication that a call to `osak_collect_pb` will generate an OSAK_S_NORMAL return rather than an OSAK_S_NOEVENT.

OSAK_S_NOEVENT, no event has occurred

A call to `osak_get_event` returns this status code when there is no event waiting to be received on the specified association. The *event_type* parameter contains the value OSAK_C_NOEVENT.

OpenVMS

> This status code is also returned when you call `osak_get_event` with a completion routine and then release or abort the association on which the call is made. The completion routine starts running when the association closes.

A call to `osak_collect_pb` returns this status code when there are no completed outbound services on the specified association, and hence no parameter blocks waiting to be collected.

A call to `osak_select` returns this status code when there are no events waiting on any of the channels specified in the call and the timeout period has expired.

OSAK_S_QUEUED, OSAK has queued the request

The OSAK interface has put the call on the queue for the transport provider. The OSAK interface retains ownership of the parameter block and the user data buffers passed in the call. This status code can be

returned by the routine `osak_associate_request`, or when you are sending segmented user data using the routine `osak_send_more`.

OpenVMS

This status code can also be returned when you are using a completion routine, and on all outbound services.

The code OSAK_S_QUEUED does not indicate that the OSAK interface has sent the data unit. Errors can occur after the interface has put the call on the queue for the transport provider. You should check the *status_block* parameter for such errors, but you cannot do this until the OSAK interface returns the ownership of the parameter block and user data buffers.

Note that in general service requests on the queue for the transport provider are completed in the order in which they are issued, but there are exceptions. For example, a call to the expedited data service may overtake a call to the normal data service.

OpenVMS

OSAK_S_QUEUED is returned by the following routines only when they include a completion routine:

- `osak_get_event`

- `osak_open_initiator`

- `osak_open_redirect`

- `osak_open_responder`

If no completion routine is included on call to one of these routines, the routine can return only OSAK_S_NORMAL or an error status.

6.3. Error Status Codes

OSAK_S_BADPARAM, there is an invalid parameter

One or more of the parameters you have used in a call was invalid. This code is rarely returned.

OSAK_S_DEALLOCERR, an error occurred when deallocating memory

A user-supplied deallocation routine has returned an error status. You can find the returned error status in the *status_block* parameter.

OSAK_S_DISRUPTED, a disruptive event has occurred

The request was canceled by a disruptive event. OSAK_S_DISRUPTED gives information about what to do when a disruptive event occurs.

OSAK_S_FILERR, an error occurred in opening the trace file

Returned when a call to `osak_trace_open` fails. Check the validity of the parameters that you passed in the call.

OSAK_S_INCPCI, incomplete PCI

Returned by `osak_get_event` when a partial event arrives with the protocol control information (PCI) incomplete. You should make another call to `osak_get_event` to collect the remaining PCI.

OSAK_S_INSFMEM, there is not enough dynamic memory

There is not enough dynamic memory to complete the service request. This error is usually fatal to the association on which it occurs. You should abort the connection. Check your memory allocation and deallocation routines (see *VSI DECnet-Plus OSAK Programming*). You may find that you have allocated more memory than you need.

If you cannot make additional memory available, you should shut down your application. Aborting a connection will also make some memory free.

OSAK_S_INSFWS, there is not enough workspace in the parameter block

The parameter block workspace is not large enough. You should make the workspace at least double its existing size. The chapter on planning in *VSI DECnet-Plus OSAK Programming* helps you decide how large the workspace should be.

OSAK_S_INVACTION, the action_result parameter is invalid

Returned by `osak_release_rsp`.

Check which functional units are in use on this association. You should not pass the *action_result* parameter in a call to `osak_release_rsp` unless you are using the negotiated release functional unit on the association.

OSAK_S_INVADDR, Invalid address for open redirect

A call to `osak_open_redirect` returns this status if the call is to a passive address that has not been registered with OSAKserver. For a discussion of passive addresses and OSAKserver, see Appendix A.

OSAK_S_INVAEI, the application entity invocation is invalid

There is an error in at least one address component. For example, the *tsel* value you are using may be unknown at the remote end of the connection.

The *osak_status2* field of the *status_block* parameter may contain a subsidiary status code that tells you which address component is invalid. The following subsidiary codes may occur with OSAK_S_INVAEI:

- OSAK_S_INVSSEL, invalid session selector

    The session selector is too long. A session selector should not be longer than 16 octets.

- OSAK_S_MULTADDR, multiple upper layer addresses for t-selector

  Indicates that you have opened more than one process (initiator or responder) using the same TSEL, but a different SSEL or PSEL. This is not allowed. If you want to specify a different SSEL or PSEL, you should also specify a different TSEL.

- OSAK_S_NOSUCHENTRY, No such entry in OSAKserver address database

  A call to `osak_open_redirect` returns this status if the address specified for the call has not been created on the server.

- OSAK_S_NOTAVAILABLE, OSAK is not available

  A call to `osak_open_initiator`, `osak_open_responder` or `osak_redirect` returns this status code when OSAK has been restricted, disabled, or deleted by OSAK network management (by means of NCL).

- OSAK_S_TSELINUSE

  Indicates that a TSEL in the *local_aei* or *calling_aei* parameter is already being used on another port or by another application.

- OSAK_S_INVTSEL

  Indicates that the TSEL is not known at the remote node.

OSAK_S_INVAPIVERSION, unsupported API version

Check that you specified the correct constant in the *api_version* parameter. Chapter 1 discusses the correct constant for this version.

OSAK_S_INVDEFCTXT, the default context response is invalid

Examine the *pdefault_context* parameter. You should specify an abstract syntax and a transfer syntax in a default context proposal.

OSAK_S_INVFUNC, the call is invalid

This code is returned for one of the following reasons:

- You have made an incorrect sequence of calls. For example:

  - Sending data before establishing an association

  - Calling `osak_send_more` when the *more_flag* parameter of the previous call is false

  - Calling an activity management service when the activity management functional unit is not selected

  - Calling `osak_get_event` on an association that has been terminated

- You are trying to enable tracing using (`osak_trace_open` or `osak_trace_start`) when you have already enabled tracing either by defining *osak_trace* or by making a previous `osak_trace_open` call (with no intervening `osak_trace_close` call).

Call `osak_trace_close` and `osak_trace_stop` to close the existing trace files and to stop tracing. After this, call `osak_trace_open` and `osak_trace_start` again.

The following subsidiary code may occur with OSAK_S_INVFUNC:

OSAK_S_READPOSTED, buffers have been given to OSAK

This code can be returned only by `osak_redirect`. It indicates that the OSAK interface holds unused buffers passed from your application before the call to `osak_redirect`. You cannot redirect an association in this situation. You should reclaim the buffers and make the call to `osak_redirect` again.

OSAK_S_INVFUS, the functional units are invalid

You have proposed an invalid combination of functional units. Examine the *functional_units* parameter. *VSI DECnet-Plus OSAK Programming* explains which functional units are interdependent.

OSAK_S_INVID, the activity identifier is too long

Examine the *activity_id* and *old_activity_id* parameters. Neither of these parameters should be more than six characters long.

OSAK_S_INVPARAM, there is an invalid parameter

Returned by `osak_give_buffers` or `osak_select`.

OSAK_S_INVPARAM is returned by `osak_give_buffers` if at least one of the buffers you have passed to the OSAK interface is less than the minimum permitted size for buffers, 512 octets. This minimum size applies only to buffers that you pass to OSAK for receiving inbound events.

You should increase the size of any buffer that is smaller than 512 octets.

OSAK_S_INVPARAM is returned by `osak_select` if:

- The *time_out* parameter is greater than 86,400 seconds (one day)

- The specified event flag is not in event flag cluster 1

OSAK_S_INVPCTXT, the presentation context list is invalid

Examine the *pcontext_id_list*, *pcontext_list*, or *pcontext_res_list* parameter (depending on which you are using). The ACSE abstract syntax and the transfer syntax should be proposed as object identifier values according to *ISO Standard 8823*.

You should check that you have followed these rules:

- You have not used the *pcontext_id_list* parameter with session version 1 in a call to `osak_abort_req`.

- You have not used the *pcontext_id_list* parameter with any kind of abort except a presentation user abort.

- You have not used the *pcontext_id_list* parameter in a call to either `osak_resync_req` or `osak_resync_rsp` if you have not selected the context management functional unit.

- You have used the *pcontext_list* parameter or the *pcontext_del_list* parameter, or both, in a call to `osak_alter_req` or `osak_alter_rsp`.

OSAK_S_INVPORT, the port identifier is invalid

Examine the port argument. You have specified a port that does not exist.

OSAK_S_INVPV, the protocol versions are invalid

The *protocol_versions* parameter contains illegal values.

OSAK_S_INVREASON, the reason code is invalid

Examine the *reason* parameter. You have specified a reason code that is not valid for the service you are using it on. Chapter 10 lists the possible reason codes and their constant values.

OSAK_S_INVRESYNCTYPE, the resynchronization type isinvalid

Returned by `osak_resync_req`.

Examine the *resync_type* parameter. There are only three valid resynchronization types: *abandon*, *restart*, and *set*.

OSAK_S_INVSCONNID, invalid session connection identifier

One of the fields in the *sconnect_id* or the *old_sconnection_id* contains an invalid value. The size restrictions on the fields in these parameters are as follows:

Field	Maximum Size
ss_user_ref	64 octets
common_ref	64 octets
add_ref_info	4 octets
called_ss_user_ref	64 octets
calling_ss_user_ref	64 octets

OSAK_S_INVSSEL, Invalid session selector

This code may be returned as a secondary status in the *osak_status2* field when the primary status is OSAK_S_INVAEI. See under OSAK_S_INVAEI.

OSAK_S_INVSYNCPNT, the synchronization point serial number is invalid

Returned by `osak_resync_req`.

Examine the *sync_point* and *resync_type* parameters. You should not specify a value for a synchronization point serial number if the resynchronization type is *abandon*.

OSAK_S_INVTEMPLATE, invalid transport template

This status is returned if the transport template parameter specified does not contain a valid transport template name.

OSAK_S_INVTOKEN, the token setting is invalid

Examine the *token_item* and *request_tokens* parameters. The token setting is illegal. For example:

- A peer entity is requesting a token it already has.

- A peer entity is requesting a token whose supporting functional unit is not selected on this association.

OSAK_S_INVTSEL, Invalid transport selector

This code may be returned as a secondary status in the *osak_status2* field when the primary status is OSAK_S_INVAEI. See under OSAK_S_INVAEI.

OSAK_S_MULTADDR, multiple upper layer addresses for one T-selector

This code may be returned as a secondary status in the *osak_status2* field when the primary status is OSAK_S_INVAEI. See under OSAK_S_INVAEI.

OSAK_S_NOBUFFERS, there are not enough user data buffers

This status is returned by `osak_get_event`. The code indicates that the OSAK interface needs more buffer space in which to return an incoming event to your application. Increase the number or the size of user buffers you are supplying. Note that the incoming event is not lost. The OSAK interface returns the event to your application when sufficient buffers are available.

If the OSAK interface receives a partial event and runs out of buffers before receiving enough data units to decode the event, the interface retains the buffers it is holding. The interface retains these buffers until you post the necessary extra buffers, or until you abort the connection.

If the OSAK interface receives a partial event and has both decoded it and passed it to the application, but has no buffers to receive the rest of the event, the application owns the buffers it has received from the OSAK interface.

VSI DECnet-Plus OSAK Programming explains how to plan the buffer capacity you require.

OSAK_S_NOCTXTNAME, the application context name is missing

The *acontext* parameter is mandatory in calls to `osak_associate_req`. Check that you have included this parameter in the parameter block.

OSAK_S_NODATA, there is no data specified

Returned by `osak_data_req`, `osak_typed_req`, `osak_expedited_req`, and `osak_capability_req`. This status is returned if no data is specified and the *more_flag* is set to false.

OSAK_S_NOPARAM, a mandatory parameter has been omitted in the call

Make the call again, including the missing parameter.

OSAK_S_NOPROCINFO, there is no process identifier and no process name

Returned by `osak_redirect`.

Examine the *process-id* and *process_name* parameters. In a request for redirection of an association, at least one of these parameters should contain a value other than null.

OSAK_S_NORESOURCE, OSAK has run out of system resources

Examine the *transport_status1* field of the *status_block* parameter for more specific information on the system error. The remedial action depends on your local situation.

OSAK_S_NORFC1006, RFC 1006 not available on VMS

Calls on an OpenVMS system return this status when your transport template specifies RFC 1006 in the Transport layer.

OSAK_S_NOSERVER, there is no response from OSAKserver

OpenVMS

This status code indicates that OSAKserver is not available. This code occurs only when you are running an application developed with a previous version of the OSI Application Developer's Toolkit.

Because a passive application cannot work without OSAKserver, you should shut down the application and start OSAKserver.

UNIX

An application running on a UNIX or ULTRIX system cannot receive this status code.

OSAK_S_NOSUCHENTRY, No such entry in OSAKserver address database

This code may be returned as a secondary status in the field *osak_status2* when the primary status is OSAK_S_INVAEI. See under OSAK_S_INVAEI.

OSAK_S_NOSYNCPNT, the synchronization point serial number is missing

This status code is returned by `osak_associate_req` and `osak_accept_rsp`.

Check that you assigned a value to the *initial_serial_number* parameter. This parameter is mandatory if you have selected the major synchronize functional unit, the minor synchronize functional unit, or the resynchronize functional unit, but you have not selected the activity management functional unit.

OSAK_S_NOTAVAILABLE, OSAK is not available

This code may be returned as a secondary status in the field *osak_status2* when the primary status is OSAK_S_INVAEI. See under OSAK_S_INVAEI.

OSAK_S_NOTRANSPORT, there is no transport connection setup

You made an inappropriate call before a transport connection is established. For example, you may have called `osak_get_handle` before calling `osak_associate_req`.

OSAK_S_OVERFLOW, too much user data has been sent for session version 1

Send the data again, dividing it into smaller units. Alternatively, you can negotiate the session version again, proposing session version 2.

However, if you are using session version 1, you cannot send any userdata on the following calls:

```
osak_act_interrupt_req
osak_act_interrupt_rsp
osak_act_discard_req
osak_act_discard_rsp
osak_token_give_req
osak_control_give_req
```

These calls return the status code OSAK_S_OVERFLOW if you try to send any user data.

OSAK_S_READPOSTED, buffers have been given to OSAK

This code may be returned as a secondary status in the *osak_status2* field when the primary status is OSAK_S_INVFUNC. See under OSAK_S_INVFUNC.

OSAK_S_REDIRECTERR, error occurred while redirecting

This code can be returned only by `osak_redirect`. For further information on the nature of the error, check the *osak_status2* field of the *status_block* parameter. This field may contain either of the following secondary statuses:

- OSAK_S_TIMEOUT, redirect processing timed out

- OSAK_S_TOOMANYREDIRECTS, tried to exceed maximum number of simultaneous redirects

OSAK_S_TIMEOUT, Redirect processing timed out

See OSAK_S_REDIRECTERR.

OSAK_S_TOOMANYREDIRECTS, maximum number of simultaneous redirects exceeded

See OSAK_S_REDIRECTERR.

OSAK_S_TRANSERR, there is an error in the transport provider

An error has occurred in the Transport layer or at the interface to the Transport layer. The OSAK interface has returned ownership of the parameter block and user data buffers to your application.

The *transport_status* field in the *status_block* parameter records a transport provider status code that gives more information about the error. Note that transport errors are simply passed through by the OSAK software, so their meanings may vary between systems. For example, given a common cause of error, a message from XTI on an ULTRIX system may not be the same as the message from XTI on a UNIX system.

Examples of errors that can occur are:

- The remote system disconnects the transport connection.

- Someone shuts down the local transport provider.

- Network connectivity is lost.

- The transport provider receives an invalid PDU.

Some errors in the Transport layer are fatal only to the connection on which the error is returned. Others are fatal to all connections. No Transport layer error can be fatal to the OSI component using OSAK, or to the OSAK software itself. But if you receive many TRANSERR messages, there might be a problem with the Transport entity that needs attention.

For example, if someone shuts down the local transport provider, all connections are affected, and you should shut down your application until the transport provider is running again.

OSAK_S_TSELINUSE, T-selector is already in use

This code may be returned as a secondary status in the *osak_status2* field when the primary status is OSAK_S_INVAEI. See OSAK_S_INVAEI.

OSAK_S_UNSPECERR, an unspecified error has occurred

Indicates the occurrence of either an internal error that does not correspond to an OSAK interface error, or a system error. The *osak_status2* field of the *status_block* parameter contains the code for the internal error or system error.

Chapter 7. Disruptive Events

This chapter explains what the OSI Applications Kernel (OSAK) interface does when a disruptive event occurs and what action you should take, if any.

The following disruptive events may occur:

- ABORT request (from the local peer entity)

- ABORT indication (from the remote peer entity)

- Transport connection loss

- P-ACTIVITY-INTERRUPT indication

- P-ACTIVITY-DISCARD indication

- P-RESYNCHRONIZE indication

- P-EXCEPTION-REPORT indication

- PREPARE (RESYNC)

7.1. ABORT request (Local Abort)

This event is fatal to an association.

If you are using segmentation and you issue an ABORT request when you have a queue of data segments waiting to be sent, the OSAK interface does not send any of these segments. The OSAK interface returns the status code OSAK_S_DISRUPTED.

To reclaim the user data buffers, call `osak_close_port` or `osak_collect_pb`. Then set up the association and send the data again.

7.2. ABORT indication (Peer Abort)

This event is fatal to an association.

Data segments sent from the remote peer entity after it issues the ABORT request do not reach the local peer entity.

If the local peer entity is sending data when it receives the ABORT indication, the OSAK interface does not send any of the data segments that are on the queue for the transport provider.

The OSAK interface returns the status code OSAK_S_DISRUPTED.

7.3. Transport Connection Loss

This event is fatal to an association. The event appears as an ABORT indication.

If the local peer entity is sending data when it receives the ABORT indication, the OSAK interface does not send any of the data segments on the queue for the transport provider.

The OSAK interface returns status OSAK_S_DISRUPTED.

7.4. P-ACTIVITY-INTERRUPT indication

This event is not fatal to an association.

If you are sending data segments when a P-ACTIVITY-INTERRUPT indication arrives, stop sending the data. Data segments you have already sent are lost, so you should send all the data again.

If you are receiving data segments when a P-ACTIVITY-INTERRUPT indication arrives, keep the data segments you have already received until a P-ACTIVITY-RESUME indication arrives.

7.5. P-ACTIVITY-DISCARD indication

This event is not fatal to an association.

When you receive a P-ACTIVITY-DISCARD indication, ignore the data that has already arrived on the activity being discarded and reclaim the buffers you posted to receive it.

7.6. P-RESYNCHRONIZE indication

This event is not fatal to an association.

When you receive a P-RESYNCHRONIZE indication, keep data segments that arrived before the synchronization point specified in the indication. Ignore data segments received after that synchronization point.

7.7. P-EXCEPTION-REPORT indication

This event is not fatal to an association.

When you receive a P-EXCEPTION-REPORT indication, follow the procedures you have defined for dealing with exception reports.

7.8. PREPARE (RESYNC)

This event is not fatal to an association.

If you receive a PREPARE (RESYNC) indication from the remote peer entity when you are sending data, check the queue of data units waiting to be sent:

* If the head of the queue is the first segment of a data unit, stop sending.

* If the head of the queue is not the first segment of a data unit, continue sending until you have sent the complete data unit, then stop sending.

If you send a PREPARE (RESYNC) indication, do not send any data after it.

Chapter 8. Parameter Passing Mechanisms

Table 8.1 lists the parameter passing mechanism for each parameter in the OSI Applications Kernel (OSAK) parameter block. The parameters are arranged in alphabetical order.

Table 8.1. Parameter Passing Mechanisms

Parameter	Passing Mechanism
acontext	By reference
abort_reason	By value
action_result	By reference
activity_id	By value
actual_aeiid	By reference
alloc_rtn	By value
called_aei	By reference
calling_aei	By reference
completion_param	By value
completion_rtn	By value
data_length	By value
dealloc_rtn	By value
event_type	By value
exception_reason	By value
func	By value
functional_units	By reference
initial_serial_number	By reference
initial_tokens	By reference
local_aei	By reference
local_data	By reference
more_flag	By value
next_pb	By reference
old_activity_id	By value
old_sconnect_id	By value
pb_length	By value
pcontext_del_list	By reference
pcontext_del_res_list	By reference
pcontext_id_list	By reference
pcontext_list	By reference
pcontext_redirect_list	By reference
pcontext_res_list	By reference

Parameter	Passing Mechanism
pdefault_context	By reference
pdefault_context_res	By reference
port_id	By value
process_id	By reference
process_name	By reference
protocol_options	By reference
protocol_versions	By reference
rcv_data_list	By reference
activity_reason	By reference
reject_reason	By value
release_reason	By value
release_resp_reason	By value
request_tokens	By reference
responding_aei	By reference
resync_type	By value
status_block	By value
sconnect_id	By reference
segmentation	By reference
redirect_state	By value
sync_confirm	By value
sync_point	By reference
osul_template	By reference
token_item	By reference
tokens	By reference
tsdu_ptr	By reference
user_data	By reference
ws_length	By value

Chapter 9. How the OSAK Interface Implements the ISO Standards

This chapter explains how the OSAK interface implements the ISO Standards for Open Systems Interconnection, and the National Institute of Standards and Technology (NIST) modifications to these standards.

VSI DECnet-Plus OSAK Programming gives a full list of the standards on which the OSAK interface is based. It is important that you have access to copies of all the standards documents.

9.1. The OSAK Interface and the ISO Protocol Definitions

The OSAK interface conforms to the following ISO protocol definitions:

* ISO 8327 *Information Processing Systems — Open Systems Interconnection — Basic Connection Oriented Session Protocol Specification*

* ISO 8823 *Information Processing Systems — Open Systems Interconnection — Connection Oriented Presentation Protocol Specification*

* ISO 8650 *Information Processing Systems — Open Systems Interconnection — Protocol Specification for the Associated Control Service Element*

ISO service definitions are given in the following documents:

* ISO 8326 *Information Processing Systems — Open Systems Interconnection — Basic Connection Oriented Session Service Definition*

* ISO 8822 *Information Processing Systems — Open Systems Interconnection — Connection Oriented Presentation Service Definition*

* ISO 8649 *Information Processing Systems — Open Systems Interconnection — Service Definition for the Association Control Service Element*

With the exception of the items listed in Section 9.2. The OSAK interface implements these standards with the NIST modifications given in NIST Special Publication 500-177, *Stable Implementation Agreements for Open Systems Interconnection Protocols Version 3 Edition 1 December 1989*, as follows:

* The OSAK implementation of the sending side of a connection conforms to the NIST agreement. For example, you may send no more than 10,240 octets of data in the *user_data* parameter (except on calls to `osak_data_req` and `osak_typed_req`).

* The OSAK implementation of the receiving side of a connection conforms to the NIST agreement.

9.2. Restrictions in the OSAK Implementation of the ISO Protocol Definitions

The OSAK interface has the following restrictions:

- The OSAK interface does not support the following items:

    - The symmetric synchronize functional unit defined in Addendum 1 to *ISO 8327*

    - The context restoration functional unit defined in *ISO 8823*

    - The authentication functional unit defined in *ISO 8650*

    - The following presentation services, which are available instead through ACSE services:

        - P-CONNECT (available through AARQ)

        - P_RELEASE (available through RLRQ)

        - P-U-ABORT (available through ABRT)

Note

No service provided by ACSE is provided separately by the Presentation layer.

- When you propose a defined context set, you should include the ACSE abstract syntax.

    The ACSE abstract syntax consists of the ACSE PCI abstract syntax with BER as the transfer syntax.

Chapter 10. Possible Values for OSAK Data Types

This chapter lists the possible values of all the OSI Applications Kernel (OSAK) data types that have constant values. The data type names are in alphabetical order.

10.1. Data Type: osak_abort_ppdu

These values are defined in ISO 8823.

Constant	Corresponding Event Identifier
OSAK_C_ABORT_PPDU_CP	cp-PPDU
OSAK_C_ABORT_PPDU_CPA	cpa-PPDU
OSAK_C_ABORT_PPDU_CPR	cpr-PPDU
OSAK_C_ABORT_PPDU_ARU	aru-PPDU
OSAK_C_ABORT_PPDU_ARP	arp-PPDU
OSAK_C_ABORT_PPDU_AC	ac-PPDU
OSAK_C_ABORT_PPDU_ACA	aca-PPDU
OSAK_C_ABORT_PPDU_TD	TD-ppdu
OSAK_C_ABORT_PPDU_TTD	ttd-PPDU
OSAK_C_ABORT_PPDU_TE	te-PPDU
OSAK_C_ABORT_PPDU_TC	tc-PPDU
OSAK_C_ABORT_PPDU_TCC	tcc-PPDU
OSAK_C_ABORT_PPDU_RS	rs-PPDU
OSAK_C_ABORT_PPDU_RSA	rsa-PPDU
OSAK_C_ABORT_PPDU_SREL_IND	s-release indication
OSAK_C_ABORT_PPDU_SREL_CNF	s-release confirm
OSAK_C_ABORT_PPDU_SGT_IND	s-token-give indication
OSAK_C_ABORT_PPDU_SPT_IND	s-token-please indication
OSAK_C_ABORT_PPDU_SCG_IND	s-control-give indication
OSAK_C_ABORT_PPDU_SSYNMIN_IND	s-sync-minor indication
OSAK_C_ABORT_PPDU_SSYNMIN_CNF	s-sync-minor confirm
OSAK_C_ABORT_PPDU_SSYNMAJ_IND	s-sync-major indication
OSAK_C_ABORT_PPDU_SSYNMAJ_CNF	s-sync-major confirm
OSAK_C_ABORT_PPDU_SPER_IND	s-p-exception-report indication
OSAK_C_ABORT_PPDU_SUER_IND	s-u-exception-report indication
OSAK_C_ABORT_PPDU_SACTS_IND	s-activity-start indication
OSAK_C_ABORT_PPDU_SACTR_IND	s-activity-start indication
OSAK_C_ABORT_PPDU_SACTI_IND	s-activity-interrupt indication
OSAK_C_ABORT_PPDU_SACTI_CNF	s-activity-interrupt confirm

Constant	Corresponding Event Identifier
OSAK_C_ABORT_PPDU_SACTD_IND	s-activity-discard indication
OSAK_C_ABORT_PPDU_SACTD_CNF	s-activity-discard confirm
OSAK_C_ABORT_PPDU_SACTE_IND	s-activity-end indication
OSAK_C_ABORT_PPDU_SACTE_CNF	s-activity-end confirm

10.2. Data type: osak_abort_reason

Constant	Meaning
OSAK_C_PP_ABORT_NOTSPECIFIED [1]	No reason is specified.
OSAK_C_PP_ABORT_UNREC_PPDU	The presentation PDU is unrecognized.
OSAK_C_PP_ABORT_UNEXP_PPDU	An unexpected presentation PDU has occurred.
OSAK_C_PP_ABORT_UNEXP_SS	An unexpected session service primitive has occurred.
OSAK_C_PP_ABORT_UNREC_PARAM	An unrecognized presentation PDU parameter was detected.
OSAK_C_PP_ABORT_UNEXP_PARAM	An unexpected presentation PDU parameter was detected.
OSAK_C_PP_ABORT_INVALID_VALUE[1]	An invalid presentation PDU parameter value was detected.
OSAK_C_SP_ABORT_BADPROT	A session protocol violation was detected.
OSAK_C_SP_ABORT_UNKNOWNERR	An unknown error has occurred.
OSAK_C_ABORT_ACSE_USER[1]	The ACSE user is aborting the association.
OSAK_C_ABORT_ACSE_PROVIDER	The ACSE provider is aborting the association.
OSAK_C_ABORT_DISCONNECT	The transport connection has been lost.

[1]These are the only values that you can use in a call to `osak_abort_req`. Other values in the table may appear in an ABORT indication, but you cannot use these other values in a call to `osak_abort_req`.

10.3. Data type: osak_action_result

Constant	Meaning
OSAK_C_ACCEPT	Request to release an association was accepted.
OSAK_C_REJECT	Request to release an association was rejected.

10.4. Data type: osak_activity_reason

Constant	Meaning
OSAK_C_ACTIVITY_NOTSPECIFIED	No reason is specified.
OSAK_C_ACTIVITY_CANTCONTINUE	The requester is temporarily unable to continue the activity.
OSAK_C_ACTIVITY_SEQUENCE	There is an error in the call sequence.
OSAK_C_ACTIVITY_USER	A local session service user error has occurred.

Constant	Meaning
OSAK_C_ACTIVITY_PROCEDURAL	A procedural error has occurred.
OSAK_C_ACTIVITY_DEMAND	The data token is required.

10.5. Data type: osak_exception_reason

The possible values vary according to the source of the exception. The exception may originate from either of the following sources:

- The user

- The presentation provider

10.5.1. Exception Originating from User

Constant	Meaning
OSAK_C_EXCEPTION_NOTSPECIFIED	No reason is specified.
OSAK_C_EXCEPTION_CANTCONTINUE	The OSAK interface is temporarily unable to continue.
OSAK_C_EXCEPTION_SEQUENCE	There is an error in the call sequence.
OSAK_C_EXCEPTION_USER	A local session service user error has occurred.
OSAK_C_EXCEPTION_PROCEDURAL	A procedural error has occurred.
OSAK_C_EXCEPTION_DEMAND	The data token is required.

10.5.2. Exception Originating from Presentation Provider

Constant	Meaning
OSAK_C_EXCEPTION_BADPROT	There has been a session protocol error.

10.6. Field: pm_state

This is a field of the *osak_state* data type.

Constant	Meaning
OSAK_C_ASSOCIATE_IND	The process has received an association indication, but has not responded to it.
OSAK_C_PARTIAL_ASSOC_IND	The process has received an association indication with incomplete user data or no user data.
OSAK_C_DATA_TRANSFER	The process has established an association and is transferring data.

10.7. Field: reason

This is a field of the *osak_pcontext_proposal_result* data type.

Constant	Meaning
OSAK_C_PREASON_NOTSPECIFIED	Reason is not specified.
OSAK_C_PREASON_UNSUPP_ABS	The responding application does not support the proposed abstract syntax.
OSAK_C_PREASON_UNSUPP_TRANS	The responding application does not support the proposed transfer syntax.
OSAK_C_PREASON_DCS_LIMIT	A local limit on the defined context set has been exceeded.

10.8. Data type: osak_pdefault_context_res

Constant	Meaning
OSAK_C_ACCEPT	The responder accepts the proposed default context.
OSAK_C_USER_REJECT	The responder rejects the proposed default context.
OSAK_C_PROVIDER_REJECT	The provider rejects the proposed default context.

10.9. Data type: osak_reject_reason

Possible values vary according to the source of the rejection of an association request. The rejection may originate from any of the following sources:

- The user

- The ACSE provider

- The presentation provider

- The session provider

10.9.1. Rejection Originating from User

Constant	Meaning
OSAK_C_REJ_NULL	Null
OSAK_C_REJ_NOREASON	No reason is given.
OSAK_C_REJ_UNSUPPORTED_ACNAME	The application context name is not supported.
OSAK_C_REJ_UNKCALLING_AP_TITLE	The calling application-process title is not recognized.
OSAK_C_REJ_UNKCALLING_AP_ID	The calling application-process invocation identifier is not recognized.
OSAK_C_REJ_UNKCALLING_AE_QUAL	The calling application-entity qualifier is not recognized.
OSAK_C_REJ_UNKCALLING_AEINV_ID	The calling application-entity invocation identifier is not recognized.
OSAK_C_REJ_UNKCALLED_AP_TITLE	The called application-process title is not recognized.

Constant	Meaning
OSAK_C_REJ_UNKCALLED_AP_ID	The called application-process invocation identifier is not recognized.
OSAK_C_REJ_UNKCALLED_AE_QUAL	The called application-entity invocation qualifier is not recognized.
OSAK_C_REJ_UNKCALLED_AEINV_ID	The called application-entity invocation identifier is not recognized.

You can give any of these values in a call to `osak_reject_rsp`.

10.9.2. Rejection Originating from ACSE Provider

Constant	Meaning
OSAK_C_REJ_P_NULL	Null
OSAK_C_REJ_P_NOREASON	No reason is specified.
OSAK_C_REJ_P_ACSE_VERSION	There is no ACSE version that. both peer entities support

10.9.3. Rejection Originating from Presentation Provider

Constant	Meaning
OSAK_C_REJ_PP_PEER_REFUSED [1]	No reason is specified.
OSAK_C_REJ_PP_CONGESTED[1]	The presentation protocol machine is temporarily congested.
OSAK_C_REJ_PP_ADDRESS_UNKNOWN	The presentation address is unknown.
OSAK_C_REJ_PP_UNSUPPORTED	The proposed presentation protocol version is not supported.
OSAK_C_REJ_PP_NO_DEFAULT[1]	The proposed default presentation context is not supported.
OSAK_C_REJ_PP_NOT_READABLE[1]	The user data is not readable.
OSAK_C_REJ_PP_NO_PSAP_AVAIL	There is no presentation service access point (SAP) available.

[1]These are the only values that you can use in a call to `osak_reject_rsp`. Other values in the table may appear in an A_ASSOCIATE-REJECT confirm, but you cannot use these other values in a call to `osak_reject_rsp`.

10.9.4. Rejection Originating from Session Provider

Constant	Meaning
OSAK_C_REJ_SP_NO_SUCH_SSAP	There is no such session service access point (SAP).
OSAK_C_REJ_SP_NO_USER	The session service user is not attached to the session SAP.
OSAK_C_REJ_SP_CONGESTED	The session protocol machine is temporarily congested.
OSAK_C_REJ_SP_UNSUPPORTED	The proposed session protocol version is not supported.

Constant	Meaning
OSAK_C_REJ_SP_REFUSED	The session protocol machine has rejected the association attempt.

10.10. Data type: osak_release_reason

Constant	Meaning
OSAK_C_RLRQ_NORMAL	Normal release
OSAK_C_RLRQ_URGENT	Urgent release
OSAK_C_RLRQ_USERDEF	The reason is user defined.

10.11. Data type: osak_release_resp_reason

Constant	Meaning
OSAK_C_RLRE_NORMAL	Normal
OSAK_C_RLRE_NOTFINISHED	The responder has not finished.
OSAK_C_RLRE_USERDEF	The response is user defined.

10.12. Field: request_mask and returned mask

This is a field of the *osak_handle* data type.

Constant	Meaning
OSAK_C_READEVENT	The `osak_select` routine writes this value when an inbound event has occurred.
OSAK_C_WRITEEVENT	The `osak_select` routine writes this value when an outbound event has been completed.

10.13. Field: result

This is a field of the *osak_pcontext_proposal_result* data type.

Constant	Meaning
OSAK_C_ACCEPT	The peer entity accepts the proposal.
OSAK_C_USER_REJECT	The user rejects the proposal.
OSAK_C_PROV_REJECT	The provider rejects the proposal.

10.14. Data type: osak_resync_type

Constant	Meaning
OSAK_C_RESYNC_ABANDON	The OSAK interface resynchronizes to a synchronization point whose serial number is

Constant	Meaning
	higher than the serial numbers of synchronization points in use on the existing association.
OSAK_C_RESYNC_RESTART	The OSAK interface resynchronizes to a synchronization point set since the last acknowledged major synchronization point.
OSAK_C_RESYNC_SET	The OSAK interface resynchronizes to any valid synchronization point serial number.

10.15. Fields: data, sync_minor, major_activity and release

These are fields of the *osak_token_setting data type* data type.

Constant	Meaning
OSAK_C_TOKEN_INIT	The token is assigned to the initiator.
OSAK_C_TOKEN_RESP	The token is assigned to the responder.
OSAK_C_TOKEN_CHOOSE	The token is assigned according to the responder's choice.

10.16. Field: type

This is a field of the *osak_nsap* data type.

Constant	Meaning
OSAK_C_CONS	Connection-Oriented Network Service
OSAK_C_CLNS	Connectionless Network Service
OSAK_C_RFC1006	RFC 1006 network

Appendix A. OSAKserver (OpenVMS Systems Only)

The main sections in this appendix are:

- Active and passive addresses

- What OSAKserver does

- OSAK databases

- NCL and the OSAK databases

- Starting OSAKserver

- Registering active and passive addresses

## A.1. Active and Passive Addresses

You can implement your OSI application using either of two types of application address, active or passive.

An active address is associated with a process that is already started on the system. A passive address is associated with a process that is started only when a connection request is received for that address.

All connections to passive addresses are handled by OSAKserver, an inbound connection handler. This reduces the possibility of losing a connection because the transport timer expires before that connection is completed, but it increases the time needed to establish an upper-layer connection. Figure A.1 illustrates what OSAKserver does.

Figure A.1. OSAKserver
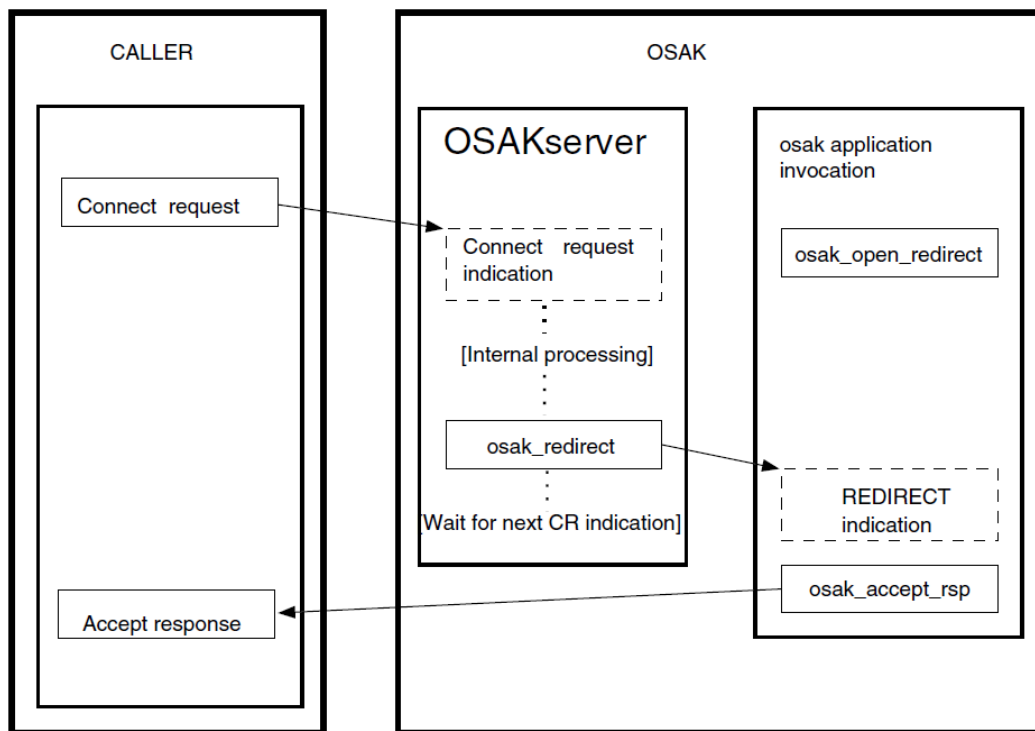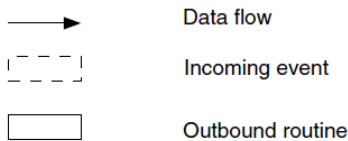
A.2. What OSAKserver Does

When OSAKserver receives an inbound connection, it does the following:

• Completes the transport connection.

• Reads the first transport service data unit (TSDU) from the connection and decodes it to find the presentation address of the application that is to receive the connection.

• Looks up the presentation address in an internal database maintained by OSAK. If there is no entry to match the address specified in the inbound connection request, OSAKserver rejects the connection attempt. The `connects rejected` counter attribute of the `osak` entity is incremented.

• Starts up a process to handle the connection.

• Hands over the connection request to an upper layer process, without making any change in the incoming data units.

A.3. OSAK Databases

OSAK maintains two databases: the application database and the port database. Users of previous versions of OSAK will be accustomed to referring to the address database; the current version of OSAK maintains similar information among the information it keeps in the application database.

A.4. NCL and the OSAK Databases

You must use NCL to inspect information held in the OSAK databases and to set attributes of entities in the OSAK module. Table A.1 shows the mapping between NCL and OSAK management.

Table A.1. Mapping Between NCL and OSAK

OSAK Database	NCL Entity
Application database	`osak application` and `osak application invocation`
Port database	`osak port`

A.5. Starting OSAKserver

You can start OSAKserver either manually or automatically.

To start OSAKserver manually, run the command procedure OSAK$START.COM, installed when you installed the DECnet-Plus software.

If you want OSAKserver start up automatically when you start up the network, edit NET$STARTUP.COM to include the line $ @OSAK$START.COM.

A.6. Registering Active and Passive Addresses

NCL creates the necessary management entities when OSAK sends or receives an appropriate programming call. You use NCL to register passive addresses.

This section describes how NCL registers active addresses(see Section A.6.1), and how you use NCL to register a passive address (see Section A.6.2).

A.6.1. Active

An application registers an active address by passing the address on a call to `osak_open_responder` or `osak_open_initiator`. NCL creates the appropriate entities. You cannot actively manage active addresses, but you can use NCL `show` commands to show attributes of these entities.

A.6.2. Passive

You register an application address using Network Control Language (NCL) commands to create an `osak application` entity and an `osak application invocation` entity. Use the `startup information` characteristic attribute of the `osak application invocation` entity to specify the following values:

Item	Value	Description
Mandatory		
user	`name`	The user name of the process that will respond to connect requests received by this application

Item	Value	Description
file	`pathname`	The name of the file to run to start up the named application
Optional		
account	`name`	The account that is to start the process
max resp	`integer`	The highest permissible number of responders, for an application with the `NEW` setting for `startup policy`
password	`password`	The user's password
sversion	{1}, {2}, or {1,2}	The session version

Further Information on OSAK Addresses

The chapter on the `osak` module in *VSI DECnet-Plus for OpenVMS Network Control Language Reference Guide* gives reference details.