VMS Software

VSI OpenVMS

VSI DECnet-Plus OSAK SPI Programming Reference

Document Number: DO-OSKSPI-01A

Publication Date: October 2021

Revision Update Information: This is a new manual.

Operating System and Version: VSI OpenVMS Integrity Version 8.4-2
VSI OpenVMS Alpha Version 8.4-2L1

VMS Software, Inc. (VSI)
Burlington, Massachusetts, USA

VSI DECnet-Plus OSAK SPI Programming Reference

VMS Software

Legal Notice

Preface

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

1. 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.

2. Prerequisites

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

- Have installed DECnet-Plus and the OSAK software on your system

  The DECnet-Plus installation and configuration documentation tells you how to install DECnet-Plus and the OSAK software.

- 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*) assumes that you understand the terminology and concepts used in the relevant standards.

3. 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*.

4. 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 OpenVMS support contracts through VSI can contact <support@vmssoftware.com> for help with this product.

5. OpenVMS Documentation

The full VSI OpenVMS documentation set can be found on the VMS Software Documentation webpage at .

6. 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*), 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.

Convention	Meaning
	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.

7. Abbreviations

The following abbreviations are used in this book:

APDU	application protocol data unit
BER	basic encoding rules
CLNS	Connectionless-Mode Network Service
CONS	Connection-Oriented Network Service
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
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 SPI Routines

This chapter contains the following information about the OSAK session programming interface (SPI):

- 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 SPI 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 SPI are as follows:

- osak_api.h
- osak_api_codes.h
- osak_api_messages.h

Their locations depend on the operating system as follows:

OpenVMS	SYS$COMMON:[SYSLIB]
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 SPI Parameters

Parameter	Brief Description	Data Type
abort_reason[1]	Reason for abort	osak_abort_reason
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)
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[2]	OSAK SPI version to be used	Unsigned long integer

Parameter	Brief Description	Data Type
called_aei[3]	Responder application-entity invocation	Address (osak_aei)
calling_aei[3]	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[3]	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
peer_data	User data (inbound)	Address (osak_buffer)
port_id	Port identifier	osak_port
process_id	Process identifier	Address (osak_process_id)
process_name	Process name	Address (osak_mem_descriptor)
protocol_versions[4]	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[5]	Reason for rejecting connection request	osak_reject_reason

Parameter	Brief Description	Data Type
request_tokens	Tokens requested from peer entity	Address (osak_token_setting)
responding_aei[3]	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)
user_context	Space for applications to store local information	Address
user_data	User data (outbound)	Address (osak_buffer)
workspace[6]	Parameter block workspace	None
ws_length	Length of workspace	Unsigned long integer

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

[2]Set this parameter to OSAK_C_API_VERSION_3.

[3]Use only the *p-address* structure within this parameter. Within the *p-address*, use only the *ssel*, *tsel*, and *nsap* fields. Set the *psel* field to null.

[4]Use only the *sversion* field in this parameter. Set the other fields to null.

[5]Use session-specific values only (see Section 6.6.1 and Section 6.6.2).

[6]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 SPI. 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_reason

Unsigned long integer

osak_action_result

Unsigned long integer

osak_activity_reason

Unsigned long integer

osak_aei

Field	Brief Description	Data Type
p-address	Presentation address	osak_paddress
ae-title[1]	Application-entity title	osak_aetitle
aeiid[1]	Application-entity invocation identifier	osak_aeiid

[1]Leave blank. This field is present in the structure but is not used by the SPI.

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 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

Field	Brief Description	Data Type
exceptions	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

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	Address (osak_nsap)
id	Address	osak_mem_descriptor
type	A constant defining the network protocol	Long integer

osak_paddress

Field	Brief Description	Data Type
psel[1]	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”)

[1]Leave blank. This field is present in the structure but is not used by the SPI.

osak_parameter_block

See Section 1.2.

osak_port

Address (Unsigned octet)

osak_protocol_versions

Field	Brief Description	Data Type
acse_version[1]	ACSE versions proposed	osak_acse_version
pversion[1]	Presentation versions proposed	osak_pversion
sversion	Session versions proposed	osak_sversion

[1]Leave blank. This field is present in the structure of the parameter block but is not used by the SPI.

osak_process_id

Unsigned long integer

osak_reject_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 connection	Unsigned octet
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_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
release	Release token selector	Bit field mask length 2

osak_transport_templates

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

Field	Brief Description	Data Type
name	Transport template name	osak_mem_descriptor

1.4. Routine Descriptions

This section contains a description of each OSAK SPI routine. Each routine has a set of arguments and parameters. Some of the arguments and parameters are common to many routines. The common arguments and parameters are described in Sections 1.4.1 and 1.4.2 respectively. Arguments and parameters that are not common to many routines are described in the individual routine descriptions.

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

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 SPI 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, in alphabetic order, 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 SPI 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 SPI 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 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;
}
```

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 SPI 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 SPI, the SPI 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 `spi_data_req` or `spi_typed_req`, the OSAK SPI 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 SPI 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 SPI always deallocates the same amount of memory as it allocated using your allocation routine. Your deallocation routine can ignore the *size* parameter.

The *ptr* parameter is a pointer to the memory to be deallocated. The possible return values of the routine are:

- Zero, indicating success
- Any other number, indicating failure

If the call to the deallocation routine fails, the OSAK SPI 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 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 SPI returns a code identifying the service you are calling. The codes are described in the include file, OSAK_API_CODES.H, which is included with the SPI.

more_flag

Use this parameter if you are sending segmented user data to indicate whether there is more user data to follow. You send this further user data on a call or calls to `spi_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 SPI 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 SPI. 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 SPI returns a status code in this parameter. If the result is OSAK_S_TRANSERR, the OSAK SPI also returns a transport provider status code. Chapter 3 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 a connection.

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 SPI as OSAK_C_MINIMUM_WS.

spi_abort_req

spi_abort_req — Aborts a connection.

Syntax

```
status = spi_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
[user_data]	osak_buffer	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

```
spi_abort_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

user_data

For session version 1, the *user_data* field can contain a maximum of 9 octets of data.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

If you want to send user data on the call, you can segment the user data between the call to `spi_abort_req` and calls to `spi_send_more`.

Return Value

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

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

spi_close_port
spi_release_req

spi_accept_rsp

spi_accept_rsp — Accepts a connection attempt.

Syntax

```
status = spi_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
[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
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_accept_rsp (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

responding_aei

The address of the responding application-entity. See the section called “osak_aei” in 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.

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 SPI does not send the associated session reference parameters.

segmentation

The address of a structure you can use to specify the 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 connection.

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 SPI uses the value that was received in the S-CONNECT 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 connection.

If the S-CONNECT 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 S-CONNECT indication. If you specify different values, the OSAK SPI aborts the connection.

If you make this parameter null when the S-CONNECT indication specifies that the responder should choose the token setting, the OSAK SPI gives all the available tokens to the initiator. The available tokens are those tokens of which the corresponding functional units are selected in the S-CONNECT indication. Section 6.9 lists the possible values of this parameter.

request_tokens

The address of a structure you can use to specify the tokens that the responder requires from the initiator. The OSAK SPI 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. If you omit this parameter, the OSAK SPI uses the functional units selected in the event S-CONNECT indication.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

Call this routine after receiving an S-CONNECT indication to accept the connection request.

Return Value

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

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

spi_connect_req
spi_open_responder
spi_reject_rsp

spi_act_discard_req

spi_act_discard_req — Terminates an activity and cancels its effects.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_act_discard_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters 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 6.3 lists the possible values of this parameter.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

You can use this service only if the activity management functional unit is selected. You must also have the major activity token.

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

spi_act_end_req
spi_act_discard_rsp

spi_act_discard_rsp

spi_act_discard_rsp — Responds to a request to discard an activity.

Syntax

```
status = spi_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

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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_act_discard_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.

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

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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

You can use this service only if the activity management functional unit is selected. You must also have the major activity token.

Call the routine after receiving a S-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 SPI has queued the request and there are free parameter blocks.

OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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

spi_act_discard_req

spi_act_end_req

spi_act_end_req — Terminates an activity and saves its effects.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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 SPI 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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Return Value

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

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

spi_act_discard_req
spi_act_end_rsp

spi_act_end_rsp

spi_act_end_rsp — Responds to a request to end an activity.

Syntax

```
status = spi_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
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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_act_end_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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

Call this routine after receiving a S-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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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

spi_act_end_req

spi_act_interrupt_req

spi_act_interrupt_req — Interrupts a lower-priority activity on a connection.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_act_interrupt_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters 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 6.3 lists the possible values of this parameter.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

You can use this service only if the activity management functional unit is selected. You must also have the major activity token.

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

spi_act_resume_req
spi_act_interrupt_rsp

spi_act_interrupt_rsp

spi_act_interrupt_rsp — Responds to a request to interrupt an activity.

Syntax

```
status = spi_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
status_block	osak_status_block	write only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_act_interrupt_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 requester does not want this token from the accepter.
- One means that the requester wants this token from the accepter.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Call this routine after receiving a S-ACTIVITY-INTERRUPT 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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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

spi_act_interrupt_req

spi_act_resume_req

spi_act_resume_req — Requests the resumption of an interrupted activity.

Syntax

```
status = spi_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

Parameters Used	Data Type	Access
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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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 a new identifier for 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. This is the *activity_id* specified in the `spi_act_start_req` routine.

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. 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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Return Value

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

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

spi_act_interrupt_req

spi_act_start_req

spi_act_start_req — Starts an activity within a connection.

Syntax

```
status = spi_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

Parameters Used	Data Type	Access
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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Return Value

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

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

spi_async_close

spi_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 = spi_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

```
spi_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 SPI 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 connection is in.

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

Parameters Used

tsdu_ptr

The address of the head of a linked list of user buffers. The OSAK SPI returns the unused buffers that the application passed in calls to `osak_give_buffers`. The OSAK SPI 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 SPI returns the parameter blocks that have been passed in outbound calls during the connection and have not already been collected.

Description

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

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

- Call `spi_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 `spi_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 `spi_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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI has queued the request.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.

See Also

spi_abort_req
spi_close_port

spi_open_redirect
spi_redirect
spi_release_req
spi_release_rsp

spi_capability_req

spi_capability_req — Transfers capability data over a connection.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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 are selected.
- There is no activity in progress on the connection.
- At least one octet of data is sent if *more_flag* is set to false.

You must have the major activity token. You must also have the data token and synchronize minor token if they are available.

Return Value

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

OSAK_S_FREE	The OSAK SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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_NODATA	No data has been specified in the call.
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

spi_capability_rsp

spi_capability_rsp

spi_capability_rsp — Responds to a request to transfer capability data.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Return Value

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

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

spi_capability_req

spi_close_port

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

Syntax

```
status = spi_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
parameter_block	osak_parameter_block	write only
destructive_flag	Unsigned octet	read only

C Binding

```
spi_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 SPI returns the unused buffers that the application passed in calls to `osak_give_buffers`. The OSAK SPI 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 SPI returns the parameter blocks that have been passed in outbound calls during the connection and have not already been collected.

destructive_flag

A flag that you can use to indicate how you want the OSAK SPI 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 connection is in.

If you set the value to OSAK_C_NON_DESTRUCTIVE, OSAK closes the port only when the connection has been terminated. If the connection is still active, the OSAK SPI returns OSAK_S_INVFUNC.

Description

Call this routine after you terminate a connection by aborting or releasing it or after you redirect a connection.

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

- Call `spi_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 `spi_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 `spi_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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI has queued the request.
OSAK_S_INVFUNC	The call is invalid.
OSAK_S_INVPORT	The port identifier is invalid.

See Also

spi_abort_req
spi_open_redirect
spi_redirect
spi_release_req
spi_release_rsp

spi_collect_pb

spi_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

```
spi_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 SPI returns any free parameter blocks that it is holding.

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

Description

The routine checks for the completion of outbound services on the specified port. The OSAK SPI 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 in the returned parameter blocks to determine the service on which the parameter block was used, for example, OSAK_C_OPENINITIATOR indicates that the call was to `spi_open_initiator`. These service codes are defined in 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.

spi_connect_req

spi_connect_req — Establishes a connection.

Syntax

```
status = spi_connect_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
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

Parameters Used	Data Type	Access
[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
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read 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

```
spi_connect_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

calling_aei

The address of your application-entity. See *the section called "osak_aei"* in *Section 1.3* for a description of the data type. The NSAP field is ignored in this routine.

called_aei

The address of a structure you should use to specify the peer entity with which you want to set up a connection. See *the section called "osak_aei"* in *Section 1.3* for a description of the data type. If you specify an invalid selector in any field, the OSAK SPI 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 internet addresses and OSI addresses.

transport_template

The address of a structure you can use to specify a list of transport templates that gives information about an application's transport requirements. If you do not specify a template, the OSAK SPI 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 *VSI DECnet-Plus for OpenVMS Network Management Guide* for further information on the OSI transport module in DECnet-Plus.

protocol_versions

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

- *acse_version*

- *pversion*
- *sversion*

However, only the *sversion* field is relevant for the SPI. If the *sversion* field contains the value zero, the OSAK SPI uses the default version number of 1 and 2 for the protocol. 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 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. 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 connection.

This parameter is mandatory if both of 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 connection, or to specify that the responder should choose the settings.

If this parameter is null, the OSAK SPI uses a default setting; all the available tokens are assigned to the initiator. Section 6.9 lists the possible values of this parameter.

functional_units

The address of a structure you can use to specify the 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 SPI uses the following default set of session functional units:

- 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 SPI assumes that you are not selecting any functional units, and returns status OSAK_S_INVFUS. You cannot set up a connection without any functional units. You should specify at least one duplex and one half-duplex functional unit.

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

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Return Value

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

OSAK_S_FREE	The OSAK SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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_INVFUNC	The call is invalid.
OSAK_S_INVFUS	The functional units are invalid.
OSAK_S_INVPORT	The port identifier is invalid.
OSAK_S_INVPV	Invalid protocol version.
OSAK_S_INVSYNCPNT	The synchronization point serial number is invalid.
OSAK_S_INVTEMPLATE	Transport template unknown.

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

spi_accept_rsp
spi_open_initiator
spi_open_responder
spi_reject_rsp

spi_control_give_req

spi_control_give_req — Relinquishes ownership of all available tokens.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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 connection 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.

Return Value

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

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

spi_token_please_req
spi_token_give_req

spi_data_req

spi_data_req — Transfers user information over a connection.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_data_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.

more_flag

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

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

You can use this service to send user information under normal circumstances. If the half-duplex functional unit is selected, you must have the data token.

Return Value

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

OSAK_S_FREE	The OSAK SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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_NODATA	No data has been specified in the call.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

spi_send_more

spi_exception_req

spi_exception_req — Signals error conditions that are not serious enough to cause termination of a connection.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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 6.4 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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

You can use this service only if the exception and half-duplex functional units have been selected. You must also 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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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.

spi_expedited_req

spi_expedited_req — Transfers expedited data over a connection.

Syntax

```
status = spi_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

```
spi_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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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.

spi_get_event

spi_get_event — Receives an event on a specified connection.

Syntax

```
status = spi_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

Parameters Used	Data Type	Access
event_type	osak_event	write only
[peer_data]	osak_buffer	write 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

```
spi_get_event (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Arguments

port

Identifies the connection on which an application is waiting to receive an event.

parameter_block

The address of a parameter block, in which the OSAK SPI returns the service parameters from the incoming event.

Parameters Used

tsdu_ptr

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

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

event_type

In this parameter, the OSAK SPI returns the type of the event. Table 1.2 lists the values that can occur in the *event_type* parameter, and their corresponding event types.

If you segment user data across multiple calls to `spi_get_event`, the OSAK SPI sets the *event_type* to OSAK_C_CONTINUE for every segment except the first one. When a disruptive event stops the reception of a partially received event, the value in the status block of the parameter block changes to OSAK_S_DISRUPTED.

When the return value of a call to `spi_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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

When an event arrives, the OSAK SPI extracts the user data and the PCI from the data units and writes them into the parameter block supplied in the call to `spi_get_event`. With the exception of the *peer_data* parameter, the parameters in the parameter block 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 `spi_get_event`, and how to use the buffers.

Note

The routine `spi_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, the incoming user data is segmented. Make additional calls to `spi_get_event` to collect all the user data, until the *more_flag* parameter is set to false.

Table 1.2. OSAK Event Types

Value in event_type Parameter	Event Type Indicated
OSAK_C_ABORT_IND	ABORT indication
OSAK_C_ACCEPT_CNF	S-CONNECT accept confirm
OSAK_C_ACT_DISCARD_CNF	S-ACTIVITY-DISCARD confirm
OSAK_C_ACT_DISCARD_IND	S-ACTIVITY-DISCARD indication
OSAK_C_ACT_END_CNF	S-ACTIVITY-END confirm
OSAK_C_ACT_END_IND	S-ACTIVITY-END indication
OSAK_C_ACT_INTERRUPT_CNF	S-ACTIVITY-INTERRUPT confirm
OSAK_C_ACT_INTERRUPT_IND	S-ACTIVITY-INTERRUPT indication
OSAK_C_ACT_RESUME_IND	S-ACTIVITY-RESUME indication
OSAK_C_ACT_START_IND	S-ACTIVITY-START indication
OSAK_C_ASSOC_IND	S-CONNECT indication
OSAK_C_CAPABILITY_CNF	S-CAPABILITY-DATA confirm

Value in event_type Parameter	Event Type Indicated
OSAK_C_CAPABILITY_IND	S-CAPABILITY-DATA indication
OSAK_C_CONTINUE	Continuation event when segmentation is in use
OSAK_C_CONTROL_GIVE_IND	S-CONTROL-GIVE indication
OSAK_C_DATA_IND	S-DATA indication
OSAK_C_EXCEPTION_IND	S-U-EXCEPTION indication
OSAK_C_EXPEDITED_DATA_IND	S-EXPEDITED-DATA indication
OSAK_C_P_EXCEPTION_IND	S-P-EXCEPTION indication
OSAK_C_NO_EVENT	Returned when the status code of the call to `spi_get_event` is OSAK_S_NOEVENT
OSAK_C_PLEASE_IND	S-TOKEN-PLEASE indication
OSAK_C_REDIRECT_IND	Redirect indication
OSAK_C_REJECT_CNF	S-CONNECT reject confirm
OSAK_C_RELEASE_CNF	S-RELEASE confirm
OSAK_C_RELEASE_IND	S-RELEASE indication
OSAK_C_RESYNC_CNF	S-RESYNCHRONIZE confirm
OSAK_C_RESYNC_IND	S-RESYNCHRONIZE indication
OSAK_C_SYNC_MAJOR_CNF	S-SYNC-MAJOR confirm
OSAK_C_SYNC_MAJOR_IND	S-SYNC-MAJOR indication
OSAK_C_SYNC_MINOR_CNF	S-SYNC-MINOR confirm
OSAK_C_SYNC_MINOR_IND	S-SYNC-MINOR indication
OSAK_C_TDIS	Transport disconnect indication
OSAK_C_TOKEN_GIVE_IND	S-TOKEN-GIVE indication
OSAK_C_TYPED_DATA_IND	S-TYPED-DATA indication

Return Value

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

OSAK_S_FREE	The OSAK SPI 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 SPI 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

spi_give_buffers
spi_select

spi_get_handle

spi_get_handle — Returns the transport connection handle for a connection.

Syntax

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

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

C Binding

```
spi_get_handle (port, handle)
osak_port port;
longword *handle;
```

Arguments

handle

In this argument, the OSAK SPI returns the address of the external handle for the connection specified in the *port* parameter.

On OpenVMS systems, the handle is an event flag number (EFN). On UNIX systems, the 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 OSAK uses to set up a transport connection. You can use this file descriptor in a call to `spi_select` as an alternative to using the *port* identifier returned by `spi_open_initiator`, `spi_open_responder`, or `spi_open_redirect`.

If your application is acting as responder, note that the file descriptor returned by the OSAK SPI before a transport connection is not the same as that returned after the 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 allows the OSAK SPI 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

spi_select

spi_give_buffers

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

Syntax

```
status = spi_give_buffers (port, rcv_buffer_list)
```

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

C Binding

```
spi_give_buffers (port, rcv_buffer_list,)

osak_port port;
struct osak_buffer *rcv_buffer_list;
```

Arguments

rcv_buffer_list

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

Description

This routine supplies buffers that the OSAK SPI passes to the transport interface. The transport interface fills the buffers with segments of incoming TSDUs, and passes them back to the OSAK SPI.

See *VSI DECnet-Plus OSAK Programming* for more 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

spi_get_event
spi_close_port

spi_major_req

spi_major_req — Requests the setting of a major synchronization point.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_major_req (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

sync_point

This parameter is only returned by the OSAK interface.

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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

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 synchronize minor 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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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

spi_act_end_req
spi_major_rsp
spi_minor_req
spi_resync_req

spi_major_rsp

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

Syntax

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

Argument	Data Type	Access
port	osak_port	read 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
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Return Value

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

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

spi_major_req

spi_minor_req

spi_minor_req — Requests the setting of a minor synchronization point.

Syntax

```
status = spi_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
[data_separation]	osak_data_separation	read only
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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. The parameter contains the *sync_point* before it is incremented by OSAK, that is, it contains the serial number sent in the protocol.

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

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 SPI 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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

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

spi_major_req
spi_minor_rsp
spi_resync_req

spi_minor_rsp

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

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

Return Value

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

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

spi_minor_req

spi_open_initiator

spi_open_initiator — Allocates a port for use in a subsequent request to establish a connection.

Syntax

```
status = spi_open_initiator (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
[local_aei]	osak_aei	read only
alloc_rtn	osak_rtn	read only
dealloc_rtn	osak_rtn	read only
[alloc_param]	unsigned integer	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_open_initiator (port, parameter_block)

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

Arguments

port

In this parameter, the OSAK SPI 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 the section called “osak_aei” in Section 1.3 for a description of the data type.

Any of the fields of this structure can be null. The NSAP field is ignored in this routine.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

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

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

Return Value

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

OSAK_S_FREE	The OSAK SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_DISRUPTED	A disruptive event has occurred.
OSAK_S_QUEUED	The OSAK SPI has queued the request (returned only if the call includes a completion routine).
OSAK_S_INSFWS	There is not enough workspace in the parameter block.
OSAK_S_INVTEMPLATE	Transport template is not in the parameter list.

See Also

spi_accept_rsp
spi_connect_req
spi_open_responder
spi_reject_rsp

spi_open_redirect

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

Syntax

```
status = spi_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
[local_aei][1]	osak_aei	read only
alloc_rtn	osak_rtn	read only
dealloc_rtn	osak_rtn	read only
[alloc_param]	Unsigned integer	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

[1]OpenVMS systems only: for a passive application, you should set this parameter to the address on which the application receives inbound connections. For an active application, set the parameter to null.

C Binding

```
spi_open_redirect (port, parameter_block)
```

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

Arguments

port

In this parameter, the OSAK SPI returns an identifier for the access point between itself and the new responder process.

Description

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

OpenVMS

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

Return Value

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

OSAK_S_FREE	The OSAK SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI has queued the request (returned only if the call includes a completion routine).
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.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

spi_close_port
spi_redirect

spi_open_responder

spi_open_responder — Opens a port for use by a responder.

Syntax

```
status = spi_open_responder (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
[transport_template]	osak_transport_templates	read only
local_aei (UNIX systems only)	osak_aei	read only
[local_aei] (OpenVMS systems only)	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
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_open_responder (port, parameter_block)

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

Arguments

port

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

Parameters Used

local_aei (UNIX systems only)

The address of a structure specifying the responder's application-entity invocation. See the section called “osak_aei” in 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.

The NSAP is ignored in this routine.

protocol_versions

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

- *acse_version*
- *pversion*
- *sversion*

However, only the *sversion* field is relevant for the SPI. If the *sversion* field contains the value zero, the OSAK SPI uses the default version numbers of 1 and 2 for the protocol. The two session default values are not mutually exclusive.

OpenVMS

On OpenVMS systems, if OSAK receives an S-CONNECT indication specifying incompatible protocol versions, OSAK rejects the connection attempt.

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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

This routine allocates a port identifier that you should specify to the OSAK SPI in all service requests made on that port. The port is ready to receive connection 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.

Return Value

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

OSAK_S_FREE	The OSAK SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI has queued the request (returned only if the call includes a completion routine).

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.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

spi_accept_rsp
spi_connect_req
spi_open_initiator
spi_reject_rsp

spi_redirect

spi_redirect — Requests the redirection of a connection from one process to another on the same system.

Syntax

```
status = spi_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
redirect_state	osak_state	read only
[calling_aei]	osak_aei	read only
[called_aei]	osak_aei	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
[alloc_param]	Unsigned integer	read only
[rcv_data_list]	osak_buffer	read only

Parameters Used	Data Type	Access
[local_data]	osak_mem_descriptor	read only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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.

redirect_state

A structure you should use to specify the state of the connection 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 connection, and to false if the local entity is the responder to the connection.

- *pm_state*

  Set this field to the state of the connection when the redirect call is made. Section 6.5 lists the possible values of this parameter.

calling_aei

The address of a structure you can use to specify the presentation address of your application-entity. See the section called “osak_aei” in 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 of the peer entity with which you want to make a connection. See the section called “osak_aei” in Section 1.3 for a description of the data type.

protocol_versions

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

- *acse_version*
- *pversion*
- *sversion*

However, only the *sversion* field is relevant for the SPI. If the *sversion* field contains the value zero, the OSAK SPI uses the default version numbers of 1 and 2 for the protocol. 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 connection.

activity_id

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

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 connection.

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 S-CONNECT indication, but has not responded to it.
- The process has received a connection 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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

A connection should be in one of the following states before you can redirect it:

- The process has received an S-CONNECT indication, but has not responded to it.
- The process has received an S-CONNECT indication with incomplete user data or no user data.
- The process has established a connection and is transferring data.

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

Before you call `spi_redirect`, the destination process should call `spi_open_redirect`, which returns a new port identifier.

When you call `spi_redirect`, the OSAK SPI 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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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_INVFUNC	The call is invalid.
OSAK_S_INVFUS	The functional units are 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_NOPROCINFO	There is no process identifier and no process name.
OSAK_S_NOSYNCPNT	The synchronization point serial number is missing.
OSAK_S_READPOSTED	The transport service is temporarily unable to accept the redirect. Try again. (OpenVMS only).
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

spi_close_port
spi_open_redirect

spi_reject_rsp

spi_reject_rsp — Rejects a connection request.

Syntax

```
status = spi_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
[responding_aei]	osak_aei	read only
[sconnect_id]	osak_sconnect_id	read only
[functional_units]	osak_fus	read only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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 a connection request. Section 6.6 lists the possible values of this parameter.

responding_aei

The address of a structure you can use to specify the presentation selector of the responding application-entity. See the section called “osak_aei” 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* are ignored by the SPI and 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 the section called “osak_sconnect_id” in Section 1.3 for a description of the data type.

user_data

User data is only allowed if the reason given for the rejection is user specified (see Section 6.6.1). Otherwise, OSAK_S_OVERFLOW is returned.

functional_units

The address of a structure you can use to specify which functional units are selected for use. If you do not set the parameter, the OSAK SPI uses the selected functional units from the S-CONNECT indication.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

Call this routine after receiving an S-CONNECT indication to reject the connection attempt.

Return Value

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

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

OSAK_S_QUEUED	The OSAK SPI 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_INVPORT	The port identifier is invalid.
OSAK_S_OVERFLOW	Too much user data has been sent for session version 1 or user data specified for reject reason other than user specified.
OSAK_S_INVREASON	The reason code is invalid.
OSAK_S_TRANSERR	There is an error in the transport provider.

See Also

spi_accept_rsp
spi_connect_req
spi_open_initiator
spi_open_responder

spi_release_req

spi_release_req — Requests the orderly release of a connection.

Syntax

```
status = spi_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
func	Unsigned long integer	write only
status_block	osak_status_block	write only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_release_req (port, parameter_block)
```

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

Description

The caller of this service must own all the available tokens that are in use. The other user can only reject the release request if the negotiated release functional unit is selected and the other user holds the release token.

Return Value

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

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

spi_abort_req
spi_release_rsp
spi_close_port

spi_release_rsp

spi_release_rsp — Responds to a request for orderly release of a connection.

Syntax

```
status = spi_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

Parameters Used	Data Type	Access
status_block	osak_status_block	write only
[action_result]	osak_action_result	read only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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 connection. Omission of this parameter means that the accepter agrees to the release request. You can reject a release request only if the negotiated release functional unit is selected and the peer entity holds the release token. See Section 6.2 for a list of the possible values of this parameter.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

Call this routine after receiving an S-RELEASE indication.

Return Value

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

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

spi_release_req
spi_close_port

spi_resync_req

spi_resync_req — Requests the resynchronization of a connection to a specified synchronization point.

Syntax

```
status = spi_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
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
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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)

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

- *restart* (resync_type=PSAK_C_RESYNC_RESTART)

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

- *set* (resync_type=OSAK_C_RESYNC_SET)

  Resynchronize 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 SPI supplies the value. If you specify a value when the resynchronization type is *abandon*, the OSAK SPI 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 6.9 lists the possible values of this parameter.

If no tokens are available on the connection, but you specify the *tokens* parameter, the OSAK SPI 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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

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

spi_major_req
spi_minor_req
spi_resync_rsp

spi_resync_rsp

spi_resync_rsp — Responds to a request for resynchronization of a connection.

Syntax

```
status = spi_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
[token_item]	osak_token_setting	read only
[user_data]	osak_buffer	read only
more_flag	Long integer	read only
[data_length]	Unsigned longword	read 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

```
spi_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 S-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 S-RESYNCHRONIZE indication.

tokens

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

token_item

The address of a structure specifies the assignment of tokens after resynchronization. If the S-RESYNCHRONIZE indication indicates that the responder can choose how to assign the tokens, you can use this parameter to specify the assignment. Otherwise, this parameter should contain the values from the S-RESYNCHRONIZE indication.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

Call this routine after receiving a S-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 SPI has queued the request and there are free parameter blocks.
OSAK_S_NORMAL	The routine has finished without error.
OSAK_S_QUEUED	The OSAK SPI 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

spi_resync_req

spi_select

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

Syntax

```
status = spi_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

```
api_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 SPI to notify you. The array has three fields:

- *id* specifies one or more ports that the OSAK SPI should inspect
- *request_event_mask* specifies the type of event for which the OSAK SPI should inspect the ports
- *returned_event_mask* lists the events that occur on the specified ports before the OSAK SPI returns control to your application

A port is defined to be one of the following:

- An OSI connection. The `spi_open_initiator`, `spi_open_redirect` and `spi_open_responder` calls return the identifier of the connection in the *port* parameter.
- An identifier specific to the operating system. The identifier refers to the source on which the OSAK SPI makes selections. On OpenVMS systems, the source is a port or an event flag number (EFN). On UNIX 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 SPI 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 parameter 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 parameter to null.
- Make several calls to `spi_select` with timers lasting less than a day. When the timer expires, re-issue the `spi_select` call.

Description

The routine examines the ports listed in the *port_list* parameter 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* parameter or when the *time_out* parameter expires.

UNIX

Although the `spi_select` call maps directly on to the UNIX `select(2)` system call, the semantics of the write bit are different. The OSAK SPI uses the write bit to indicate that a write event has finished. The UNIX operating system uses the write bit to indicate that writing is possible.

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

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

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

OpenVMS

You can pass an event flag number (EFN) to `spi_select` as a port identifier. If you do this, you should use an EFN from cluster 1. The OSAK SPI 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 SPI 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* argument. When an inbound event arrives, the OSAK SPI returns OSAK_S_NORMAL.

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

Table 1.3 shows the meanings of values in the 2-bit masks.

Table 1.3. 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 SPI to wait for an event to arrive. If no event arrives within that time, the OSAK SPI returns the status OSAK_S_NOEVENT and clears the returned-event bit mask. If an event arrives, the OSAK SPI 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* field 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

spi_get_event
spi_get_handle
spi_give_buffers
spi_open_initiator
spi_open_redirect
spi_open_responder
spi_select

spi_send_more

spi_send_more — Sends a further segment of user data.

Syntax

```
status = spi_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
more_flag	Long integer	read only
data_length	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_send_more (port, parameter_block)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Description

The OSAK SPI allows you to segment the user data you are sending on any service. Call `spi_send_more` as many times as necessary to send user data to complete a service. The completion of each call to `spi_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 a service call that has the *more_flag* set to true. If you set *more_flag* to true on the original service call, you can send all the user data on calls to `spi_send_more`.

Return Value

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

OSAK_S_FREE	The OSAK SPI has queued the request and there are free parameter blocks.
OSAK_S_QUEUED	The OSAK SPI 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.
OSAK_S_TRANSERR	There is an error in the transport provider.

spi_token_give_req

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

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_token_give_req (port, parameter_block,)

osak_port port;
struct osak_parameter_block *parameter_block;
```

Parameters Used

token_item

The address of a structure specifies the tokens 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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

Description

You must own a token before you can give it away.

If you are using session version 1, there is no user data on this service and therefore, no segmentation is allowed.

Return Value

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

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

spi_control_give_req
spi_token_please_req

spi_token_please_req

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

Syntax

```
status = spi_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

Parameters Used	Data Type	Access
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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.

Other parameters

All other parameters used in this routine are described in Section 1.4.1 and Section 1.4.2.

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

spi_control_give_req
spi_token_give_req

spi_typed_req

spi_typed_req — Transfers typed data over a connection.

Syntax

```
status = spi_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
more_flag	Long integer	read only
[data_length]	Unsigned longword	read only
[completion_rtn]	osak_rtn	read only
[completion_param]	Longword	read only

C Binding

```
spi_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.

Return Value

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

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

OSAK_S_QUEUED	The OSAK SPI 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 `spi_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:

event_type

The type of the event that `spi_get_event` receives. Table 1.2 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.

This parameter is not used for a redirect indication.

status_block

When `spi_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 `spi_give_buffers`.

To free the buffers when they are returned by `spi_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
[peer_data]	osak_buffer

Parameters Returned	Data Type
[more_flag]	Long integer
local_abort	Long integer

Parameters Returned

abort_reason

This parameter explains why the connection is being aborted. Section 6.1 lists the possible values.

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 `spi_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 connection. You should delete all the data structures for the connection.

If this is a user abort, your application should make multiple calls to `spi_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, using the `spi_close_port` routine.

REDIRECT indication

REDIRECT indication — An application entity receives this event when the remote peer entity calls `spi_redirect`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
redirect_state	osak_state
[calling_aei]	osak_aei
[called_aei]	osak_aei
[protocol_versions]	osak_protocol_versions
[sconnect_id]	osak_sconnect_id
[segmentation]	osak_segmentation

Parameters Returned	Data Type
[sync_point]	osak_sync_point
[tokens]	osak_token_setting
[activity_id]	osak_mem_descriptor
[functional_units]	osak_fus
[rcv_data_list]	osak_buffer
[local_data]	osak_mem_descriptor
[more_flag]	Long integer

Parameters Returned

redirect_state

A structure defining the state of the connection. The values in the structure indicate whether the local entity is acting as initiator of or responder to the redirection request.

calling_aei

The address of a structure containing selector information about the calling application entity. 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 selector information about the called application entity. If the incoming data units do not include this information, the OSAK interface returns a null address.

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 a default value of either 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. 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 a connection 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 connection when the requester called `spi_redirect`
- Whether all the user data fits into the buffer supplied in the most recent call to `spi_give_buffers`

The description of the routine `spi_redirect` in Chapter 1 describes the possible states of a connection. Table 2.1 shows the sequence of calls you should make for each state. In this table, each state is represented by its constant value. Section 6.5 lists the meanings of these constants.

Table 2.1. Sequence of Calls After the Arrival of a REDIRECTIndication

State	Sequence of Calls
OSAK_C_ASSOC_IND	Call `spi_get_event` to receive the REDIRECT indication.

State	Sequence of Calls
	Call `spi_accept_rsp` to accept or `spi_reject_rsp` to reject the REDIRECT indication.
OSAK_C_PARTIAL_ASSOC_IND	Make multiple calls to `spi_get_event` to receive all the user data from the REDIRECT indication. Call `spi_accept_rsp` to accept or `spi_reject_rsp` to reject the REDIRECT indication.
OSAK_C_DATA_TRANSFER	Continue exchange of data between peer entities.

S-ACTIVITY-DISCARD confirm

S-ACTIVITY-DISCARD confirm — An application entity receives this event when the remote peer entity calls `spi_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
[more_flag]	Long integer

Description

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

S-ACTIVITY-DISCARD indication

S-ACTIVITY-DISCARD indication — An application entity receives this event when the remote peer entity calls `spi_act_discard_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
[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 6.3 lists the values of this parameter.

Description

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

S-ACTIVITY-END confirm

S-ACTIVITY-END confirm — An application entity receives this event when the remote peer entity calls `spi_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
[more_flag]	Long integer

Description

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

S-ACTIVITY-END indication

S-ACTIVITY-END indication — An application entity receives this event when the remote peer entity calls `spi_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

Parameters Returned	Data Type
[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 `spi_act_end_rsp`.

S-ACTIVITY-INTERRUPT confirm

S-ACTIVITY-INTERRUPT confirm — An application entity receives this event when the remote peer entity calls `spi_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
[more_flag]	Long integer

Description

This event responds to a request to interrupt the current activity.

S-ACTIVITY-INTERRUPT indication

S-ACTIVITY-INTERRUPT indication — An application entity receives this event when the remote peer entity calls `spi_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

Parameters Returned	Data Type
[peer_data]	osak_buffer
[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 6.3 lists the values of this parameter.

Description

This event indicates a request to interrupt the current activity. Respond by calling `spi_act_interrupt_rsp`.

S-ACTIVITY-RESUME indication

S-ACTIVITY-RESUME indication — An application entity receives this event when the remote peer entity calls `spi_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
[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.

S-ACTIVITY-START indication

S-ACTIVITY-START indication — An application entity receives this event when the remote peer entity calls `spi_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
[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.

S-CAPABILITY-DATA confirm

S-CAPABILITY-DATA confirm — An application entity receives this event when the remote peer entity calls `spi_capability_rsp`.

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
[more_flag]	Long integer

Description

This event confirms that capability data sent by the remote peer entity has been received.

S-CAPABILITY-DATA indication

S-CAPABILITY-DATA indication — An application entity receives this event when the remote peer entity calls `spi_capability_req`.

Syntax

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

Description

This event indicates that capability data is being sent. Respond by calling `spi_capability_rsp`.

S-CONNECT indication

S-CONNECT indication — An application entity receives this event when the remote peer entity calls `spi_connect_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
called_aei	osak_aei
[calling_aei]	osak_aei
protocol_versions	osak_protocol_versions
[sconnect_id]	osak_sconnect_id

Parameters Returned	Data Type
[segmentation]	osak_segmentation
[initial_serial_number]	osak_sync_point
[initial_tokens]	osak_token_setting
functional_units	osak_fus
[peer_data]	osak_buffer
[more_flag]	Long integer

Parameters Returned

called_aei

The address of selector information about the called application entity. If the incoming data units do not include this information, the OSAK interface returns a null address.

calling_aei

The address of selector information about the calling application-entity. 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 connection. If the incoming data units do not specify any proposed protocol version identifiers, the OSAK interface returns a default value of 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 connection. 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 connection. 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 functional units accepted for this connection. 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

Description

This event indicates a request to establish a connection. Accept the connection request by calling `spi_accept_rsp`; reject the connection by calling `spi_reject_rsp`.

S-CONNECT-ACCEPT confirm

S-CONNECT-ACCEPT confirm — An application entity receives this event when the remote peer entity calls `spi_accept_rsp`.

Syntax

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

Parameters Returned

responding_aei

The address of information about the application entity that is responding to a request to set up a connection. 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 version in use on the connection.

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 connection. 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 connection. 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 identifiers of the tokens 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 connection. 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

Description

This event indicates a positive response to a request to establish a connection. It means that the connection has been established andyou can start transferring data.

S-CONNECT-REJECT confirm

S-CONNECT-REJECT confirm — An application entity receives this event when the remote peer entity calls `spi_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
[responding_aei]	osak_aei
[sconnect_id]	osak_sconnect_id
[functional_units]	osak_fus
[peer_data]	osak_buffer
[more_flag]	Long integer

Parameters Returned

reject_reason

This parameter explains why the connection request is being rejected. The parameter also specifies whether this is a rejection by the service user or by the service provider.

Section 6.6 lists the values of this parameter.

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

responding_aei

The address of selector information about the application entity that is responding to a request to set up a connection.

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 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

Description

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

S-CONTROL-GIVE indication

S-CONTROL-GIVE indication — An application entity receives this event when the remote peer entity calls `spi_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
[more_flag]	Long integer

Description

This event indicates that the remote peer entity relinquishes ownership of all the tokens available on a connection. No response is necessary because relinquishing tokens is not a confirmed service.

S-DATA indication

S-DATA indication — An application entity receives this event when the remote peer entity calls `spi_data_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[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.

S-EXPEDITED-DATA indication

S-EXPEDITED-DATA indication — An application entity receives this event when the remote peer entity calls `spi_expedited_req`.

Syntax

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

Description

This event indicates the sending of expedited data over a connection. No response is necessary because sending expedited data is not a confirmed service.

S-P-EXCEPTION-REPORT indication

S-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
tsdu_ptr	osak_buffer
exception_reason	osak_exception_reason
[peer_data]	osak_buffer
[more_flag]	Long integer

Parameters Returned

exception_reason

A value indicating the reason for the exception report.

Section 6.4 lists the values of this parameter.

Description

This event indicates a user error has occurred that is not severe enough to cause a connection 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.

S-RELEASE confirm

S-RELEASE confirm — An application entity receives this event when the remote peer entity calls `spi_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
[peer_data]	osak_buffer
[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.

Description

This event responds to a request to terminate a connection. To reclaim parameter blocks and user buffers that were used on the connection, you can call `spi_close_port`, or use a completion routine (OpenVMS systems only).

S-RELEASE indication

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

Syntax

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

Description

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

- If the negotiated release functional unit is not in use on the connection, call `spi_release_rsp` to accept the release request.
- If the negotiated release functional unit is in use on the connection, call `spi_release_rsp` to accept or reject the release request.

S-RESYNCHRONIZE confirm

S-RESYNCHRONIZE confirm — An application entity receives this event when the remote peer entity calls `spi_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
[token_item]	osak_token_setting
[peer_data]	osak_buffer
[more_flag]	Long integer

Parameters Returned

sync_point

The address of the synchronization point serial number specified in the resynchronization request.

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.

Description

This event confirms resynchronization from a specified synchronization point.

S-RESYNCHRONIZE indication

S-RESYNCHRONIZE indication — An application entity receives this event when the remote peer entity calls `spi_resync_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block

Parameters Returned	Data Type
tsdu_ptr	osak_buffer
resync_type	osak_resync_type
sync_point	osak_sync_point
[tokens]	osak_token_setting
[peer_data]	osak_buffer
[more_flag]	Long integer

Parameters Returned

resync_type

A value indicating the type of resynchronization.

Section 6.8 lists the values of this parameter.

sync_point

The address of the synchronization point serial number from which resynchronization should start.

tokens

The address of a structure indicating the assignment of tokens after resynchronization. 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 6.9 lists the values of this parameter.

Description

This event indicates that a request to resynchronize a connection from a specified synchronization point. Respond by calling `spi_resync_rsp`.

When you receive an S-RESYNCHRONIZE indication, keep data segments that arrived before the synchronization point specified in the indication. Ignore data segments received after that synchronization point.

S-SYNC-MAJOR confirm

S-SYNC-MAJOR confirm — An application entity receives this event when the remote peer entity calls `spi_major_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer

Parameters Returned	Data Type
[more_flag]	Long integer
sync_point	osak_sync_point

Description

This event confirms the setting of a major synchronization point.

S-SYNC-MAJOR indication

S-SYNC-MAJOR indication — An application entity receives this event when the remote peer entity calls `spi_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
[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 `spi_major_rsp`.

S-SYNC-MINOR confirm

S-SYNC-MINOR confirm — An application entity receives this event when the remote peer entity calls `spi_minor_rsp`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer

Parameters Returned	Data Type
sync_point	osak_sync_point
[peer_data]	osak_buffer
[more_flag]	Long integer

Parameters Returned

sync_point

The address of the minor synchronization point serial number.

Description

This event confirms the setting of a minor synchronization point.

S-SYNC-MINOR indication

S-SYNC-MINOR indication — An application entity receives this event when the remote peer entity calls `spi_minor_req`.

Syntax

Parameters Returned	Data Type
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
sync_point	osak_sync_point
sync_confirm	osak_sync_confirm
[peer_data]	osak_buffer
[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.

Description

This event indicates a request to set a minor synchronization point. You can respond by calling `spi_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.

S-TOKEN-GIVE indication

S-TOKEN-GIVE indication — An application entity receives this event when the remote peer entity calls `spi_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
[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 a connection. The *token_item* parameter specifies which tokens are being relinquished. No response is necessary because relinquishing tokens is not a confirmed service.

S-TOKEN-PLEASE indication

S-TOKEN-PLEASE indication — An application entity receives this event when the remote peer entity calls `spi_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
[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 connection. The *token_item* parameter specifies which tokens are being requested. No response to the request is necessary because this is not a confirmed service.

See Section 6.9 for the possible values.

S-TYPED-DATA indication

S-TYPED-DATA indication — An application entity receives this event when the remote peer entity calls `spi_typed_req`.

Syntax

Parameters Returned	**Data Type**
event_type	osak_event
status_block	osak_status_block
tsdu_ptr	osak_buffer
[peer_data]	osak_buffer
[more_flag]	Long integer

Description

This event indicates that typed data is being sent over a connection. No response is necessary because sending typed data is not a confirmed service.

S-U-EXCEPTION-REPORT indication

S-U-EXCEPTION-REPORT indication — An application entity receives this event when the remote peer entity calls `spi_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
[more_flag]	Long integer

Parameters Returned

exception_reason

A value indicating the reason for the exception report.

Section 6.4 lists the values of this parameter.

Description

This event indicates that an error has occurred that is not severe enough to cause a connection 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.

Chapter 3. 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 3.1)
- Informational (Section 3.2)
- Error (Section 3.3)

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 `spi_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.

3.1. Success Status Codes

OSAK_S_NORMAL, the routine has finished without error

This status code 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 `spi_close_port` returns this status code when the OSAK interface has closed the port specified in the call.

- A call to `spi_get_event` returns this status code when an event has arrived on the connection specified in the call.
- A call to `spi_get_handle` returns this status code when the OSAK interface has identified the channel or channels specified in the call.
- A call to `spi_give_buffers` returns this status code when user data buffers have been passed to OSAK.
- A call to `spi_open_initiator`, `spi_open_responder`, or `spi_redirect` returns this status code when the OSAK interface has opened a port.
- A call to `spi_collect_pb` returns this status code when a parameter block used on an outbound service has been returned.
- A call to `spi_select` returns this status code when an event is present on the channel or channels specified in the call.

3.2. 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 `spi_collect_pb`.

The code means the same as OSAK_S_QUEUED, with the added indication that a call to `spi_collect_pb` will generate an OSAK_S_NORMAL return rather than an OSAK_S_NO_EVENT.

OSAK_S_NO_EVENT, no event has occurred

A call to `spi_get_event` returns this status code when there is no event waiting to be received on the specified connection. The *event_type* parameter contains the value OSAK_C_NO_EVENT.

OpenVMS

> This status code is also returned when you call *spi_get_event* with a completion routine and then release or abort the connection on which the call is made. The completion routine starts running when the connection closes.

A call to `spi_collect_pb` returns this status code when there are no completed outbound services on the specified connection, and hence no parameter blocks waiting to be collected.

A call to `spi_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 `spi_connect_request`, or when you are sending segmented user data using the routine `spi_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:

- `spi_get_event`
- `spi_open_initiator`
- `spi_open_redirect`
- `spi_open_responder`

If no completion routine is included on a call to one of these routines, the routine can return only OSAK_S_NORMAL or an error status.

3.3. 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. It indicates that there is a problem in your application.

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. This error indicates a problem with your application.

OSAK_S_DISRUPTED, a disruptive event has occurred

The request was canceled by a disruptive event. Chapter 4 explains what to do when a disruptive event occurs.

OSAK_S_FILERR, an error occurred in opening the trace file

Returned when a call to `spi_trace_open` fails. Check the validity of the parameters that you passed in the call.

OSAK_S_INCPCI, incomplete PCI

Returned by `spi_get_event` when a partial event arrives with the protocol control information (PCI) incomplete. You should make another call to `spi_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 connection 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 release some memory.

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 `spi_release_rsp`.

Check which functional units are in use on this connection. You should not pass the *action_result* parameter in a call to `spi_release_rsp` unless you are using the negotiated release functional unit on the connection.

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_INVTSEL, Invalid Transport Selector

  Indicates that the TSEL is not known at the remote node.

- 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. This is not allowed. If you want to specify a different SSEL, you should also specify a different TSEL.

- OSAK_S_NOSUCHENTRY, No such entry in OSAKserver address database

  A call to `spi_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 `spi_open_initiator`, `spi_open_responder` or `spi_redirect` returns this status code when OSAK has been restricted, disabled or deleted by OSAK network management (by means of NCL).

- OSAK_S_TSELINUSE, T-Selector is already in use

  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_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_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 a connection
  - Calling `spi_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 `spi_get_event` on a connection that has been terminated
- You are trying to enable tracing using `spi_trace_open` or `spi_trace_start`when you have already enabled tracing either by defining *osak_trace* or by making a previous `spi_trace_open` call (with no intervening `spi_trace_close` call).

  Call `spi_trace_close` and `spi_trace_stop`, to close the existing trace files and to stop tracing. After this, call `spi_trace_open` and `spi_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 `spi_redirect`. It indicates that the OSAK interface holds unused buffers passed from your application before the call to `spi_redirect`. You cannot redirect a connection in this situation. You should reclaim the buffers and make the call to `spi_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 `spi_give_buffers` or `spi_select`.

OSAK_S_INVPARAM is returned by `spi_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.

`spi_select` returns OSAK_S_INVPARAM 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_INVPORT, the port identifier is invalid

You have specified a port that does not exist. Examine the *port* argument.

OSAK_S_INVPV, the protocol versions are invalid

The *protocol_versions* parameter contains illegal values.

OSAK_S_INVREASON, the reason code is invalid

You have specified a reason code that is not valid for the service you are using it on. Examine the *reason* parameter. Chapter 6 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 OSAK_S_INVAEI.

OSAK_S_INVSYNCPNT, the synchronization point serial number is invalid

Returned by `spi_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 connection.

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 OSAK_S_INVAEI.

OSAK_S_MULTADDR, multiple upper layer addresses fort-selector

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_NOBUFFERS, there are not enough user data buffers

This status is returned by `spi_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_NODATA, there is no data specified

Returned by `spi_data_req`, `spi_typed_req`, `spi_expedited_req`,and `spi_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 `spi_redirect`.

Examine the *process-id* and *process_name* parameters. In a request for redirection of a connection, 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_NOSYNCPNT, the synchronization point serial number is missing

This status code is returned by `spi_connect_req` and `spi_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 *osak_status2* field when the primary status is OSAK_S_INVAEI. See 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 `spi_get_handle` before calling `spi_connect_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:

```
spi_act_interrupt_req
spi_act_interrupt_rsp
spi_act_discard_req
spi_act_discard_rsp
spi_token_give_req
spi_control_give_req
```

These calls return the status code OSAK_S_OVERFLOW if you try to send any user data.

This status code is also returned when user data is included in the `spi_reject_rsp` call and the reason given for the rejection is anything other than user specified.

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 OSAK_S_INVFUNC.

OSAK_S_REDIRECTERR, error occurred while redirecting

This code can be returned only by `spi_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

This code may be returned as a secondary status in the *osak_status2* field when the primary status is OSAK_S_REDIRECTERR. See OSAK_S_REDIRECTERR.

OSAK_S_TOOMANYREDIRECTS, Tried to exceed maximum number of simultaneous redirects

This code may be returned as a secondary status in the *osak_status2* field when the primary status is OSAK_S_REDIRECTERR. 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 the Transport layer on an OpenVMS system may not be the same as the message from the Transport layer 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 field *osak_status2* 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 4. Disruptive Events

This chapter explains what the OSI Applications Kernel (OSAK) interface does when a disruptive event occurs and what action, if any, you should take.

The following disruptive events may occur:

- ABORT request (from the local peer entity)
- ABORT indication (from the remote peer entity)
- PREPARE (RESYNC)
- S-ACTIVITY-DISCARD indication
- S-ACTIVITY-INTERRUPT indication
- S-EXCEPTION-REPORT indication
- S-RESYNCHRONIZE indication
- Transport connection loss

4.1. ABORT request (Local Peer Abort)

This event is fatal to a connection.

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 `spi_close_port` or `spi_collect_pb`. Then set up the connection and send the data again.

4.2. ABORT indication (Remote Peer Abort)

This event is fatal to a connection.

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.

4.3. Transport Connection Loss

This event is fatal to a connection. 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.

4.4. S-ACTIVITY-INTERRUPT indication

This event is not fatal to a connection.

If you are sending data segments when an S-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 an S-ACTIVITY-INTERRUPT indication arrives, keep the data segments you have already received until an S-ACTIVITY-RESUME indication arrives.

4.5. S-ACTIVITY-DISCARD indication

This event is not fatal to a connection.

When you receive an S-ACTIVITY-DISCARD indication, ignore the data that has already arrived on the activity being discarded and reclaim the buffers you posted to receive it.

4.6. S-RESYNCHRONIZE indication

This event is not fatal to a connection.

When you receive an S-RESYNCHRONIZE indication, keep data segments that arrived before the synchronization point specified in the indication. Ignore data segments received after that synchronization point.

4.7. S-EXCEPTION-REPORT indication

This event is not fatal to a connection.

When you receive a S-EXCEPTION-REPORT indication, follow the procedures you have defined for dealing with exception reports.

4.8. PREPARE (RESYNC)

This event is not fatal to a connection.

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 5. How the OSAK SPI Implements the ISO Standards

This chapter explains how the OSAK SPI 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 SPI is based. It is important that you have access to copies of all the standards documents.

5.1. The OSAK SPI and the ISO Protocol Definitions

The OSAK SPI conforms to the following ISO protocol definition:

- ISO 8327 *Information Processing Systems — Open Systems Interconnection — Basic Connection Oriented Session Protocol Specification*

ISO service definitions are given in the following document:

- ISO 8326 *Information Processing Systems — Open Systems Interconnection — Basic Connection Oriented Session Service Definition*

With the exception of the items listed in Section 5.2, the OSAK SPI 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 `spi_data_req` and `spi_typed_req`.
- The OSAK implementation of the receiving side of a connection conforms to the NIST agreement.

5.2. Restrictions in the OSAK Implementation of the ISO Protocol Definitions

The OSAK SPI has the following restrictions:

- The OSAK SPI does not support the symmetric synchronize functional unit defined in Addendum 1 to *ISO 8327*
- The data overflow parameter in the S-CONNECT request is not supported.

Chapter 6. Possible Values for OSAK Data Types

This chapter lists the possible values of all the OSI Applications Kernel (OSAK) parameters that have constant values. The parameter names are in alphabetical order.

6.1. Data Type: osak_abort_reason

These values can only be received.

Constant	Meaning
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	The ACSE user is aborting the connection.
OSAK_C_ABORT_DISCONNECT	The transport connection has been lost.

6.2. Data Type: osak_action_result

Constant	Meaning
OSAK_C_ACCEPT	Acceptance of request to release a connection.
OSAK_C_REJECT	Rejection of request to release a connection.

6.3. 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.
OSAK_C_ACTIVITY_PROCEDURAL	A procedural error has occurred.
OSAK_C_ACTIVITY_DEMAND	The data token is required.

6.4. Data Type: osak_exception_reason

The exception originates from the 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.

Constant	Meaning
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.

6.5. Field: pm_state

This is a field of the *osak_state* data type.

Constant	Meaning
OSAK_C_ASSOCIATE_IND	The process has received a connection indication, but has not responded to it.
OSAK_C_PARTIAL_ASSOC_IND	The process has received a connection indication with incomplete user data or no user data.
OSAK_C_DATA_TRANSFER	The process has established a connection and is transferring data.

6.6. Data Type: osak_reject_reason

Possible values vary according to the source of the rejection of a connection request. The rejection may originate from either of the following sources:

- The service user
- The service provider

6.6.1. Rejection Originating from User

Constant	Meaning
OSAK_C_REJ_NOREASON[1]	No reason is given.
OSAK_C_REJ_SP_NO_REASON	No reason is given.
OSAK_C_REJ_SP_USER_CONGESTED	Temporary congestion
OSAK_C_REJ_SP_USER_REASON	User specified reason

[1]User can specify this in the call, but this value is never received because it is converted to OSAK_C_REJ_SP_NO_REASON.

You can give any of these values in a call to `spi_reject_rsp`.

6.6.2. 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.

Constant	Meaning
OSAK_C_REJ_SP_UNSUPPORTED	The proposed session protocol version is not supported.
OSAK_C_REJ_SP_REFUSED	The session protocol machine has rejected the connection attempt.
OSAK_C_REJ_SP_RESTRICTION	The connection is rejected by the session protocol machine due to an implementation restriction stated in the OSAK PICS.

6.7. Field: request_returned_mask

These are fields of the *osak_handle* data type.

Constant	Meaning
OSAK_C_READEVENT	The routine `spi_select` writes this value when an inbound event has occurred.
OSAK_C_WRITEEVENT	The routine `spi_select` writes this value when an outbound event has been completed.

6.8. Parameter: osak_resync_type

Constant	Meaning
OSAK_C_RESYNC_ABANDON	The OSAK interface resynchronizes to a synchronization point the serial number of which is higher than the serial numbers of synchronization points in use on the existing connection.
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.

6.9. Fields: data, sync_mimnor, major_activity, and release

These are fields of the *osak_token_setting* 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.

6.10. 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