SoftStax Programming Reference

Version 3.6

www.radisys.com

World Headquarters
5445 NE Dawson Creek Drive • Hillsboro, OR
97124 USA
Phone: 503-615-1100 • Fax: 503-615-1121
Toll-Free: 800-950-0044

International Headquarters
Gebouw Flevopoort • Televisieweg 1A
NL-1322 AC • Almere, The Netherlands
Phone: 31 36 5365595 • Fax: 31 36 5365620

RadiSys Microware Communications Software Division, Inc.
1500 N.W. 118th Street
Des Moines, Iowa 50325
515-223-8000

Revision A
November 2001

Copyright and publication information

This manual reflects version 3.6 of SoftStax. Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from RadiSys Microware Communications Software Division, Inc.

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, RadiSys Corporation will not be liable for any damages including indirect or consequential, from use of the OS-9 operating system, Microware-provided software, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

Reproduction notice

The software described in this document is intended to be used on a single computer system. RadiSys Corporation expressly prohibits any reproduction of the software on tape, disk, or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of RadiSys Corporation and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

November 2001

Table of Contents

Chapter 1: ITEM Library

This document contains descriptions, in alphabetical order, of the telecommunications Application Programming Interface (API) functions in the Integrated Telephony Environment for Multimedia (ITEM) library.

Function Descriptions

The function descriptions in this manual are, for the most part, self-explanatory. Each section of a function description is defined below.

The **Syntax** section shows the function prototype with the required parameters and their data types. This section also lists the files that you need to include when using the function and coding dependencies.

The **OS-9 Attributes** section lists various attributes of each function in relation to OS-9—including whether the function is compatible with OS-9 and/or OS-9 for 68K; whether the function is in user state and/or system state; and whether the function is safe for use in a threaded application.

The **Libraries** section lists the name of the library in which you can find the function.

The **Description** section provides a description of the function.

The **Parameters** section provides details about each of the parameters.

Direct Errors are errors that are detected within the library call and are a direct result of that particular call.

Indirect Errors are the result of invalid parameter values passed to and detected by another function call. They are not directly returned by the original calling function.

The **See Also** section tells you about related functions or materials providing more information about the function.

ite_ctl_addrset()

Sets ITEM Address Information

Syntax

```
#include <SPF/item.h>
error_code ite_ctl_addrset(
   path_id      path,
   addr_type    *our_num,
   addr_type    *their_num);
```

Libraries

`item.l`

Description

`ite_ctl_addrset()` allows you to set the address information for the ITEM path. Specifically, it sets the `dev_ournum` and `dev_theirnum` structures in the item path descriptor. If `our_num` or `their_num` is set to `NULL`, the corresponding value of `dev_ournum` and `dev_theirnum` respectively is left unchanged.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

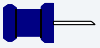
Parameters

`path`	contains a handle identifying the input/output (I/O) path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc. For in-band protocols, the path is also used to read and write data. For out-of-band protocols, the path is used only as the signalling path; it is not used to read and write data.

`our_num`	points to the caller-allocated `addr_type` structure set up to represent the new address information for your device.
`their_num`	points to the caller-allocated `addr_type` structure. It is set up to represent the new address of the far-end device with which to communicate.

Indirect Errors

`EOS_ILLPRM`	`addr_size` field in `addr_type` structure is greater than 32 bytes.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path was lost and is no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.



Note

Other errors may be returned by SoftStax drivers.

See Also

`item.h` in ***Using SoftStax*** for information about the `addr_type` structure.

ite_ctl_answer()

Answers Incoming Call

Syntax

```
#include <SPF/item.h>
error_code ite_ctl_answer(
   path_id      path,
   ite_cctl_pb  *ccpb,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_ctl_answer()` allows a process to answer an incoming call on the specified path. The path must have previously performed an `ite_ctl_rcvrasgn()` call to enable notification of incoming calls.

Alternatively, the application can continuously poll using the `ite_ctl_connstat()` call until the `dev_callstate` field equals `ITE_CS_INCALL`.

If the `npb` parameter is `NULL`, control is returned to the caller after the call control procedure puts this connection into the active state, indicating an end-to-end connection is being established.

If a notification parameter block is passed in, the driver sends notification when the end-to-end connection has been confirmed by the network. In either case, before answering, the `ite_ctl_connstat()` call can be used to screen the call or get the display information, if there is any, for the incoming call (that is, caller ID).

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc. For in-band protocols, the path is also used to read and write data. For out-of-band protocols, the path is used only as the signalling path; it is not used to read and write data.
`ccpb`	points to the call control parameter block. This parameter block provides additional information from the application to the driver responsible for performing the answer. If no additional information is needed, the application may pass `NULL` for this pointer.
`npb`	points to the notification parameter block telling ITEM the type of notification the caller requests when the connection is established.

For More Information

Refer to the `notify_type` **Structure** section in ***Using SoftStax*** for information about setting up the notification parameter block.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.
`EOS_PTHLOST`	returned when the path was lost and is no longer valid.

EOS_TSTATE	path is in the wrong state to answer incoming call.
EOS_UNKSVC	returned when drivers for connection-oriented ITEM calls (connect, disconnect, or answer) are not connected.

See Also

ite_ctl_addrset()
ite_ctl_connect()
ite_ctl_connstat()
ite_ctl_disconnect()

ite_ctl_connect()

Establishes an
End-to-end Connection

Syntax

```
#include <SPF/item.h>
error_code ite_ctl_connect(
   path_id      path,
   addr_type    *ournum,
   addr_type    *theirnum,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_ctl_connect()` sets the `dev_ournum` and `dev_theirnum` structures in the `item` path descriptor, to the `addr_type` structures that you passed in. This is only done if the `ournum` or `theirnum` pointer is not `NULL`. Then, this function establishes an end-to-end connection from the ITEM device referenced by path to the far-end device with the address found in the `dev_theirnum` structure of your device. If the `ournum` and/or `theirnum` pointers passed in are `NULL`, ITEM uses the default addresses found in the path descriptor.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

	For in-band protocols, the path is also used to read and write data. For out-of-band protocols, the path is used only as the signalling path; it is not used to read and write data.
`ournum`	points to the caller-allocated `addr_type` structure set up to represent the new local address information for your device.
`theirnum`	points to the caller-allocated `addr_type` structure. It is set up to represent the new remote address of the far-end device with which to communicate.
`npb`	points to the notification parameter block. Tells ITEM the type of notification the caller requests upon establishing a connection.

For More Information

Refer to the `notify_type` **Structure** section in ***Using SoftStax*** for information about setting up the notification parameter block.

Direct Errors

`EOS_ILLPRM`	the notify parameter block (`npb`) passed in is `NULL`.

Indirect Errors

`EOS_DEVBSY`	returned when trying to connect to a path that already has, or is establishing, a connection. (The `device_type` cellstate is `ITE_CS_CONNECT, ITE_CS_ACTIVE`, or `ITE_CS_CONNTERM`). In order for a connection to be successful, the path must be in the `ITE_CS_IDLE` state.

EOPNOTSUPP	returned when trying to connect on a connectionless protocol.
EOS_UNKSVC	returned if no protocol on the stack performs cell control functions.
EOS_BPNUM	returned when the path number is invalid.
EOS_PTHLOST	returned when the path was lost and is no longer valid.

See Also

ite_ctl_addrset()
ite_ctl_answer()
ite_ctl_connstat()
ite_ctl_disconnect()

ite_ctl_connstat()

Returns Device Type Status Information

Syntax

```
#include <SPF/item.h>
error_code ite_ctl_connstat(
   path_id        path,
   device_type    *dev_info);
```

Libraries

```
item.l
```

Description

`ite_ctl_connstat()` returns a copy of the `dev_type` structure for the specified path. This structure contains information about the our-end and far-end connection addresses, call state, display information, and the network type of the device.

For More Information

Refer to the `device_type` **Structure** section in ***Using SoftStax*** for information about the fields in that structure.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc. For in-band protocols, the path is also used to read and write data. For out-of-band protocols, the path is used only as the signalling path; it is not used to read and write data.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_BPADD`	returned if `dev_info` is `NULL` or points to memory not owned by the process. This error is only returned on systems with SSM (system security module) running.
`EOS_PTHLOST`	returned when the path was lost and is no longer valid.

See Also

`item.h` in ***Using SoftStax*** for information about the `dev_type` structure.

`ite_ctl_addrset()`

ite_ctl_disconnect()

Disconnects End-to-end Connection

Syntax

```
#include <SPF/item.h>
error_code ite_ctl_disconnect(
   path_id     path,
   ite_cctl_pb  *ccpb);
```

Libraries

`item.l`

Description

`ite_ctl_disconnect()` disconnects the end-to-end connection established by `ite_ctl_connect()`.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc. For in-band protocols, the path is also used to read and write data. For out-of-band protocols, the path is used only as the signalling path; it is not used to read and write data.
`ccpb`	points to the call control parameter block. This parameter block provides additional information from the application to the driver for performing

the disconnection. If no additional information is needed, the application can pass `NULL` for this pointer.

Indirect Errors

`EOS_UNKSVC`	returned if no protocol on the stack performs cell control functions.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_ctl_connect()`

ite_ctl_rcvrasgn()

Sets Up Path for Incoming Calls

Syntax

```
#include <SPF/item.h>
error_code ite_ctl_rcvrasgn(
   path_id      path,
   addr_type    *their_num,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_ctl_rcvrasgn()` sets up the calling process to receive notification of an incoming call. If a notification has already been registered on this path, this call returns `EOS_DEVBSY`. You can set up only one process per path to receive incoming calls.

If `their_num` is `NULL`, the path receives a notification for any incoming call.

If `their_num` is not `NULL`, the path only receives notification for an incoming call with a calling address matching the `their_num` address.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path` contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

For in-band protocols, the path is also used to read and write data.

	For out-of-band protocols, the path is used only as the signalling path; it is not used to read and write data
`their_num`	contains either a `NULL` value, or points to the `addr_type` structure. The caller allocates and sets up this structure to receive incoming calls from a specified address. If `NULL` is used, the application receives notification on any incoming call.
`npb`	points to the notification parameter block structure ITEM uses to send notification to the caller.

For More Information

Refer to ***Using SoftStax*** for information about the `notify_type` structure.

Direct Errors

`EOS_ILLPRM`	returned when `npb` is `NULL`.

Indirect Errors

`EOS_ILLPRM`	returned if the `addr_size` parameter in the `their_num` structure is greater than 32.
`EOS_DEVBSY`	returned if any process has already registered for receiving notification of an incoming call.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_ctl_rcvrrmv()`

ite_ctl_rcvrrmv()

Removes Notification Request

Syntax

```
#include <SPF/item.h>
error_code ite_ctl_rcvrrmv(path_id path);
```

Libraries

`item.l`

Description

`ite_ctl_rcvrrmv()` removes the notification request created with `ite_ctl_rcvrasgn()`. This call resolves successfully even if the application executes the call without a prior notification assignment.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc. For in-band protocols, the path is also used to read and write data. For out-of-band protocols, the path is used only as the signalling path; it is not used to read and write data.

Indirect Errors

`EOS_BPNUM`	returned on a bad path number.
`EOS_BPNUM`	returned when the path number is invalid.

`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_ctl_rcvrasgn()`

ite_data_avail_asgn()

Notifies Path of Incoming Data

Syntax

```
#include <SPF/item.h>
error_code ite_data_avail_asgn(
   path_id      path,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_data_avail_asgn()` sets up the calling process to receive a notification when incoming data is available for the specified path.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`npb`	points to the notification parameter block structure ITEM uses to send notification to the caller.

For More Information

Refer to ***Using SoftStax*** for information about the `notify_type` structure.

Direct Errors

`EOS_ILLPRM`	returned when `npb` is `NULL` or, if requesting an event, the `ntfy_evid` is `0`.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_data_avail_rmv()`
`ite_data_ready()`

`_os_ss_sendsig()` in the ***OS-9 Technical Manual***

ite_data_avail_rmv()

Removes Notification Request

Syntax

```
#include <SPF/item.h>
error_code ite_data_avail_rmv(path_id path);
```

Libraries

`item.l`

Description

`ite_data_avail_rmv()` removes the notification request created with `ite_data_avail_asgn()`. This call resolves successfully even if the application executes it without a prior notification assignment.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path was lost and is no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_data_avail_asgn()`

`_os_ss_relea()` in the ***OS-9 Technical Manual***.

ite_data_read()

Reads From a Path

Syntax

```
#include <SPF/item.h>
error_code ite_data_read(
   path_id   path,
   void      *buffer,
   u_int32   *count);
```

Libraries

`item.l`

Description

`ite_data_read()` performs a read on a path. Data is removed from the path's receive queue and passed back to the caller in `buffer`. The read operation depends on the I/O options of the path, (`pd_ispacket`, `pd_ioasync`, and `pd_iotime`) which may be changed via the `_os_ss_popt()` call.

For More Information

Refer to ***Using SoftStax*** for more information about `pd_iopacket`, `pd_ioasync`, and `pd_iotime`.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`buffer`	points to the caller-allocated buffer in which to place the data. SoftStax copies the data from the receive queue into this buffer.
`count`	points to the number of bytes to read. When the call is completed, the address that is pointed to is updated with the number of bytes SoftStax copies into the buffer.

Indirect Errors

`EOS_DEVBSY`	returned when another process is already waiting for incoming data on this path.
`EOS_NOTRDY`	returned when the SoftStax driver's `lu_ioenabled` flag is `0`. This error indicates either that the protocol stack contains a driver that is not initialized or the end-to-end protocol has an error causing a break in the end-to-end protocol link.
`EOS_SIGNAL`	returned when a fatal signal is received.
`ETIMEOUT`	returned when the read request timed out before completion.
`EWOULDBLOCK`	returned when a read request is made, but there is currently not enough data available to actually read. This error occurs only if the `IO_READ_ASYNC` bit is set in the `pd_ioasync` byte in the path options, indicating the read side of the path is in asynchronous mode.
`E_BPADR`	returned if buffer is `NULL` or does not point to memory owned by the calling process.

EOS_RXMB_ERR	returned when data being read into the application buffer is flagged as incorrect by the received data hardware. This could happen as a result of bad CRC, overflow, abort sequence, or bad hardware. At this point, the application should request hardware statistics to discover the source of the error. The data is returned through the read call. The hardware error occurred somewhere in the current received buffer.
EOS_BPNUM	returned when the path number is invalid.
EOS_PTHLOST	returned when the path is lost and no longer valid.
EOS_PPS_NOTFND	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`_os_gs_popt()` and `_os_ss_popt()` in the ***OS-9 Technical Manual***

ite_data_readmbuf()

Gets mbuf Data

Syntax

```
#include <SPF/item.h>
error_code ite_data_readmbuf(
   path_id   path,
   Mbuf      *mb_ptr);
```

Libraries

`item.l`

Description

`ite_data_readmbuf()` allows the caller to get the buffer SoftStax uses to store the mbuf (incoming packets) instead of passing in a user buffer with data copied into it. This call returns one mbuf packet chain in `mb_ptr`. This call facilitates true zero copy reads for faster throughput. SoftStax gives the caller permissions to access the returned mbuf packet chain.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`mb_ptr`	contains the pointer to the mbuf packet chain.

Indirect Errors

`EOS_DEVBSY`	returned when another process is already waiting for incoming data on this path.

EOS_NOTRDY	returned when the SoftStax driver's `lu_ioenabled` flag is `0`. This error indicates that the protocol stack contains a driver that is not initialized or the end-to-end protocol has an error causing a break in the end-to-end protocol link.
EOS_SIGNAL	returned when a fatal signal is received.
ETIMEOUT	returned when the read request timed out before completion.
EWOULDBLOCK	returned when a read request is made but there is currently not enough data available to read. This error only occurs if the `IO_READ_ASYNC` bit is set in the `pd_ioasync` byte in the path options, indicating the read side of the path is in asynchronous mode.
EOS_BPNUM	returned when the path number is invalid.
EOS_PTHLOST	returned when the path is lost and no longer valid.
EOS_PPS_NOTFND	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_data_read()`
`ite_data_writembuf()`

The Mbuf facility in ***Using SoftStax***.

ite_data_ready()

Returns Number of Bytes Available

Syntax

```
#include <SPF/item.h>
error_code ite_data_ready(
   path_id   path,
   u_int32   *avail_count);
```

Libraries

`item.l`

Description

`ite_data_ready()` counts the number of bytes on the specified path that are available for reading. An `EOS_NOTRDY` error is returned if there is no data available.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`avail_count`	points to a `u_int32` field where the number of bytes available for reading is returned.

Indirect Errors

`EOS_NOTRDY`	returned if no data is ready to read.
`EOS_BPNUM`	returned when the path number is invalid.

`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`_os_gs_ready()` in the ***OS-9 Technical Manual***

ite_data_recvfrom

Receive Data from a Specified Location

Syntax

```
#include <SPF/item.h>
error_code ite_data_recvfrom(
   path_id      path,
   void         *buffer,
   u_int32      size,
   u_int32      flags,
   addr_type    *recvfrom_addr);
```

Libraries

`item.l`

Description

`ite_data_recvfrom()` allows the caller to send data to a particular remote location.

WARNING

The call might block if the `IO_WRITE_ASYNCH` bit in the `pd_ioasync` byte in the path options structure is set and there are no mbufs available in the mbuf pool.

Attributes

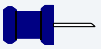
Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`buffer`	points to the data to be transmitted.
`size`	is the number of valid bytes of data in the buffer.
`flags`	is not used.
`sendto_addr`	points to the caller-allocated `addr_type` structure. It is set up to represent the remote address of the far-end device to which the caller is sending data.

Indirect Errors

`ENOBUFS`	returned when both the `IO_WRITE_ASYNC` bit in the `pd_ioasync` byte is set and the transmit mbuf cannot be allocated for the data being written. This indicated that either the mbuf pool is empty or the system cannot allocate an mbuf large enough for this transport packet.
`EOS_NOTRDY`	returned when the SoftStax driver's `lu_ioenabled` flag is `0`, indicating the protocol stack contains a driver that is not initialized or the end-to-end protocol has an error causing a break in the end-to-end protocol link.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.



Note

Other errors may be returned by the SoftStax drivers.

ite_data_sendto()

Send Data to a Specified Location

Syntax

```
#include <SPF/item.h>
error_code ite_data_sendto(
   path_id      path,
   void         *buffer,
   u_int32      size,
   u_int32      flags,
   addr_type    *sendto_addr);
```

Libraries

`item.l`

Description

`ite_data_sendto()` allows the caller to send data to a particular remote location.

WARNING

The call might block if the `IO_WRITE_ASYNCH` bit in the `pd_ioasync` byte in the path options structure is set and there are no mbufs available in the mbuf pool.

Attributes

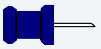
Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`buffer`	points to the data to be transmitted.
`size`	is the number of valid bytes of data in the buffer.
`flags`	is not used.
`sendto_addr`	points to the caller-allocated `addr_type` structure. It is set up to represent the remote address of the far-end device to which the caller is sending data.

Indirect Errors

`ENOBUFS`	returned when both the `IO_WRITE_ASYNC` bit in the `pd_ioasync` byte is set and the transmit mbuf cannot be allocated for the data being written. This indicated that either the mbuf pool is empty or the system cannot allocate an mbuf large enough for this transport packet.
`EOS_NOTRDY`	returned when the SoftStax driver's `lu_ioenabled` flag is `0`, indicating the protocol stack contains a driver that is not initialized or the end-to-end protocol has an error causing a break in the end-to-end protocol link.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.



Note

Other errors may be returned by the SoftStax drivers.

ite_data_write()

Writes Data to Packet

Syntax

```
#include <SPF/item.h>
error_code ite_data_write(
   path_id   path,
   void      *buffer,
   u_int32   *count);
```

Libraries

`item.l`

Description

`ite_data_write()` causes SoftStax to create a packet out of the buffer that is to be written. In addition, it sends the packet down the protocol chain for the path until the hardware driver queues it up for transmission.

WARNING

The call might block if the `IO_WRITE_ASYNCH` bit in the `pd_ioasync` byte in the path options structure is set and there are no mbufs available in the mbuf pool.

A return from this call does not ensure the data has been transmitted. It only indicates that the data has been queued for transmission by the device.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`buffer`	points to the data to be transmitted.
`count`	points to the number of valid bytes of data in the buffer.

Indirect Errors

`ENOBUFS`	returned when both the `IO_WRITE_ASYNC` bit in the `pd_ioasync` byte is set and the transmit mbuf cannot be allocated for the data being written. This indicated that either the mbuf pool is empty or the system cannot allocate an mbuf large enough for this transport packet.
`EOS_NOTRDY`	returned when the SoftStax driver's `lu_ioenabled` flag is `0`, indicating the protocol stack contains a driver that is not initialized or the end-to-end protocol has an error causing a break in the end-to-end protocol link.
`E_BPADDR`	returned if buffer is `NULL`.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

ite_data_writembuf()

Writes Mbuf-packet

Syntax

```
#include <SPF/item.h>
error_code ite_data_writembuf(
   path_id   path,
   Mbuf      mb_ptr);
```

Libraries

`item.l`

Description

`ite_data_writembuf()` accepts an mbuf for transmission through the protocol stack and out to the hardware device. For best efficiency, the caller should save enough space at the beginning of the mbuf to allow the protocol stack to add headers to the same mbuf passed in. The caller can obtain the number of bytes the stack needs by getting the path options and looking at the `pd_txoffset` field. Otherwise, the protocol must find another mbuf in which to place the header information and chain those packets together.

Drivers that cannot operate correctly on mbuf chains are unable to process this call if there is not enough space for all headers at the beginning of the mbuf.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

`mb_ptr`	points to the mbuf packet chain that is to be transmitted.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_data_readmbuf()`
`ite_data_write()`

The Mbuf facility in ***Using SoftStax***

ite_dev_attach()

Attaches a Device

Syntax

```
#include <SPF/item.h>
error_code ite_dev_attach(
   char      *name,
   u_int32   mode,
   u_int32   *handle);
```

Libraries

`item.l`

Description

`ite_dev_attach()` performs a simple attach of a device into ITEM.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

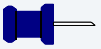
Parameters

`name`	points to the name of the device descriptor with which to attach.
`mode`	contains the mode in which to open the device. Recommended default value for mode is `FAM_READ` or `FAM_WRITE`.
`handle`	points to the unique ID returned by ITEM. It identifies this device.

Indirect Errors

`EOS_UNKSVC`	returned when the `sysmbuf` utility is not installed.

EOS_MNF	returned when the driver, descriptor, or file manager is not loaded into memory.



Note

Drivers may return their own error codes. Refer to the specific driver's description for more information.

See Also

`ite_dev_detach()`

`_os_attach()` in the ***Ultra C Library Reference Manual***

ite_dev_detach()

Detaches Device

Syntax

```
#include <SPF/item.h>
error_code ite_dev_detach(u_int32 handle);
```

Libraries

`item.l`

Description

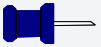
`ite_dev_detach()` allows you to detach a device from ITEM. If the attach count is greater than `1` (was attached multiple times by multiple applications), only the attach count of the device is decremented; thus, the device remains in the set of active devices used by ITEM. When the attach count is `1` (the last application detaches from the ITEM device), the device is removed from the set of initialized devices used by ITEM.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`handle`	contains the unique ID returned by ITEM. It identifies this device.



Note

Drivers may return their own error codes. Refer to the specific driver's description for more information.

See Also

`ite_dev_attach()`

`_os_detach()` in the ***Ultra C Library Reference Manual***.

ite_dev_getmode()

Gets Device Mode

Syntax

```
#include <SPF/item.h>
error_code ite_dev_getmode(
   path_id   path,
   u_int16   *mode);
```

Libraries

`item.l`

Description

`ite_dev_getmode()` allows you to determine the mode of the device using the specified path.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`mode`	points to where SoftStax returns the mode of the device the path is using. The possible bit field values for mode are: `FAM_READ`, `FAM_WRITE`, or `FAM_NONSHARE`.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.

See Also

`modes.h`, which contains macros for the mode bit fields

ite_dev_getname()

Gets Device Name

Syntax

```
#include <SPF/item.h>
error_code ite_dev_getname(
   path_id   path,
   char      *name);
```

Libraries

`item.l`

Description

`ite_dev_getname()` allows you to determine the name of the device using the specified path.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`name`	points to the user allocated buffer where SoftStax puts the device name string.

Note

In OS-9, the buffer pointed to by `name` must be at least 64 bytes to accommodate the device name. In OS-9 for 68K processors, the buffer should be at least 32 bytes.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.

ite_dev_gettype()

Gets Device Type

Syntax

```
#include <SPF/item.h>
error_code ite_dev_gettype(
   path_id   path,
   u_char    *type_in,
   u_char    *type_out);
```

Libraries

`item.l`

Description

`ite_dev_gettype()` allows you to determine the type of the device using the specified path.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`type_in`	points to where SoftStax puts the type of the receive-side of the ITEM device.
`type_out`	points to where SoftStax places the type of the transmit-side of the ITEM device.

Table 1-1 Device Types

Value	Description
`ITE_NET_NONE`	If only one side has this value, the device is uni-directional.
`ITE_NET_CTL`	Set-top box control channel device.
`ITE_NET_DATA`	Set-top box data channel device.
`ITE_NET_MPEG2`	MPEG-2 device.
`ITE_NET_CHMGR`	Set-top box channel management device.
`ITE_NET_OOB`	Out-of-band code signalling device.
`ITE_NET_ANY`	Generic network device. No specific payload is required.
`ITE_NET_VIPDIR`	Video Information Provider Directory
`ITE_NET_SESCTL`	Session control device.
`ITE_NET_X25`	X.25 network device.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and is no longer valid.

See Also

`item.h` and ***Using SoftStax*** for the `device_type` structure

ite_dev_setmode()

Sets Device Mode

Syntax

```
#include <SPF/item.h>
error_code ite_dev_setmode(
   path_id   path,
   u_int16   mode);
```

Libraries

`item.l`

Description

`ite_dev_setmode()` allows you to set the device mode.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`mode`	contains the mode to set. Bit field values for mode include the following: `FAM_READ`, `FAM_WRITE`, or `FAM_NONSHARE`.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.

`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_dev_getmode()`

`item.h` and ***Using SoftStax*** for the `device_type` structure and `modes.h` for the macros that define the mode bits.

ite_fehangup_asgn()

Registers Caller for Notification of Far End Disconnect

Syntax

```
#include <SPF/item.h>
error_code ite_fehangup_asgn(
   path_id      path,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_fehangup_asgn()` registers the caller for notification of disconnection by the far-end.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`npb`	points to the notification parameter block structure ITEM uses to send notification to the caller.

For More Information

Refer to ***Using SoftStax*** for information about the `notify_type` structure.

Indirect Errors

`EOS_ILLPRM`	returned when `npb` is `NULL`.
`EOS_DEVBSY`	returned if another process has already registered for this notification on the path.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_fehangup_rmv()`

ite_fehangup_rmv()

Removes Request for Far-end Hang-up

Syntax

```
#include <SPF/item.h>
error_code ite_fehangup_rmv(path_id path);
```

Libraries

`item.l`

Description

`ite_fehangup_rmv()` removes the notification request for far-end disconnection. This call resolves successfully even if the application executes it without a prior notification assignment.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_fehangup_asgn()`

ite_linkdown_asgn()

Notifies Caller of Link Failure

Syntax

```
#include <SPF/item.h>
error_code ite_linkdown_asgn(
   path_id     path,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_linkdown_asgn()` tells SoftStax drivers to notify the caller if the link fails at any layer of the protocol stack.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`npb`	points to the notification parameter block structure ITEM uses to send notification to the caller.

For More Information

Refer to ***Using SoftStax*** for information about the `notify_type` structure.

Indirect Errors

`EOS_ILLPRM`	returned when `npb` is `NULL`.
`EOS_DEVBSY`	returned if another process has already registered for this notification on the path.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_linkdown_rmv()`

ite_linkdown_rmv()

Removes Linkdown Notification Assignment

Syntax

```
#include <SPF/item.h>
error_code ite_linkdown_rmv(path_id path);
```

Libraries

`item.l`

Description

`ite_linkdown_rmv()` removes the previous link failure notification assignment created with the `ite_linkdown_asgn()` call. This call returns successfully even if the application executes it without a prior notification assignment.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_linkdown_asgn()`

ite_linkup_asgn

Notifies Caller of Link Failure

Syntax

```
#include <SPF/item.h>
error_code ite_linkup_asgn(
   path_id     path,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_linkup_asgn()` tells SoftStax drivers to notify the caller if the link fails at any layer of the protocol stack.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`npb`	points to the notification parameter block structure ITEM uses to send notification to the caller.

For More Information

Refer to ***Using SoftStax*** for information about the `notify_type` structure.

Indirect Errors

`EOS_ILLPRM`	returned when `npb` is `NULL`.
`EOS_DEVBSY`	returned if another process has already registered for this notification on the path.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_linkup_rmv`

ite_linkup_rmv

Removes Linkup Notification Assignment

Syntax

```
#include <SPF/item.h>
error_code ite_linkup_rmv(path_id path);
```

Libraries

`item.l`

Description

`ite_linkup_rmv()` removes the previous link failure notification assignment created with the `ite_linkup_asgn()` call. This call returns successfully even if the application executes it without a prior notification assignment.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

ite_linkup_asgn

ite_path_clone()

Duplicates a Path - Independent Connection

Syntax

```
#include <SPF/item.h>
error_code ite_path_clone(
   path_id      dup_path,
   path_id      *new_path,
   notify_type  *npb);
```

Libraries

`item.l`

Description

`ite_path_clone()` creates a new path with the same device type information, properties, and connection state as the original path. It assigns a new path identifier to the newly cloned path.

`ite_path_clone()` is similar to `ite_path_dup()`, except that if the original path has an active connection, the `ite_clone_path()` call creates a new connection to the same two endpoints as the original path. If the paths are duplicated (with `ite_path_dup()`), both paths share a connection. Also, unlike duplicated paths, cloned paths point to different SoftStax path descriptors.

If you are duplicating a connection-oriented device, this call uses the asynchronous `ite_ctl_connect()` call. In this case, the `ite_path_clone()` call becomes asynchronous and the caller is notified using the `npb` notification. Otherwise, `ite_path_clone()` is synchronous for connectionless devices.

For More Information

Refer to `ite_ctl_connect()` for more information about `npb`.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`dup_path`	specifies the path to clone.
`new_path`	points to the location where the ID of the new path is returned. A new connection is established between the ITEM device and the far-end address, as specified in the `device_type` structure for the original path. This ensures that the `dup_path` and `new_path` do not share a connection.
`npb`	points to the notification parameter block structure ITEM uses to send notification to the caller.

For More Information

Refer to ***Using SoftStax*** for information about the `notify_type` structure.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_DEVBSY`	returned when you are trying to clone a non-sharable device on the second open to the device. You can avoid this error if you call `ite_dev_getmode()` to make sure the path you are cloning is a sharable device.

See Also

ite_ctl_connect()
ite_path_dup()

ite_path_close()

Closes a Path

Syntax

```
#include <SPF/item.h>
error_code ite_path_close(path_id path);
```

Libraries

`item.l`

Description

`ite_path_close()` closes the specified path. If there is a connection to that path, it is terminated.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.

See Also

`ite_path_open()`

ite_path_dup()

Duplicates Path and Shares Connection

Syntax

```
#include <SPF/item.h>
error_code ite_path_dup(
   path_id   dup_path,
   path_id   *new_path);
```

Libraries

`item.l`

Description

`ite_path_dup()` creates a new path ID referencing the same path as the original.

`ite_path_dup()` is similar to `ite_path_clone()`, except that if the original path has an active connection, the `ite_clone_path()` creates a new connection to the same two endpoints used by the original path. Also, unlike cloned paths, duplicated paths point to the same SoftStax path description.

With `ite_path_dup()`, both paths share the same connection. If one path terminates the connection, both paths lose that connection to the endpoint. However, if the new path created by this call is closed, the original path's connection remains open.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`dup_path`	specifies the path to duplicate.
`new_path`	points to where the new paths's ID is returned.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_path_clone()`

ite_path_open()

Opens a Path

Syntax

```
#include <SPF/item.h>
error_code ite_path_open(
   char         *dev_name,
   u_int32      mode,
   path_id      *new_path,
   addr_type    *our_num);
```

Libraries

`item.l`

Description

`ite_path_open()` performs two operations:

- Opens and initializes the device type structure for the path
- Initializes the `our_num` field in the ITEM device's `device_type` structure

For More Information

Refer to the **device_type Structure** section in ***Using SoftStax*** for information about the `device_type` structure and its fields.

The result of the `ite_path_open()` call is that a path to the device pointed to by `dev_name` is returned. If `our_num` is not `NULL`, the call also initializes the `dev_ournum` structure in the ITEM device. If `our_num` equals `NULL`, ITEM uses the default local addressing out of the device descriptor being opened.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`dev_name`	points to the string name specifying the device to which you are opening a path. This string can contain the name of an OS-9 device descriptor that refers to a single protocol driver, a string defining all the protocols to stack on the path, the far end address to convert to, or an OS-9 device descriptor containing the stack to be built.

For More Information

Refer to ***Using SoftStax*** for more details about devices.

`mode`	contains the mode in which to open the device. Possible bit field values for mode include the following: `FAM_READ`, `FAM_WRITE`, or `FAM_NONSHARE`.
`new_path`	points to the location where SoftStax returns the new path ID.
`our_num`	points to the `addr_type` structure allocated by the application to initialize the `dev_ournum` field in the ITEM device you are opening.

For More Information

For valid parameters for the `addr_type` structure, refer to the `item.h` file.

	You can pass the `our_num` pointer in as `NULL`. In this case, the `ite_path_open()` call uses the default address information for the ITEM device.

Indirect Errors

`EOS_EVBSY`	returned when the receive-thread event cannot be created because it already exists.
`EOS_MNF`	returned when the device descriptor, driver, or the SoftStax manager is not loaded.
`EOS_STKFULL`	returned if more than six protocols are being stacked on the path.

See Also

`ite_path_close()`

`item.h` for `addr_type` parameters.

ite_path_pop()

Removes Driver from Top of Stack

Syntax

```
#include <SPF/item.h>
error_code ite_path_pop(path_id);
```

Libraries

`item.l`

Description

`ite_path_pop()` removes the driver from the top of the protocol stack on the path. It does not alter any other protocols on the stack.

For More Information

For information about how SoftStax stacks and unstacks protocols, refer to ***Using SoftStax*** and the ***SoftStax Porting Guide***.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.

Indirect Errors

EOS_BPNUM	returned when the path number is invalid.
EOS_BTMSTK	returned if the last driver on the stack has already been reached and no more pops can take place.
EOS_NOSTACK	returned if there are no protocol drivers on the stack.
EOS_PTHLOST	returned when the path is lost and no longer valid.

See Also

ite_path_pop()

ite_path_profileget()

Get Path Profile

Syntax

```
#include <SPF/item.h>
#include <SPF/spf_oob.h>
error_code ite_path_profileget(
   path_id      path,
   conn_type    *conn,
   u_int32      *pr_size,
   void         *pr_buffer);
```

Libraries

`item.l`

Description

The `conn_type` pointer may be passed in by the application as `NULL` or a pointer to a valid `conn_type` structure. If the `conn_type` pointer is `NULL`, the driver should return the per path storage profile structure (`pp_profile`) in the user buffer. If the `conn_type` pointer is not `NULL`, the `conn_svc_type` field in the `conn_type` pointer is set to the `ITE_SVC_xxx` for the profile the user wants to get. If the service profile is not supported, the driver should return `EOS_UNKSVC`. If the profile number is invalid, the driver should return `EOS_ILLPRM`. Once the profile structure element in the profile array of the logical unit has been determined, the driver should check to make sure the user buffer size is big enough to fit the entire profile. If not, the driver should copy as much of the profile as can be put in the buffer and return `EOS_BADSIZ` error. If the buffer is big enough, the driver should copy the profile into the buffer and return `SUCCESS`.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`conn`	points to the caller-allocated `conn_type` structure set up to represent the connection type information for your device.
`pr_size`	points to the number of bytes in the `pr_buffer` to read.
`pr_buffer`	points to the caller-allocated buffer in which the path profile is placed.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.
`EOS_ILLPATH`	returned when `conn_type` structure is `NULL`, or `conn_svc_type` in `conn_type` structure is invalid.

Note

Other errors may be returned by SoftStax drivers.

See Also

`ite_path_profileset()`

ite_path_profileset()

Syntax

```
#include <SPF/item.h>
#include <SPF/spf_oob.h>
error_code ite_path_profileset(
   path_id   path,
   conn_type *conn,
   u_int32   *pr_size,
   void      *pr_buffer);
```

Description

For the set profile setstat, the user passes in the new parameters for the profile of the path. The driver should attempt to validate the new parameters of the profile as needed to ensure the application has not made any illegal parameter settings for the protocol. If so, `EOS_ILLPRM` can be returned. This profile should be copied into the `pp_profile` structure in the per path storage by the driver.

If the buffer field is `NULL` and the param field is non-NULL, the user has passed in a valid `conn_type` pointer. The `conn_svc_type` field in the `conn_type` structure should be used by the driver to set the new profile for the path. For instance, when a path opens and the default profile is a voice call, this profile is copied into the per path storage. If a set profile call is made with a `conn_type` structure and the `conn_svc_type` field is `ITE_SVC_DATA_ANY`, the driver should copy the `ITE_SVC_DATA_ANY` profile from the logical unit array into the per path storage profile structure.

If the buffer field is non-NULL, then it points to an `xxx_profile` structure that has probably been modified by the application. The driver should do some integrity checking on the modifications, then change the profile structure in the per path storage to the contents of the passed in buffer.

Once copied, this profile becomes valid for the path. Notice the only way to set profiles in the logical unit array is to make a change to the descriptor.

Attributes

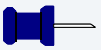
Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`conn`	points to the caller-allocated `conn_type` structure set up to represent the connection type information for your device.
`pr_size`	points to the number of bytes to read in the `pr_buffer`.
`pr_buffer`	points to the caller-allocated buffer in which the path profile is placed.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.
`EOS_ILLPATH`	returned when `conn_type` structure is `NULL`, or `conn_svc_type` in `conn_type` structure is invalid.



Note

Other errors may be returned by SoftStax drivers.

See Also

`ite_path_profileget()`

ite_path_push()

Pushes Protocol or Hardware-Driver Onto Path

Syntax

```
#include <SPF/item.h>
error_code ite_path_push(
   path_id   path,
   char      *dev_name);
```

Libraries

`item.l`

Description

`ite_path_push()` pushes a protocol or hardware driver onto the path.

For More Information

For information about how SoftStax stacks and unstacks protocols, refer to ***Using SoftStax*** and the ***SoftStax Porting Guide***.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc. This path may already have other protocol drivers stacked on it.

`dev_name`	contains the ITEM device name string referencing the ITEM device to be stacked on this path.

Indirect Errors

`EOS_MNF`	returns when the module referenced by `dev_name` is not currently in memory.
`EOS_STKFULL`	returned if there are already six protocols stacked on the path being pushed.
`EOS_BUSERR`	returned if `dev_name` is `NULL`.
`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`ite_path_pop()`

Chapter 2: Conv and OS Libraries

This document contains descriptions, in alphabetical order, of the telecommunications Application Programming Interface (API) functions.

Function Descriptions

The function descriptions in this manual are, for the most part, self-explanatory. Each section of a function description is defined below.

The **Syntax** section shows the function prototype with the required parameters and their data types. This section also lists the files that you need to include when using the function and coding dependencies.

The **Libraries** section lists the name of the library in which you can find the function.

The **OS-9 Attributes** section lists various attributes of each function in relation to OS-9—including whether the function is compatible with OS-9 and/or OS-9 for 68K; whether the function is in user state and/or system state; and whether the function is safe for use in a threaded application.

The **Description** section provides a description of the function.

The **Parameters** section provides details about each of the parameters.

Direct Errors are errors that are detected within the library call and are a direct result of that particular call.

Indirect Errors are the result of invalid parameter values passed to and detected by another function call. They are not directly returned by the original calling function.

The **See Also** section tells you about related functions or materials providing more information about the function.

_os_getstat()

Getstat

Syntax

```
#include <sg_codes.h>
#include <types.h>
#include <SPF/spf.h>
error_code _os_getstat(
   path_id   path,
   u_int32   code,
   void      *pb);
```

Libraries

```
conv_lib.l (OS-9 for 68K)
os_lib.l (OS-9)
```

Description

`_os_getstat()` is a wildcard call used to get individual device parameters that are not uniform on all devices or that are highly hardware dependent.

Note

The Ultra C library says this call is only available for OS-9. However, this call is available for OS-9 for 68K by using the `conv_lib.l` library.

For More Information

Refer to ***Using SoftStax*** for an example of using the `os_lib.l` library to create your own getstat calls.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the input/output (I/O) path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`code`	contains the getstat code.
`pb`	points to the getstat `spf_ss_pb` parameter block.

For More Information

Refer to ***Using SoftStax*** for a description of `spf_ss_pb`.

Indirect Errors

`EOS_BPNUM`	returned when the path number is invalid.
`EOS_PTHLOST`	returned when the path is lost and no longer valid.
`EOS_PPS_NOTFND`	returned when the driver cannot find its local path storage for the `path_id` passed in.

See Also

`_os_setstat()`

_os_setstat()

Setstat

Syntax

```
#include <sg_codes.h>
#include <types.h>
#include <SPF/spf.h>
error_code _os_setstat(
   path_id   path,
   u_int32   code,
   void      *pb);
```

Libraries

```
conv_lib.l (OS-9 for 68K)
os_lib.l (OS-9)
```

Description

`_os_setstat()` is a wildcard call used to set individual device parameters that are not uniform on all devices or are highly hardware dependent.

Note

The Ultra C library says this call is only available for OS-9. However, this call is available for OS-9 for 68K by using the `conv_lib.l` library.

For More Information

Refer to ***Using SoftStax*** for an example of using the `os_lib.l` library to create your own setstat calls.

Attributes

Operating System:	OS-9 and OS-9 for 68K
State:	User
Threads:	Safe

Parameters

`path`	contains a handle identifying the I/O path. This handle is usually obtained from calls such as `ite_path_open()`, `_os_open()`, `socket()`, etc.
`code`	contains the setstat code.
`pb`	points to the setstat parameter block.

For More Information

Refer to ***Using SoftStax*** for a description of `spf_ss_pb`.

See Also

`_os_getstat()`

Index

A

B

C

D

E

G

H

I

L

M

N

O

P

R

S

W

Product Discrepancy Report

To: Microware Customer Support

FAX: 515-224-1352

From:_____

Company:_____

Phone:_____

Fax:_____Email:_____

Product Name:

Description of Problem:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Host Platform_____

Target Platform_____

RadiSys.

MICROWARE SOFTWARE