NAME

errors – errors returned by Shore Value-Added Server

SYNOPSIS

```
typedef int  VASResult;

enum TxStatus { Stale, Active, Prepared, Aborting,
                        Committing, Ended, NoTx };

enum error_type { ET_USER=0x1, ET_VAS=0x2, ET_FATAL=0x4000 };

struct Status {
    int         vasresult;
    int         vasreason;
    int         smresult;
    int         smreason;
    int         unixreason;
    TxStatus    txstate;
};

void            shore_vas::perr(
                    const char *message,
                    int         line = -1, // not printed if <0
                    const char *filename=0,// not printed if null
                              error_type  ekind = ET_VAS // not printed if
                                    // ekind == ET_USER and !this->printusererro
                ) const;

void            shore_vas::perr(
                    ostream &out,
                    const char *message,
                    int         line = -1, // not printed if <0
                    const char *filename=0,// not printed if null
                              error_type  ekind = ET_VAS // not printed if
                                    // ekind == ET_USER and !this->printusererro
                ) const;

static const char*  p_status(unsigned int x);
static bool         unp_status(const char *msg, unsigned int *res) ;
```

DESCRIPTION

Each function (method) in the Shore Value-Added Server class returns either SVAS_OK (zero), SVAS_FAILURE (non-zero), or SVAS_WARNING (non-zero). In general, return values of functions (through output arguments) are not valid unless the function returns SVAS_OK. If the function returns SVAS_FAILURE or SVAS_WARNING, the *status* member indicates the reason for the failure. The reason can be a condition discovered by the SVAS, or it might have been a failure discovered by a lower level, such as the operating system or the Storage Manager.

If the condition is discovered by the SVAS, *status.vasresult* is SVAS_FAILURE or SVAS_WARNING, *status.vasreason* has a value from the list below, and the fields *status.smresult, status.smreason*, *status.osreason* have the value zero.

If the condition is discovered by the Storage Manager, *status.vasresult* is SVAS_FAILURE or SVAS_WARNING, *status.vasreason* has the value SVAS_SmFailure, and the fields *status.smresult, status.smreason, status.osreason* have nonzero values, which indicate the reason for the problem.

The field *status.osreason* takes its values from the list found in `<errno.h>`.

The functions **perr(...)** can be called to print a message that interprets the SVAS's *status* member. The *message* argument is any string that the caller wishes to be printed along with the interpretive messages (similar to the C library's **perror()**).

The function **p_status** interprets its argument as a candidate for *status.vasreason,* and if the value is one of those that the SVAS can return in the circumstance of an error, **p_status** returns the string name of the error condition.

The function **unp_status** converts a string name to an integer error code. If the string is associated with an error condition, the integer error code is returned in **res,* and the Boolean value TRUE is returned. If the string is not associated with any error condition, the Boolean value FALSE is returned and **res,* is unchanged.

```
SVAS_WARNING
SVAS_FAILURE
SVAS_OK
SVAS_NotImplemented - function or feature isn't implemented yet
SVAS_TxNotAllowed - this method cannot be called inside a transaction
SVAS_TxRequired - this method must be called inside a transaction
SVAS_Missing - object with the given OID does not exist
SVAS_Already - function has already been completed (e.g., format)
SVAS_CantFormat - volume cannot be formatted
SVAS_SmFailure - Storage Manager discovered an error
SVAS_RpcFailure - network failure, server crashed
SVAS_WrongObjectType - tried to use an object as if it were
    a type other than what it is
SVAS_NotFound - could not find object with the given path name
SVAS_BadParam1 - problem with first argument to method
SVAS_BadParam2 - problem with second argument to method
SVAS_BadParam3 - problem with third argument to method
SVAS_BadParam4 - problem with fourth argument to method
SVAS_BadParam5 - problem with fifth argument to method
SVAS_BadParam6 - problem with sixth argument to method
SVAS_BadParam7 - problem with seventh argument to method
SVAS_BadParam8 - problem with eighth argument to method
SVAS_BadParam9 - problem with ninth argument to method
SVAS_InUse - volume cannot be unmounted because it is in use
    by some user
SVAS_TxNotActive - the current transaction was not found to be
    in the active state, as expected.
SVAS_MallocFailure - internal error: server could not malloc space
SVAS_BadRange - the given range of bytes is inappropriate for the
    object being read or written
SVAS_IntegrityBreach - the integrity of the type system has been compromised,
    or would be compromised if this were to completed and the
    transaction were to commit
SVAS_IndexScanIsOpen - this function cannot be invoked during an index scan
SVAS_PoolScanIsOpen - this function cannot be invoked during a pool scan
SVAS_ScanNotInProgress - a scan must be opened before this function can
    be invoked
```

```
SVAS_CantChangeCoreSize - an attempt to truncate an object would have
    removed part of the object’s core.
SVAS_NotAPool - this function applies to a pool, and the object
    given is not a pool
SVAS_BadPathSyntax - the string given does not have proper syntax
    for a pathname
SVAS_BadFileNameSyntax - the string given does not have proper
    syntax for a file name
SVAS_IsAnonymous - the object referenced is anonymous; the function
    applies to registered objects
SVAS_VolumesDontMatch - an attempt to create an object with a given,
    preallocated OID failed because the OID was created on a volume
    different from the volume on which the pool resides.
SVAS_XdrError - the object could not be byte-swapped
SVAS_ShmError - the SVAS could not allocate or attach shared memory
SVAS_InternalError - unspecified internal error
SVAS_BadType - an attempt to operate on the object’s type object
    failed, possibly because the type object’s OID is bad
SVAS_AuthenticationFailure - could not authenticate the client to the server
SVAS_UnixFailure - a failure was returned from a Unix library function,
    such as getpwuid()
SVAS_BadSerial - this is an internal error
SVAS_UserAbort - the reason for aborting the transaction is that
    the user (caller, client, application) requested it
SVAS_NotMounted -  the directory requested is not on a mounted file system
SVAS_NotADirectory - this function applies to directories; the object
    named is not a directory
SVAS_IsADirectory - this function does not apply to directories; the object
    named is a directory
SVAS_NotEmpty - an attempt to remove something failed because the
    object was not empty (e.g., a directory)
SVAS_NotOwner - you have to be the owner of the object to do this
SVAS_PermissionDenied - you don’t have the necessary permission
    (read/write/execute) to do this
SVAS_ReadOnlyFS - an attempt to update an object failed because the
    object resides on a read-only file system
SVAS_TooManySymlinks - while expanding symbolic links during path
    -resolution, too many symbolic links were encountered (possibly a loop)
SVAS_TooManyLinks - an attempt to create a hard link to an object failed
    - because the maximum number of links per object was reached
SVAS_PathTooLong - after expansion of symbolic links, the pathname
    exceeds the maximum length permitted
SVAS_CrossDeviceRef - you cannot make cross-references  or links across
    devices (volumes, file system)
SVAS_AlreadyExists - an attempt to create an object with a pre-allocated
    OID failed because there is already an object with that OID
SVAS_BadAddress - the caller gave an invalid address for an argument
```

VERSION

This manual page applies to Version 1.1 of the Shore software.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518.

COPYRIGHT

SEE ALSO

transaction(svas).