NAME

errors – error codes and messages for the Shore language-independent library

SYNOPSIS

```
#include <ShoreApp.h>
class Shore;
```

DESCRIPTION

Errors in Shore are indicated by an unsigned integer wrapped in a mechanism that includes stack traces and other debugging aids. The mechanism is the class **shrc,** which is the return-type for most Shore and and object cache methods.

The domain of error codes is an extension of the Unix error codes found in `#include <errno.h>`. Each layer of the Shore software adds its own extension to the domain. The Shore object cache compresses all the extensions into the set of error codes found in `#include <errno.h>`, extended by the list below. The left column shows **#define -ed** constants; the right column is a brief description of the meaning of the error.

Manifest constant	Meaning
SH_AlreadyInitialized	Object cache already initialized
SH_NotInitialized	Object cache not initialized
SH_OptionsNotInitialized	Process_options or default_options not called yet
SH_BadObject	Invalid object
SH_BadVolume	Invalid volume
SH_BadPool	Invalid pool
SH_BadType	Invalid type
SH_BadLockMode	Invalid lock mode
SH_NotAPool	Object is not a pool
SH_NotADir	Object is not a directory
SH_ScanAlreadyOpen	Scan is already open
SH_ScanNotOpen	Scan is not open
SH_EndOfScan	Reached EOF in scan
SH_ScanBufTooSmall	Buffer provided by user is too small
SH_OutOfMemory	Out of memory
SH_TxNotAllowed	This operation cannot be performed in a transaction
SH_TxRequired	This operation must be performed in a transaction
SH_TxAborted	The transaction was aborted
SH_NotFound	The named object was not found
SH_Already	The requested operation was already performed
SH_ROFailure	Failure in remote operation
SH_Internal	Object cache internal error
SH_NoIndex	Index operation attempted on object with no indexes
SH_NoSuchIndex	Index operation attempted on non-index
SH_DuplicateEntry	Operation on unique index would cause duplicate entries

To create a return code representing the error code X, use the macro `RC(X)` as follows:

```
        return RC(EINVAL);
        // or
        return RC(SH_TxAborted);
```

To print an error code and its stack trace, simply print the return code. You can print the descriptive string, without the entire stack trace, with the method

```
    static int Shore::perror(shrc &)
```

and you can print the descriptive string for an integer with the method

```
    static int Shore::perror(int)
```

For example:

```
        shrc rc;
        rc = ...

        // print stack trace, etc
        if(rc) {
              cerr << rc << endl;
        }

        // print error string only
        if(rc) {
              cerr << Shore::perror(rc) << endl;
        }
```

Lower layers can return other error codes, too numerous to mention. They are mostly self-describing, when printed. When any lower layers return an error, the object cache converts the error code to an appropriate one of the above error codes. If there is no obvious good conversion, the object cache pushes SH_ROFailure on the top of the error code's stack, leaving the lower layer's error code and stack trace in place.

This means that you can write programs that check for specific errors, and the only set of values you need to check is the set of Unix error codes (found in `<errno.h>` and the above set of `SH_*` values. You check for specific values by invoking the macro `RC(),` which accepts an integer and returns an *shrc*:

```
        shrc rc;
        rc =  ...

        if( rc == RC(EINVAL) || rc == RC(SH_Already) ) {
              ...
        }
```

ERROR HANDLER

If the object cache encounters an error while following a reference (*Ref<T>*), it calls a error-handling function. There is a default function, which you can override.

```
    void
    my_error_handler(shrc rc)
    {
        cerr << "ERROR IN DEREFERENCE  : " << rc << endl;
        cerr << "To debug this, put a breakpoint in my_error_handler()"
        <<endl;
    }
    //
    // replace the default error-handler with my function
    //
    (void) Shore::set_error_handler(my_error_handler);
```

The default handler,

```
    void default_error_handler(shrc rc)
```

simply prints *rc*.

BUGS

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

intro(oc), init(oc), process_options(oc).