HoloStor Interface Specification 1.0a

May 15, 2011

This document provides a specification of exported data structures and functions for the HoloStor library. It assumes that the reader has a good understanding of HoloStor nomenclature and operation.

Data Structures

```
typedef struct _HOLOSTOR_CFG {
  unsigned int    BlockSize;  // Block size in bytes (2**N and >= 64)
  unsigned int    DataBlocks; // Data blocks per reliability group
  unsigned int    EccBlocks;  // Redundancy blocks per " group (1-4)
} HOLOSTOR_CFG;
```

Notes on configuration constrains:

1. Block size must be a 2**N value and be greater then or equal to 64.
2. The number of ECC blocks must be between 1 and 4.
3. The number of blocks in a reliability group (Data plus ECC) may not exceed 17.
4. The total physical nodes in a cluster must be greater then or equal to the number of blocks per reliability group; so that all blocks can be dispersed to different nodes.

Functions

Return Values

All functions return 0 or a positive value on success and a negative value on failure. The failure codes include:

```
#define HOLOSTOR_STATUS_SUCCESS                 (0)
#define HOLOSTOR_STATUS_INVALID_PARAMETER       (-1)
#define HOLOSTOR_STATUS_BAD_CONFIGURATION       (-2)
#define HOLOSTOR_STATUS_NO_MEMORY               (-3)
#define HOLOSTOR_STATUS_TOO_MANY_BAD_BLOCKS     (-4)
#define HOLOSTOR_STATUS_BAD_SESSION             (-5)
#define HOLOSTOR_STATUS_MISALIGNED_BUFFER       (-6)
#define HOLOSTOR_STATUS_TOO_MANY_SESSIONS       (-7)
```

Setup Functions

Create a session for the specified configuration. Based on the configuration, memory for various tables is allocated and values pre-calculated for subsequent use. These functions make calls to underlying "*malloc*" and "*free*" functions provide by the user. Multiple sessions (presumably with different configurations) can simultaneously be in use.

```
typedef int HOLOSTOR_SESSION;

HOLOSTOR_SESSION
HoloStor_CreateSession(
  IN const HOLOSTOR_CFG*      lpConfiguration
  );
```

Free memory resources that have been allocated for the session.

```
int
HoloStor_CloseSession(
  IN HOLOSTOR_SESSION   hSession
  );
```

Basic Encoding and Decoding Functions

These functions perform the basic encode, decode and rebuild operations on data in block buffers. All block buffers are allocated by the caller, are 16-byte aligned (ideally cache line size aligned) and of size "*BlockSize*".

Most functions actually operate on a reliability group consisting of "*DataBlocks*" block buffers plus "*EccBlocks*" block buffers. These buffers are grouped into an array of pointers to blocks, the "*BlockGroup"*, where the first "*DataBlocks*" entries point to ordered data blocks and the next "*EccBlocks*" entries point to ordered ECC blocks. The order of blocks in a reliability group is important and each is identified by its index value in the array, starting at 0. This same block order must be used on all calls to these functions.

The "*ValidBlockMask*" argument indicates which buffers in the group have valid data, where bit 0 refers to index value 0 and when set indicates that the first buffer in the array contains valid data. Buffers for any bits that are clear do not contain valid data, but such buffers are still present and will be used to return data (as needed).

```
int
HoloStor_Encode(
  IN HOLOSTOR_SESSION   hSession,
  IN OUT void**   lpBlockGroup      // IN Data; OUT all ECC
  );

int
HoloStor_Decode(
  IN HOLOSTOR_SESSION   hSession,
  IN OUT void**   lpBlockGroup,     // IN Data/ECC; OUT missing data
  IN unsigned int uInvalidBlockMask // Mask of buffers with no data
  );

int
HoloStor_Rebuild(
  IN HOLOSTOR_SESSION   hSession,
  IN OUT void**   lpBlockGroup,     // IN Data/ECC; OUT as specified
  IN unsigned int uInvalidBlockMask,// Mask of buffers with no data
  IN int          lWhichBlock       // Block index to rebuild (-1 all)
  );
```

Small Write Encoding Functions

These functions complement the basic encoding functions to provide optimized methods to write blocks within a reliability group, without the need to operate on unmodified data blocks.

```
int
HoloStor_WriteDelta(
  IN HOLOSTOR_SESSION   hSession,
  IN const void*  lpDataBlockOld,   // Data block before updating
  IN const void*  lpDataBlockNew,   // New contents of the data block
  OUT void*       lpDeltaBlock      // Delta to forward to ECC nodes
  );

int
HoloStor_EncodeDelta(
  IN HOLOSTOR_SESSION   hSession,
  IN OUT void**   lpDeltaBlockGroup,// IN forwarded delta blocks
  IN unsigned int uInvalidBlockMask,// Mask of buffers with no data
  IN unsigned int lEccIndex,        // ECC block index being updated
  IN const void*  lpEccBlockOld,    // Old ECC block
  OUT void*       lpEccBlockNew     // Returned new ECC block
  );
```