LSS Interface

```
// An input stream is closed when the interface ptr is deleted
struct IInputStream
{
    virtual ~IInputStream() {}

    // Try to read numBytesRequested from the stream.  Returns the actual number of bytes read
    // which may be less than numBytesRequested
    virtual int ReadStream(void* buffer, int numBytesRequested) = 0;
};

// An output stream is closed when the interface ptr is deleted
struct IOutputStream
{
    virtual ~IOutputStream() {}
    virtual void WriteStream(const void* buffer, int numBytes) = 0;
    virtual void FlushStream() = 0;
};

typedef DWORD OID_HIGH;
typedef DWORD OID_LOW;

struct OID
{
    OID_LOW m_low;
    OID_HIGH m_high;
};

struct ILSS
{
    virtual ~ILSS() {}
    virtual void Close() = 0;         // The LSS *must* be explicitly closed
    virtual void Flush() = 0;         // Blocks until all data has been flushed to disk

    virtual OID_HIGH CreateOidSpace() = 0;

    // Retrieve all the OIDs (actually only the low 32 bit part of each OID) in the
    // OID space associated with the given OID_HIGH.
    virtual void GetOidsInOidSpace(std::vector<OID_LOW>& oidLows, OID_HIGH osid) = 0;

    virtual OID AllocateOID(OID_HIGH osid) = 0;

    // Does an object with the given OID exist?
    virtual bool SerialElementExists(OID oid) const = 0;

    // Provides a stream for reading the serial element with the given OID.
    // The returned stream must be deleted to avoid a memory leak
    // Returns NULL if no serial element exists with the given OID
    virtual /*gives*/ IInputStream* ReadSerialElement(OID oid) = 0;

    /////////////////////////// Commit phase ////////////////////////////

    // All changes (ie mutative work) done on an LSS (apart from OID allocations) must
    // be done by a thread that has opened a commit phase.
    // A commit phase is intended for a single thread.  Only the thread that called
    // BeginCommitPhase() on the LSS is permitted to call DeleteSerialElement(),
    // WriteSerialElement() or EndCommitPhase().
    // It is an error to close or delete the LSS while there is an open commit phase.
    virtual void BeginCommitPhase() = 0;
    virtual void EndCommitPhase() = 0;

    // Returns a stream that can be used to write the serial element with the given
    // OID.  If the serial element already exists then the previous rendition will be
    // replaced by a new one.
    // The returned stream must be deleted to avoid a memory leak.
    // Must only be called by a thread that has begun a commit phase
    virtual /*gives*/ IOutputStream* WriteSerialElement(OID oid) = 0;

    // Permanently delete the serial element with the given OID
    // Must only be called by a thread that has begun a commit phase
    virtual void DeleteSerialElement(OID oid) = 0;

    // Delete the OID space associated with the given OSID.  The OID space must be
    // empty.  Must only be called by a thread that has begun a commit phase
    virtual void DeleteOidSpace(OID_HIGH oidHigh) = 0;
};


ILSS* gfnCreateLSS(
    const String& path,
    bool* alreadyExists,
    int segmentSize = LSS_USE_DEFAULT_SEGMENT_SIZE);
```