LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Topics

Here is a list of all topics with brief descriptions:

[detail level 1 2]

General Functions	General convenience functions
Application Functions	Functions that all SDK Applications will need to use
Licensing Functions	
Metadata Functions	Functions for reading, writing, and parsing metadata objects
▼ Reader Functions	
Basic Reader Functions	
Project Functions	
Setup Functions	
Target Functions	
Run Functions	
GeoTag Functions	
SiteMap Functions	
Asset Functions	
Model and Model Node Functions	
Point Cloud Functions	
User Coordinate System Functions	

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

General Functions

General convenience functions. More...

Functions

const wchar_t *	**LgsxUtil_GetCurrentTimeString** () Convenience function: Return current local time formatted as "HHMMSS".
const wchar_t *	**LgsxUtil_GetCurrentDateString** () Convenience function: Return local date formatted as "YYMMDD".
void *	**LgsxUtil_TimeStart** () Create a new timer and starts it.
double	**LgsxUtil_TimeGetLapse** (void *pTime) Returns elapsed time for the given timer in seconds.
const wchar_t *	**LgsxUtil_TimeGetLapseStr** (void *pTime, int format) Returns elapsed time for the given timer in seconds as a string.
int	**LgsxUtil_TimeEnd** (void **pTime) Frees the timer object.

int	**LgsxUtil_Sleep** (int millisec) Suspends execution for the given number of milliseconds.
void *	**LgsxUtil_AllocMem** (int size) Allocate a block of memory with the given size and return a pointer to it.
void	**LgsxUtil_FreeMem** (void **ptr) Free a block of memory previously created by the SDK.
int	**LgsxUtil_A2W** (const char *value, wchar_t *tValue) Convert UTF8 encoded text to UTF16.
int	**LgsxUtil_W2A** (const wchar_t *value, char *sValue) Convert UTF16 encoded text to UTF8.
void	**Lgsx_FreeHandle** (void *handle) Frees the SDK object referenced by the given handle.
const wchar_t *	**Lgsx_GetLastError** () Returns error string for most recent SDK error.
int	**Lgsx_GetLastErrorNo** () Returns error number for most recent SDK error.
int	**Lgsx_GetProductVersion** (wchar_t *version, long *buildNum) Get LGSx SDK application's version and build number.

int	**Lgsx_InitProductVersion** (const wchar_t *customVersion) Initialize LGSx SDK application's version string.
int	**Lgsx_InitProductVersionEx** (const wchar_t *customVersion, const wchar_t *fileVersion) Initialize LGSx SDK application's version string and file version.
int	**Lgsx_InitProductName** (const wchar_t *prodName) Define the name of the product you want to display in About dialog.
int	**Lgsx_InitProductBuildNumber** (int build) Initialize LGSx SDK application's build number.

Detailed Description

General convenience functions.

Methods in this class are actually global C APIs.

Function Documentation

◆ Lgsx_FreeHandle()

void Lgsx_FreeHandle (void * handle)

Frees the SDK object referenced by the given handle.

Parameters

handle [In] Handle to free.

◆ Lgsx_GetLastError()

const wchar_t * Lgsx_GetLastError ( )

Returns error string for most recent SDK error.

Returns

String containing error message. Caller does **not** need to free string memory.

◆ Lgsx_GetLastErrorNo()

int Lgsx_GetLastErrorNo ( )

Returns error number for most recent SDK error.

Returns
Error number.

Lgsx_GetProductVersion()

int Lgsx_GetProductVersion (wchar_t * version,
long * buildNum
)

Get LGSx SDK application's version and build number.

The version string and build number are displayed in About dialog.

Parameters

version [Out] Returned version string. Need to pre-allocate at least 40 characters.

buildNum [Out] Returned build number.

Returns
0 for success.

Lgsx_InitProductBuildNumber()

int Lgsx_InitProductBuildNumber (int build)

Initialize LGSx SDK application's build number.

The version string and build number are displayed in About dialog.

Parameters

build Build number.

Returns
0 for success.

Lgsx_InitProductName()

int Lgsx_InitProductName (const wchar_t * prodName)

Define the name of the product you want to display in About dialog.

Returns
0 for success.

Lgsx_InitProductVersion()

int Lgsx_InitProductVersion (const wchar_t * customVersion)

Initialize LGSx SDK application's version string.

The version string and build number are displayed in About dialog.

Parameters
customVersion [In] Custom version string, such as "2.1A".

Returns
0 for success.

Lgsx_InitProductVersionEx()

```
int Lgsx_InitProductVersionEx ( const wchar_t *  customVersion,
                                const wchar_t *  fileVersion
                              )
```

Initialize LGSx SDK application's version string and file version.

The version string and build number are displayed in About dialog.

Parameters

customVersion [In] Custom version string, such as "2.1A".

fileVersion [In] Version string to be displayed in module/dll.

Returns

0 for success.

LgsxUtil_A2W()

```
int LgsxUtil_A2W ( const char *  value,
                   wchar_t *     tValue
                 )
```

Convert UTF8 encoded text to UTF16.

Parameters

value [In] Source string.

tValue [out] Converted string.

Returns

0 for success.

LgsxUtil_AllocMem()

void * LgsxUtil_AllocMem (int size)

Allocate a block of memory with the given size and return a pointer to it.

Parameters

size [In] Size of memory to create, in bytes.

Returns

Pointer to allocated memory.

LgsxUtil_FreeMem()

void LgsxUtil_FreeMem (void ** ptr)

Free a block of memory previously created by the SDK.

Parameters

ptr [In] Pointer to memory to be freed.

LgsxUtil_GetCurrentDateString()

const wchar_t * LgsxUtil_GetCurrentDateString ( )

Convenience function: Return local date formatted as "YYMMDD".

Range of years is 0 to 99. Range of months is 1 to 12. Range of days is 1 to 31.

Returns
String pointer to formatted date. String memory is managed by the SDK.

LgsxUtil_GetCurrentTimeString()

const wchar_t * LgsxUtil_GetCurrentTimeString ( )

Convenience function: Return current local time formatted as "HHMMSS".

Hours are in 24 hour format (0 to 23).

Returns
String pointer to formatted local time. String memory is managed by the SDK.

LgsxUtil_Sleep()

int LgsxUtil_Sleep (int millisec)

Suspends execution for the given number of milliseconds.

Parameters
millisec [In] Number of milliseconds to sleep.

Returns
0 for success.

LgsxUtil_TimeEnd()

int LgsxUtil_TimeEnd (void ** pTime)

Frees the timer object.

Parameters

pTime [In] Timer object handle.

Returns

0 for success.

LgsxUtil_TimeGetLapse()

double LgsxUtil_TimeGetLapse (void * pTime)

Returns elapsed time for the given timer in seconds.

Parameters

pTime [In] Timer object handle.

Returns

Elapsed time in seconds.

LgsxUtil_TimeGetLapseStr()

const wchar_t * LgsxUtil_TimeGetLapseStr (void * pTime,
int format
)

Returns elapsed time for the given timer in seconds as a string.

Parameters

pTime [In] Timer object handle.

format [In] String format to return. 0 returns time string in seconds. 1 returns time as **hh:mm:ss**, where **hh** is two-digit hours, **mm** is two-digit minutes, and **ss** is seconds..

Returns

Buffer containing time string. Buffer must be freed by the caller.

LgsxUtil_TimeStart()

```
void * LgsxUtil_TimeStart ( )
```

Create a new timer and starts it.

Returns

Handle for the new Timer object.

LgsxUtil_W2A()

```
int LgsxUtil_W2A ( const wchar_t *  value,
                   char *           sValue
                 )
```

Convert UTF16 encoded text to UTF8.

Parameters

value [In] Source string.

sValue [out] Converted string.

Returns

0 for success.

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Application Functions

Functions that all SDK Applications will need to use. More...

Functions

int **Lgsx_Initialize** (const wchar_t *pTempPath, bool enableLog)
Initialize the LGSx SDK APIs.

int **Lgsx_Uninitialize** ()
Unintialize the connection from Cyclone service.

Detailed Description

Functions that all SDK Applications will need to use.

Function Documentation

◆ Lgsx_Initialize()

```
int Lgsx_Initialize ( const wchar_t *  pTempPath,
                      bool             enableLog
                    )
```

Initialize the LGSx SDK APIs.

This should be the first call before checking license or loading or importing a project.

Parameters

pTempPath	[In] Path to folder to be used for temporary files used by the SDK.
enableLog	[In] Set to TRUE to enable the SDK's logging/log file. Logging is disable if set to FALSE.

Returns

0 for success.

◆ Lgsx_Uninitialize()

```
int Lgsx_Uninitialize ( )
```

Unintialize the connection from Cyclone service.

This should always be called in order to cut connection from service. It is not recommended to call the pair of Lgsx_Initialize and Lgsx_Uninitialize multiple times during a session.

Returns

0 for success.

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Licensing Functions

Functions

int **Lgsx_InitLicense** (const wchar_t *lic, const wchar_t *ver)
Initialize the license to be loaded.

int **Lgsx_InitLicenseEx** (const wchar_t *lic, const wchar_t *ver, const wchar_t *product)
Initialize the license to be loaded.

int **Lgsx_InitLicenseEx2** (const wchar_t *lic, const wchar_t *ult, const wchar_t *ver, const wchar_t *product)
Initialize the license to be loaded.

int **Lgsx_IsLicensed** ()
Check if the license is valid.

int **Lgsx_LicenseDaysLeft** ()
Check the number of days left for the license.

int **Lgsx_LoadLicense** ()
Load license and hold the usage.

int **Lgsx_UnloadLicense** ()
Unload license.

int **Lgsx_IsLicensedExA** (const char *lic, const char *ver)

Check if a named license is valid.

int **Lgsx_LicenseDaysLeftExA** (const char *lic, const char *ver)
Check the number of days left for given license.

int **Lgsx_LoadLicenseExA** (const char *lic, const char *ver)
Load given license and hold usage until UnloadLicense.

int **Lgsx_UnloadLicenseExA** (const char *lic)
Unload a named license.

Detailed Description

Function Documentation

◆ Lgsx_InitLicense()

```
int Lgsx_InitLicense ( const wchar_t *  lic,
                       const wchar_t *  ver
                     )
```

Initialize the license to be loaded.

This must be called before opening a LGSx file or other functions that require an SDK license. Opening LGSx file for reading is free when SDK license is valid.

Following functions will only be unlocked when a license check is successful. If any of these functions involves opening a JetStream point cloud, it will also check if JetSteam licensing is enabled. A special key containing the support of JS module is needed to unlock these APIs to support JetStream.

Lgsx_LoadProject
Lgsx_ImportModelSpaceView
Lgsx_AddModelSpaceView

Parameters

lic [In] The name string of the license.

ver [In] The version string of the license.

Returns
0 for success.

Lgsx_InitLicenseEx()

```
int Lgsx_InitLicenseEx ( const wchar_t *  lic,
                         const wchar_t *  ver,
                         const wchar_t *  product
                       )
```

Initialize the license to be loaded.

This must be called before opening a LGSx file or other functions that require a license. This is an extended version of **Lgsx_InitLicense()**.

Parameters

lic	[In] The name string of the license.
ver	[In] The version string of the license.
product	[In] The product name.

Returns
0 for success.

Lgsx_InitLicenseEx2()

```
int Lgsx_InitLicenseEx2 ( const wchar_t *  lic,
                          const wchar_t *  ult,
                          const wchar_t *  ver,
                          const wchar_t *  product
                        )
```

Initialize the license to be loaded.

This must be called before opening a LGSx File or other functions that require a license. This is an extended version of **Lgsx_InitLicense()**.

Parameters

lic	[In] The name string of the license.
ult	[In] The name string of ultimate license (or empty to not support ultimate).
ver	[In] The version string of the license.
product	[In] The product name.

Returns

0 for success.

◆ Lgsx_IsLicensed()

```
int Lgsx_IsLicensed ( )
```

Check if the license is valid.

Returns

1 means it is licensed.

◆ Lgsx_IsLicensedExA()

```
int Lgsx_IsLicensedExA ( const char *  lic,
                         const char *  ver
                       )
```

Check if a named license is valid.

Parameters

lic [In] The name string of the license.

ver [In] The version string of the license.

Returns

1 means it is licensed.

Lgsx_LicenseDaysLeft()

```
int Lgsx_LicenseDaysLeft ( )
```

Check the number of days left for the license.

Returns

The number of days.

Lgsx_LicenseDaysLeftExA()

```
int Lgsx_LicenseDaysLeftExA ( const char *  lic,
                              const char *  ver
                            )
```

Check the number of days left for given license.

Parameters

lic [In] The name string of the license.

ver [In] The version string of the license.

Returns

The number of days.

Lgsx_LoadLicense()

int Lgsx_LoadLicense ( )

Load license and hold the usage.

Returns
0 for success.

Lgsx_LoadLicenseExA()

int Lgsx_LoadLicenseExA (const char * lic,
const char * ver
)

Load given license and hold usage until UnloadLicense.

Parameters
lic [In] The name string of the license.
ver [In] The version string of the license.

Returns
0 for success.

Lgsx_UnloadLicense()

int Lgsx_UnloadLicense ( )

Unload license.

Returns
0 for success.

◆ Lgsx_UnloadLicenseExA()

int Lgsx_UnloadLicenseExA (const char * lic)

Unload a named license.

Parameters
lic [In] The name string of the license.

Returns
0 for success.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Metadata Functions

Functions for reading, writing, and parsing metadata objects.
More...

Functions

LgsxMetadataPtr	**Lgsx_MetaCreateInstance** () Creates an instance of metadata container.
LgsxMetadataPtr	**Lgsx_MetaClone** (LgsxMetadataPtr pMetadata) Clones a metadata container.
int	**Lgsx_MetaRemove** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName) Remove a key pair from metadata.
int	**Lgsx_MetaHas** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName) TRUE if the key exists in the metadata.
int	**Lgsx_MetaReset** (LgsxMetadataPtr pMetadata) Resets traverse pointer in the metadata.
int	**Lgsx_MetaMoveNext** (LgsxMetadataPtr pMetadata, LgsxPropertyName pName, int *pType)

Move to next key pair of the metadata.

int **Lgsx_MetaGetBool** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, bool *pValue)
Get the value associated with the given name in metadata.

int **Lgsx_MetaGetInt** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, int *pValue)
Get the value associated with the given name in metadata.

int **Lgsx_MetaGetInt64** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, __int64 *pValue)
Get the value associated with the given name in metadata.

int **Lgsx_MetaGetDouble** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, double *pValue)
Get the value associated with the given name in metadata.

int **Lgsx_MetaGetString** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, wchar_t *pValue, int maxLength)
Get the value associated with the given name in metadata.

int **Lgsx_MetaGetDouble3** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, double *pValue)
Get the value associated with the given name in metadata.

int **Lgsx_MetaGetDouble4** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, double *pValue)
Get the value associated with the given name in metadata.

int **Lgsx_MetaGetMetadata** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, LgsxMetadataPtr *pMetadataOut)
Get the value associated with the given name in metadata.

int **Lgsx_MetaAddBool** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, bool value)
Add name value pair to metadata.

int **Lgsx_MetaAddInt** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, int value)
Add name value pair to metadata.

int **Lgsx_MetaAddInt64** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, __int64 value)
Add name value pair to metadata.

int **Lgsx_MetaAddDouble** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, double value)
Add name value pair to metadata.

int **Lgsx_MetaAddString** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, const wchar_t *pValue)
Add name value pair to metadata.

int **Lgsx_MetaAddDouble3** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, const double *pValue)
Add name value pair to metadata.

int **Lgsx_MetaAddDouble4** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, const double *pValue)
Add name value pair to metadata.

int **Lgsx_MetaAddMetadata** (LgsxMetadataPtr pMetadata, const LgsxPropertyName pName, LgsxMetadataPtr pMetadataIn)
Add name value pair to metadata.

Detailed Description

Functions for reading, writing, and parsing metadata objects.

A metadata object is a property container that holds a set of key-value pairs. Key is a unique name (should use less than 40 chars), and value can be of different types. Metadata objects can be recursive - a property value of a metadata object can be another metadata object.

Function Documentation

◆ Lgsx_MetaAddBool()

```
int Lgsx_MetaAddBool ( LgsxMetadataPtr          pMetadata,
                       const LgsxPropertyName   pName,
                       bool                     value
                     )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
value	[in]Value to add. Be careful that the type of value is important.

Returns

0 as success, otherwise failed.

◆ Lgsx_MetaAddDouble()

```
int Lgsx_MetaAddDouble ( LgsxMetadataPtr          pMetadata,
                         const LgsxPropertyName   pName,
                         double                   value
                       )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
value	[in]Value to add. Be careful that the type of value is important.

Returns

0 as success, otherwise failed.

Lgsx_MetaAddDouble3()

```
int
Lgsx_MetaAddDouble3    ( LgsxMetadataPtr          pMetadata,
                         const LgsxPropertyName   pName,
                         const double *           pValue
                       )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pValue	[in]Value to add. Be careful that the type of value is important.

Returns

0 as success, otherwise failed.

Lgsx_MetaAddDouble4()

```
int
Lgsx_MetaAddDouble4    ( LgsxMetadataPtr          pMetadata,
```

```
                                    const LgsxPropertyName  pName,
                                    const double *          pValue
                                  )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pValue	[in]Value to add. Be careful that the type of value is important.

Returns

0 as success, otherwise failed.

◆ Lgsx_MetaAddInt()

```
int Lgsx_MetaAddInt ( LgsxMetadataPtr         pMetadata,
                      const LgsxPropertyName  pName,
                      int                     value
                    )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
value	[in]Value to add. Be careful that the type of value is important.

Returns

0 as success, otherwise failed.

Lgsx_MetaAddInt64()

```
int Lgsx_MetaAddInt64 ( LgsxMetadataPtr          pMetadata,
                        const LgsxPropertyName   pName,
                        __int64                  value
                      )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
value	[in]Value to add. Be careful that the type of value is important.

Returns

0 as success, otherwise failed.

Lgsx_MetaAddMetadata()

```
int
Lgsx_MetaAddMetadata ( LgsxMetadataPtr          pMetadata,
                       const LgsxPropertyName   pName,
                       LgsxMetadataPtr          pMetadataIn
                     )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pMetadataIn	[in]Value to add. Be careful that the type of

value is important.

Returns
0 as success, otherwise failed.

Lgsx_MetaAddString()

```
int Lgsx_MetaAddString ( LgsxMetadataPtr          pMetadata,
                         const LgsxPropertyName   pName,
                         const wchar_t *          pValue
                       )
```

Add name value pair to metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pValue	[in]Value to add. Be careful that the type of value is important.

Returns
0 as success, otherwise failed.

Lgsx_MetaClone()

```
LgsxMetadataPtr
Lgsx_MetaClone                    ( LgsxMetadataPtr  pMetadata )
```

Clones a metadata container.

Returns

Metadata object. You need to delete it using Lgsx_FreeHandle.

Lgsx_MetaCreateInstance()

LgsxMetadataPtr Lgsx_MetaCreateInstance ( )

Creates an instance of metadata container.

Returns

Metadata object. You need to delete it using Lgsx_FreeHandle.

Lgsx_MetaGetBool()

```
int Lgsx_MetaGetBool ( LgsxMetadataPtr          pMetadata,
                       const LgsxPropertyName   pName,
                       bool *                   pValue
                     )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible.

Parameters

pMetadata [In] Metadata container object.

pName [In] Key name.

pValue [Out] Returned value.

Returns

0 as success, otherwise failed.

Lgsx_MetaGetDouble()

```
int Lgsx_MetaGetDouble ( LgsxMetadataPtr          pMetadata,
                         const LgsxPropertyName   pName,
                         double *                 pValue
                       )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible.

Parameters

pMetadata [In] Metadata container object.
pName [In] Key name.
pValue [Out] Returned value.

Returns

0 as success, otherwise failed.

Lgsx_MetaGetDouble3()

```
int
Lgsx_MetaGetDouble3    ( LgsxMetadataPtr          pMetadata,
                         const LgsxPropertyName   pName,
                         double *                 pValue
                       )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pValue	[Out] Returned value. Need to pre-allocate 3 doubles for returning data.

Returns

0 as success, otherwise failed.

Lgsx_MetaGetDouble4()

```
int
Lgsx_MetaGetDouble4     ( LgsxMetadataPtr           pMetadata,
                          const LgsxPropertyName    pName,
                          double *                  pValue
                        )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pValue	[Out] Returned value. Need to pre-allocate 4 doubles for returning data.

Returns

0 as success, otherwise failed.

Lgsx_MetaGetInt()

```
int Lgsx_MetaGetInt ( LgsxMetadataPtr          pMetadata,
                      const LgsxPropertyName   pName,
                      int *                    pValue
                    )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible.

Parameters

pMetadata [In] Metadata container object.
pName [In] Key name.
pValue [Out] Returned value.

Returns

0 as success, otherwise failed.

Lgsx_MetaGetInt64()

```
int Lgsx_MetaGetInt64 ( LgsxMetadataPtr          pMetadata,
                        const LgsxPropertyName   pName,
                        __int64 *                pValue
                      )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible.

Parameters

pMetadata [In] Metadata container object.
pName [In] Key name.
pValue [Out] Returned value.

Returns
0 as success, otherwise failed.

Lgsx_MetaGetMetadata()

```
int
Lgsx_MetaGetMetadata ( LgsxMetadataPtr            pMetadata,
                       const LgsxPropertyName     pName,
                       LgsxMetadataPtr *          pMetadataOut
                     )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pMetadataOut	[Out] Returned value which is of metadata type.

Returns
0 as success, otherwise failed.

Lgsx_MetaGetString()

```
int Lgsx_MetaGetString ( LgsxMetadataPtr            pMetadata,
                         const LgsxPropertyName     pName,
                         wchar_t *                  pValue,
                         int                        maxLength
                       )
```

Get the value associated with the given name in metadata.

A negative returned value means the name does not exist or the data type is not compatible. If you get a positive value, it indicates that the allocated maxLength is not big enough for the value. In this case, the returned value is the actual length. You may call it again with a bigger buffer (at lease the actual length plus 1).

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.
pValue	[Out] Returned value.
maxLength	Allocated length of pValue.

Returns

0 as success, negative failed, positive value is the actual length which indicates value is longer than allocated maxLength.

◆ Lgsx_MetaHas()

```
int Lgsx_MetaHas ( LgsxMetadataPtr          pMetadata,
                   const LgsxPropertyName   pName
                 )
```

TRUE if the key exists in the metadata.

Parameters

pMetadata	[In] Metadata container object.
pName	[In] Key name.

Returns

1 as true and 0 as false.

Lgsx_MetaMoveNext()

```
int Lgsx_MetaMoveNext ( LgsxMetadataPtr    pMetadata,
                        LgsxPropertyName   pName,
                        int *              pType
                      )
```

Move to next key pair of the metadata.

Note that we have not implemented API to get complex data (arbitrary dimensions).

MetadataType	Value
MetadataType_BOOL	0
MetadataType_INT	1
MetadataType_INT64	2
MetadataType_DOUBLE	3
MetadataType_STRING	4
MetadataType_DOUBLE3D	5
MetadataType_QUAT4D	6
MetadataType_METADATA	7
MetadataType_COMPLEX	8

Parameters

pMetadata [In] Metadata container object.

pName [Out] Returned key name. Please reserve 40 chars for returning string.

pType [Out] Returned value type. See table for more detail of the returned type.

Returns

0 for success. None-zero means end.

Lgsx_MetaRemove()

```
int Lgsx_MetaRemove ( LgsxMetadataPtr          pMetadata,
                      const LgsxPropertyName   pName
                    )
```

Remove a key pair from metadata.

Parameters

pMetadata [In] Metadata container object.
pName [In] Key name.

Returns

0 for success.

Lgsx_MetaReset()

```
int Lgsx_MetaReset ( LgsxMetadataPtr  pMetadata )
```

Resets traverse pointer in the metadata.

This is called before calling Lgsx_MetaMoveNext.

Parameters

pMetadata [In] Metadata container object.

Returns

0 for success.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Reader Functions

Modules

Basic Reader Functions

Project Functions

Setup Functions

Target Functions

Run Functions

GeoTag Functions

SiteMap Functions

Asset Functions

Model and Model Node Functions

Point Cloud Functions

User Coordinate System Functions

Typedefs

typedef void * **LgsxReaderPtr**
Handle for LGSx reader object.

Detailed Description

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Basic Reader Functions

Reader Functions

Functions

LgsxReaderPtr	**Lgsx_ReaderCreate** (int *rError) Create a Reader and return its handle.
int	**Lgsx_ReaderOpen** (**LgsxReaderPtr** reader, const wchar_t *pFilePath, const char *password) Open the given LGSx file with the given Reader using the given password.
int	**Lgsx_ReaderClose** (**LgsxReaderPtr** reader) Closes the LGSx file for this Reader.
int	**Lgsx_ReaderGetLastError** (**LgsxReaderPtr** reader, char *pError, int maxlen) Returns a string description of the last error that occurred with the given Reader.

Detailed Description

Function Documentation

◆ Lgsx_ReaderClose()

int Lgsx_ReaderClose (**LgsxReaderPtr** reader)

Closes the LGSx file for this Reader.

Parameters

reader [in] Reader handle.

Returns

0 for success.

◆ Lgsx_ReaderCreate()

LgsxReaderPtr Lgsx_ReaderCreate (int * rError)

Create a Reader and return its handle.

Parameters

rError [Out] Error code.

Returns

Error code is set to 0 for success and -1 for failure.

Lgsx_ReaderGetLastError()

```
int Lgsx_ReaderGetLastError ( LgsxReaderPtr  reader,
                              char *         pError,
                              int            maxlen
                            )
```

Returns a string description of the last error that occurred with the given Reader.

Parameters

reader [in] Reader handle.
pError [out] Error string buffer.
maxlen [in] Size of error string buffer.

Returns

0 for success.

Lgsx_ReaderOpen()

```
int Lgsx_ReaderOpen ( LgsxReaderPtr  reader,
                      const wchar_t * pFilePath,
                      const char *    password
                    )
```

Open the given LGSx file with the given Reader using the given password.

Parameters

reader [in] Reader handle.
pFilePath [in] Full path to the LGSx file to be opened.
password [in] Password needed to open the LGSx file.

Set to null if there is no password.

Returns

0 for success.

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Project Functions

Reader Functions

Functions

int **Lgsx_ReaderGetMetadata** (**LgsxReaderPtr** reader, uint64_t *pPointCount, **CYBBOX** *pBbox, LgsxMetadataPtr *pMetadata)
Returns the project metadata for the opened project.

int **Lgsx_ReaderGetProjectInfo** (**LgsxReaderPtr** reader, LgsxMetadataPtr *pOwnerInfo, int *width, int *height, int *widthInBytes, ImagePixelType *pixelType, unsigned char **pThumbImage)
Returns the project metadata and thumbnail for the opened project.

int **Lgsx_ReaderGetProjectDescription** (**LgsxReaderPtr** reader, wchar_t *pDesc, int maxlen)
Returns the project description string for the opened project.

Detailed Description

Function Documentation

◆ Lgsx_ReaderGetMetadata()

```
int Lgsx_ReaderGetMetadata ( LgsxReaderPtr      reader,
                             uint64_t *         pPointCount,
                             CYBBOX *           pBbox,
                             LgsxMetadataPtr *  pMetadata
                           )
```

Returns the project metadata for the opened project.

Parameters

reader	[in] Reader handle.
pPointCount	[out] Number of points in the point cloud.
pBbox	[out] Bounding box that contains the project.
pMetadata	[out] Metadata object that contains additional project metadata.

Returns

0 for success.

◆ Lgsx_ReaderGetProjectDescription()

```
int                                 ( LgsxReaderPtr  reader,
```

```
Lgsx_ReaderGetProjectDescription
                                    wchar_t *       pDesc,
                                    int             maxlen
                                   )
```

Returns the project description string for the opened project.

Parameters

reader [in] Reader handle.
pDesc [out] Description string buffer.
maxlen [in] Size of error string buffer.

Returns

0 for success.

◆ Lgsx_ReaderGetProjectInfo()

```
int
Lgsx_ReaderGetProjectInfo ( LgsxReaderPtr       reader,
                            LgsxMetadataPtr *   pOwnerInfo,
                            int *               width,
                            int *               height,
                            int *               widthInBytes,
                            ImagePixelType *    pixelType,
                            unsigned char **    pThumbImage
                           )
```

Returns the project metadata and thumbnail for the opened project.

Parameters

reader [in] Reader handle.
pOwnerInfo [out] Project metadata.

width	[out] Thumbnail image width.
height	[out] Thumbnail image height.
widthInBytes	[out] Thumbnail image byte-width.
pixelType	[out] Thumbnail image pixel type.
pThumbImage	[out] Thumbnail image data.

Returns

0 for success.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Setup Functions

Reader Functions

Functions

int **Lgsx_ReaderEnumSetups** (**LgsxReaderPtr** reader, int *pNumSetup)
Collects Setups in the project and returns the count.

int **Lgsx_ReaderEnumSetupsEx** (**LgsxReaderPtr** reader, **PNT3D** *pLocation, double range, int *pNumSetup)
Collects Setups in the project that are within the given range from the given position.

int **Lgsx_ReaderMoveNextSetup** (**LgsxReaderPtr** reader, uint64_t *setupId, **CYSETUPINFO** *pSetup, LgsxMetadataPtr *pMetadata)
Advances to the next setup in the active collection and returns ID, info, and metadata.

int **Lgsx_ReaderEndSetups** (**LgsxReaderPtr** reader)
Frees the active collection.

int **Lgsx_ReaderGetImageLayerNames** (**LgsxReaderPtr** reader, wchar_t *pLayers, int maxlen)
Retrieve the Image Layer names in the active Setup.

int **Lgsx_ReaderGetPanoImage** (**LgsxReaderPtr** reader, int *width, int *height, int *widthInBytes, ImagePixelType

*pixelType, unsigned char **panoImage)
Get the pano image for the active setup.

int **Lgsx_ReaderGetCubeImage** (**LgsxReaderPtr** reader, const wchar_t *pLayer, **SCyRgdBdyTxf** *txfFromSetup, int *width, int *height, int *widthInBytes, ImagePixelType *pixelType, unsigned char *cubeImages[6])
Get the cubemap corresponding to the given layer name for the active setup.

Detailed Description

Function Documentation

◆ Lgsx_ReaderEndSetups()

int Lgsx_ReaderEndSetups (**LgsxReaderPtr** reader)

Frees the active collection.

Parameters

reader [in] Reader handle.

Returns

Returns 0 for success.

◆ Lgsx_ReaderEnumSetups()

int Lgsx_ReaderEnumSetups (**LgsxReaderPtr** reader,
int * pNumSetup
)

Collects Setups in the project and returns the count.

Parameters

reader [in] Reader handle.

pNumSetup [out] Number of collected setups.

Returns
0 for success.

Lgsx_ReaderEnumSetupsEx()

```
int Lgsx_ReaderEnumSetupsEx ( LgsxReaderPtr  reader,
                              PNT3D *        pLocation,
                              double         range,
                              int *          pNumSetup
                            )
```

Collects Setups in the project that are within the given range from the given position.

Parameters

reader	[in] Reader handle.
pLocation	[in] Position of range filter.
range	[in] Distance from position for range filter.
pNumSetup	[out] Number of collected setups.

Returns
0 for success.

Lgsx_ReaderGetCubeImage()

```
int
Lgsx_ReaderGetCubeImage ( LgsxReaderPtr      reader,
                          const wchar_t *    pLayer,
                          SCyRgdBdyTxf *     txfFromSetup,
                          int *              width,
                          int *              height,
```

```
                                int *                   widthInBytes,
                                ImagePixelType *        pixelType,
                                unsigned char *         cubeImages[6]
                            )
```

Get the cubemap corresponding to the given layer name for the active setup.

Parameters

reader	[in] Reader handle.
pLayer	[in] Cubemap layer to get, see **Lgsx_ReaderGetImageLayerNames()**.
txfFromSetup	[out] Offset from the origin of the scan data (typically at the apparent center of laser beam origin) to the location of the cameras that are used to create the panoramic images.
width	[out] Cubemap image width.
height	[out] Cubemap image height.
widthInBytes	[out] Cubemap image byte-width.
pixelType	[out] Cubemap image pixel type.
cubeImages	[out] Array of cubemap images, one for each cube face.

Returns

Returns 0 for success.

◆ Lgsx_ReaderGetImageLayerNames()

```
int
Lgsx_ReaderGetImageLayerNames   ( LgsxReaderPtr   reader,
                                  wchar_t *       pLayers,
                                  int             maxlen
```

)

Retrieve the Image Layer names in the active Setup.

A Setup may contain one or more "layers" of panoramic raster data. This function returns the list of layer names that are present in the active Setup. Layer names are separated by commas.

Layer names include:

- Camera : Color camera imagery.
- HDR : High Dynamic Range imagery.
- IR : Non-visual Infrared temperature data.
- DisplayIR : Infrared temperature imagery. This is a processed version of the IR data that makes it easier to see the temperature differences.
- ScanIntensity : Grayscale intensity data.
- ScanDepth : Non-visual depth data. Each "pixel" of data represents the return distance for that location.

Parameters

reader	[in] Reader handle.
pLayers	[out] Comma-separated ist of layer names present in Setup object.
maxlen	[in] Length of the given layer string buffer.

Returns

Returns 0 for success.

◆ Lgsx_ReaderGetPanoImage()

```
int
Lgsx_ReaderGetPanoImage ( LgsxReaderPtr reader,
                          int *         width,
```

```
                                int *                height,
                                int *                widthInBytes,
                                ImagePixelType *     pixelType,
                                unsigned char **     panoImage
                              )
```

Get the pano image for the active setup.

Lgsx_ReaderGetPanoImage() is provided to support projects created prior to JetStream version 1.5.0. The vast majority of LGSx files will use cubemaps, which can be retrieved via **Lgsx_ReaderGetCubeImage()**.

Parameters

reader	[in] Reader handle.
width	[out] Pano image width.
height	[out] Pano image height.
widthInBytes	[out] Pano image byte-width.
pixelType	[out] Pano image pixel type.
panoImage	[out] Pano image.

Returns

Returns 0 for success.

◆ Lgsx_ReaderMoveNextSetup()

```
int
Lgsx_ReaderMoveNextSetup    ( LgsxReaderPtr        reader,
                              uint64_t *           setupId,
                              CYSETUPINFO *        pSetup,
                              LgsxMetadataPtr *    pMetadata
                            )
```

Advances to the next setup in the active collection and returns ID, info, and metadata.

Parameters

reader	[in] Reader handle.
setupId	[out] Setup ID.
pSetup	[out] Setup info.
pMetadata	[out] Setup metadata.

Returns

Returns 0 for success. Returns -1 when end of collection is reached.

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Target Functions

Reader Functions

Functions

int **Lgsx_ReaderEnumTarget** (**LgsxReaderPtr** reader, int *pNumTarget)
Collects Targets in the project and returns the count.

int **Lgsx_ReaderEnumTargetOfSetup** (**LgsxReaderPtr** reader, uint64_t setupId, int *pNumTarget)
Collects Targets in the given Setup and returns the count.

int **Lgsx_ReaderMoveNextTarget** (**LgsxReaderPtr** reader, uint64_t *targetId, **CYTARGETINFO** *pTarget, LgsxMetadataPtr *pMetadata)
Advance to the next target in the active collection and return the ID, info, and metadata.

Detailed Description

Function Documentation

◆ Lgsx_ReaderEnumTarget()

```
int Lgsx_ReaderEnumTarget ( LgsxReaderPtr  reader,
                            int *          pNumTarget
                          )
```

Collects Targets in the project and returns the count.

Parameters

reader	[in] Reader handle.
pNumTarget	[out] Number of collected Targets.

Returns

0 for success.

◆ Lgsx_ReaderEnumTargetOfSetup()

```
int
Lgsx_ReaderEnumTargetOfSetup ( LgsxReaderPtr  reader,
                               uint64_t       setupId,
                               int *          pNumTarget
                             )
```

Collects Targets in the given Setup and returns the count.

Parameters

reader	[in] Reader handle.
setupId	[in] Parent Setup ID.
pNumTarget	[out] Number of collected targets.

Returns

0 for success.

◆ Lgsx_ReaderMoveNextTarget()

```
int
Lgsx_ReaderMoveNextTarget ( LgsxReaderPtr      reader,
                            uint64_t *         targetId,
                            CYTARGETINFO *     pTarget,
                            LgsxMetadataPtr *  pMetadata
                          )
```

Advance to the next target in the active collection and return the ID, info, and metadata.

Parameters

reader	[in] Reader handle.
targetId	[out] Target ID.
pTarget	[out] Target info.
pMetadata	[out] Target metadata.

Returns

Returns 0 for success. Returns -1 when end of collection is reached.

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Run Functions

Reader Functions

Functions

int **Lgsx_ReaderEnumRuns** (**LgsxReaderPtr** reader, int *pNumRun)
Collects Runs in the project and returns the count.

int **Lgsx_ReaderMoveNextRun** (**LgsxReaderPtr** reader, uint64_t *runId, **CYTRAJECTORYINFO** *info, int *nVertex, **CYTRAJECTORYVERTEX** **ppPath, LgsxMetadataPtr *pMetadata)
Advance to the next Run in the active collection and return the ID, info, and metadata.

int **Lgsx_ReaderEndRuns** (**LgsxReaderPtr** reader)
Frees the active Run collection.

int **Lgsx_ReaderGetLUTImage** (**LgsxReaderPtr** reader, wchar_t *typeName, int *nBytes, void **lut)
Gets the Setup Camera LUT (Lookup Table) for the desired type.

int **Lgsx_ReaderGetLUTImageOfSetup** (**LgsxReaderPtr** reader, uint64_t setupId, wchar_t *typeName, int *nBytes, void **lut)
Gets the Setup Camera LUT (Lookup Table) for the given Setup.

int **Lgsx_ReaderEnumSetupCamera** (**LgsxReaderPtr** reader, int *pNumSetupCamera)
Collects Setup Cameras in the project and returns the count.

int **Lgsx_ReaderEnumSetupCameraOfSetup** (**LgsxReaderPtr** reader, uint64_t setupId, int *pNumSetupCamera)
Collects Setup Cameras for the given Setup returns the count.

int **Lgsx_ReaderMoveNextSetupCamera** (**LgsxReaderPtr** reader, uint64_t *pSetupCameraId, **CYSETUPCAMERAINFO** *pSetupCamera, LgsxMetadataPtr *pMetadata)
Advance to the next Setup Camera in the active collection and return the ID, info, and metadata.

int **Lgsx_ReaderGetSetupCameraImage** (**LgsxReaderPtr** reader, wchar_t *typeName, int *nBytes, void **blob, int *nThumbBytes, void **thumbBlob)

Get the Setup Camera image and thumbnail image corresponding to the given layer name for the active Setup Camera.

Detailed Description

Function Documentation

Lgsx_ReaderEndRuns()

int Lgsx_ReaderEndRuns (**LgsxReaderPtr** reader)

Frees the active Run collection.

Parameters

reader [in] Reader handle.

Returns

Returns 0 for success.

Lgsx_ReaderEnumRuns()

int Lgsx_ReaderEnumRuns (**LgsxReaderPtr** reader,
int * pNumRun
)

Collects Runs in the project and returns the count.

Runs are also known as "Tracks" or "Walks" to some integrators. Runs are generated by kinnematic scanners.

Parameters

reader [in] Reader handle.
pNumRun [out] Number of collected Runs.

Returns

0 for success.

Lgsx_ReaderEnumSetupCamera()

```
int Lgsx_ReaderEnumSetupCamera ( LgsxReaderPtr  reader,
                                 int *          pNumSetupCamera
                               )
```

Collects Setup Cameras in the project and returns the count.

Parameters

reader	[in] Reader handle.
pNumSetupCamera	[out] Number of collected Setup Cameras.

Returns

0 for success.

Lgsx_ReaderEnumSetupCameraOfSetup()

```
int
Lgsx_ReaderEnumSetupCameraOfSetup ( LgsxReaderPtr  reader,
                                    uint64_t       setupId,
                                    int *          pNumSetupCamera
                                  )
```

Collects Setup Cameras for the given Setup returns the count.

Parameters

reader	[in] Reader handle.
setupId	[in] Parent Setup ID.
pNumSetupCamera	[out] Number of collected Setup Cameras.

Returns

0 for success.

Lgsx_ReaderGetLUTImage()

```
int Lgsx_ReaderGetLUTImage ( LgsxReaderPtr  reader,
                             wchar_t *      typeName,
                             int *          nBytes,
                             void **        lut
                           )
```

Gets the Setup Camera LUT (Lookup Table) for the desired type.

Parameters

reader	[in] Reader handle.
typeName	[in] Desired lookup table type.
nBytes	[out] Number of bytes in lookup table.
lut	[out] Lookup table.

Returns

0 for success.

Lgsx_ReaderGetLUTImageOfSetup()

```
int Lgsx_ReaderGetLUTImageOfSetup ( LgsxReaderPtr  reader,
                                    uint64_t       setupId,
                                    wchar_t *      typeName,
                                    int *          nBytes,
                                    void **        lut
                                  )
```

Gets the Setup Camera LUT (Lookup Table) for the given Setup.

Parameters

reader	[in] Reader handle.
setupId	[in] Setup that owns the lookup table.
typeName	[in] Desired lookup table type.
nBytes	[out] Number of bytes in lookup table.
lut	[out] Lookup table.

Returns

0 for success.

Lgsx_ReaderGetSetupCameraImage()

```
int Lgsx_ReaderGetSetupCameraImage ( LgsxReaderPtr  reader,
                                     wchar_t *      typeName,
                                     int *          nBytes,
                                     void **        blob,
                                     int *          nThumbBytes,
                                     void **        thumbBlob
                                   )
```

Get the Setup Camera image and thumbnail image corresponding to the given layer name for the active Setup Camera.

Parameters

reader	[in] Reader handle.
typeName	[in] Lookup table type.
nBytes	[out] Number of bytes in Setup Camera image.
blob	[out] Setup Camera image.
nThumbBytes	[out] Number of bytes in Setup Camera thumbnail image.
thumbBlob	[out] Setup Camera thumbnail image.

Returns

Returns 0 for success.

Lgsx_ReaderMoveNextRun()

```
int Lgsx_ReaderMoveNextRun ( LgsxReaderPtr          reader,
                             uint64_t *             runId,
                             CYTRAJECTORYINFO *     info,
                             int *                  nVertex,
                             CYTRAJECTORYVERTEX **  ppPath,
                             LgsxMetadataPtr *      pMetadata
                           )
```

Advance to the next Run in the active collection and return the ID, info, and metadata.

Parameters

reader	[in] Reader handle.
runId	[out] Run ID.
info	[out] Target info.
nVertex	[out] Number of vertices in the Run's trajectory.
ppPath	[out] Array of Trajectory Vertex info structs.
pMetadata	[out] Run metadata.

Returns

Returns 0 for success. Returns -1 when end of collection is reached.

Lgsx_ReaderMoveNextSetupCamera()

int Lgsx_ReaderMoveNextSetupCamera	(**LgsxReaderPtr**	reader,
	uint64_t *	pSetupCameraId,
	CYSETUPCAMERAINFO *	pSetupCamera,
	LgsxMetadataPtr *	pMetadata
	)	

Advance to the next Setup Camera in the active collection and return the ID, info, and metadata.

Parameters

reader	[in] Reader handle.
pSetupCameraId	[out] Run ID.
pSetupCamera	[out] Setup Camera info.
pMetadata	[out] Setup Camera metadata.

Returns

Returns 0 for success. Returns -1 when end of collection is reached.

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

GeoTag Functions

Reader Functions

Functions

int **Lgsx_ReaderEnumGeoTags** (**LgsxReaderPtr** reader, int *pNumGeoTag)
Collects GeoTags in the project and returns the count.

int **Lgsx_ReaderEnumGeoTagsOfSetup** (**LgsxReaderPtr** reader, uint64_t setupId, int *pNumGeoTag)
Collects GeoTags associated with the given Setup and returns the count.

int **Lgsx_ReaderMoveNextGeoTag** (**LgsxReaderPtr** reader, uint64_t *geotagId, **CYGEOTAG** *pGeoTag, LgsxMetadataPtr *pMetadata)
Advance to the next GeoTag in the active collection and return the ID, info, and metadata.

Detailed Description

Function Documentation

◆ Lgsx_ReaderEnumGeoTags()

```
int Lgsx_ReaderEnumGeoTags ( LgsxReaderPtr  reader,
                             int *          pNumGeoTag
                           )
```

Collects GeoTags in the project and returns the count.

Parameters

reader	[in] Reader handle.
pNumGeoTag	[out] Number of collected GeoTags.

Returns

0 for success.

◆ Lgsx_ReaderEnumGeoTagsOfSetup()

```
int
Lgsx_ReaderEnumGeoTagsOfSetup ( LgsxReaderPtr  reader,
                                uint64_t       setupId,
                                int *          pNumGeoTag
                              )
```

Collects GeoTags associated with the given Setup and returns the count.

Parameters

reader	[in] Reader handle.
setupId	[in] Setup associated with GeoTags.
pNumGeoTag	[out] Number of collected GeoTags.

Returns

0 for success.

◆ Lgsx_ReaderMoveNextGeoTag()

```
int Lgsx_ReaderMoveNextGeoTag ( LgsxReaderPtr      reader,
                                uint64_t *         geotagId,
                                CYGEOTAG *         pGeoTag,
                                LgsxMetadataPtr *  pMetadata
                              )
```

Advance to the next GeoTag in the active collection and return the ID, info, and metadata.

Parameters

reader	[in] Reader handle.
geotagId	[out] GeoTag ID.
pGeoTag	[out] GeoTag info.
pMetadata	[out] GeoTag metadata.

Returns

Returns 0 for success. Returns -1 when end of collection is reached.

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

SiteMap Functions

Reader Functions

Functions

int **Lgsx_ReaderEnumSiteMaps** (**LgsxReaderPtr** reader, int *pNumSiteMap)
Collects SiteMaps in the project and returns the count.

int **Lgsx_ReaderMoveNextSiteMapImage** (**LgsxReaderPtr** reader, int *width, int *height, int *widthInBytes, ImagePixelType *pixelType, unsigned char **pImage, double world2screen[16])
Advance to the next Sitemap in the active collection and return its background image.

Detailed Description

Function Documentation

◆ Lgsx_ReaderEnumSiteMaps()

```
int Lgsx_ReaderEnumSiteMaps ( LgsxReaderPtr  reader,
                              int *          pNumSiteMap
                            )
```

Collects SiteMaps in the project and returns the count.

Parameters

reader	[in] Reader handle.
pNumSiteMap	[out] Number of collected SiteMaps.

Returns

0 for success.

◆ Lgsx_ReaderMoveNextSiteMapImage()

```
int
Lgsx_ReaderMoveNextSiteMapImage ( LgsxReaderPtr    reader,
                                  int *            width,
                                  int *            height,
                                  int *            widthInBytes,
                                  ImagePixelType * pixelType,
                                  unsigned char ** pImage,
                                  double           world2screen[16]
                                )
```

Advance to the next Sitemap in the active collection and return its background image.

Parameters

reader	[in] Reader handle.
width	[out] Background image width.
height	[out] Background image height.
widthInBytes	[out] Background image width in bytes.
pixelType	[out] Background image pixel type.
pImage	[out] Background image.
world2screen	[out] Background image world to screen matrix.

Returns

Returns 0 for success. Returns -1 when end of collection is reached.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Asset Functions

Reader Functions

Functions

int **Lgsx_ReaderEnumAssets** (**LgsxReaderPtr** reader, int *pNumAsset, bool bGetAll)
Collects SiteMaps in the project and returns the count.

int **Lgsx_ReaderMoveNextAsset** (**LgsxReaderPtr** reader, uint64_t *assetId)
Advance to the next Asset in the active collection and return its ID.

int **Lgsx_ReaderGetAsset** (**LgsxReaderPtr** reader, uint64_t assetId, **CYASSET** *pAsset, LgsxMetadataPtr *pMetadata)
Get Info and metadata for the given Asset ID.

int **Lgsx_GetAssetAsImage** (**LgsxReaderPtr** reader, uint64_t assetId, wchar_t *pName, int maxLenName, int *width, int *height, int *widthInBytes, ImagePixelType *pixelType, unsigned char **pImage)
Get the "payload" image for the given Asset ID.

int **Lgsx_GetAssetThumbImage** (**LgsxReaderPtr** reader, uint64_t assetId, int *width, int *height, int *widthInBytes, ImagePixelType *pixelType, unsigned char **pImage)
Get the thumbnail image for the given Asset ID.

Detailed Description

Function Documentation

◆ Lgsx_GetAssetAsImage()

```
int Lgsx_GetAssetAsImage ( LgsxReaderPtr    reader,
                           uint64_t         assetId,
                           wchar_t *        pName,
                           int              maxLenName,
                           int *            width,
                           int *            height,
                           int *            widthInBytes,
                           ImagePixelType * pixelType,
                           unsigned char ** pImage
                         )
```

Get the "payload" image for the given Asset ID.

Parameters

reader	[in] Reader handle.
assetId	[in] Asset ID.
pName	[out] Asset filename buffer.
maxLenName	[out] Asset filename buffer length.
width	[out] Image width.
height	[out] Image height.
widthInBytes	[out] Image width in bytes.
pixelType	[out] Pixel type.

pImage [out] Asset image.

Returns
Returns 0 for success.

◆ Lgsx_GetAssetThumbImage()

```
int
Lgsx_GetAssetThumbImage ( LgsxReaderPtr reader,
                          uint64_t assetId,
                          int * width,
                          int * height,
                          int * widthInBytes,
                          ImagePixelType * pixelType,
                          unsigned char ** pImage
                        )
```

Get the thumbnail image for the given Asset ID.

Parameters

reader	[in] Reader handle.
assetId	[in] Asset ID.
width	[out] Image width.
height	[out] Image height.
widthInBytes	[out] Image width in bytes.
pixelType	[out] Pixel type.
pImage	[out] Asset thumbnail image.

Returns
Returns 0 for success.

Lgsx_ReaderEnumAssets()

```
int Lgsx_ReaderEnumAssets ( LgsxReaderPtr  reader,
                            int *          pNumAsset,
                            bool           bGetAll
                          )
```

Collects SiteMaps in the project and returns the count.

Parameters

reader	[in] Reader handle.
pNumAsset	[out] Number of collected Assets.
bGetAll	[in] If TRUE, retrieve all assets in the project. If FALSE, only return Assets that are owned directly by the project (that is, exclude Assets that are owned by GeoTags, SiteMaps, or Models).

Returns

0 for success.

Lgsx_ReaderGetAsset()

```
int Lgsx_ReaderGetAsset ( LgsxReaderPtr      reader,
                          uint64_t           assetId,
                          CYASSET *          pAsset,
                          LgsxMetadataPtr *  pMetadata
                        )
```

Get Info and metadata for the given Asset ID.

Parameters

reader [in] Reader handle.
assetId [in] Asset ID.
pAsset [in] Asset info.
pMetadata [in] Asset metadata.

Returns
Returns 0 for success.

◆ Lgsx_ReaderMoveNextAsset()

int Lgsx_ReaderMoveNextAsset (**LgsxReaderPtr** reader,
uint64_t * assetId
)

Advance to the next Asset in the active collection and return its ID.

Parameters
reader [in] Reader handle.
assetId [out] Asset ID.

Returns
Returns 0 for success. Returns -1 when end of collection is reached.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Model and Model Node Functions

Reader Functions

Functions

int **Lgsx_ReaderGetModelNodes** (**LgsxReaderPtr** reader, int *pNumModelNode, uint64_t **ppModelNodeIds)
Retrieve the Model Nodes for this project.

int **Lgsx_ReaderGetModelNodeInfo** (**LgsxReaderPtr** reader, uint64_t nodeId, **CYMODELNODEINFO** *pNodeInfo, uint64_t **ppChildIds)
Retrieve the info for a given Model Node.

int **Lgsx_ReaderGetModelInfo** (**LgsxReaderPtr** reader, uint64_t modelId, **CYMODELINFO** *pModelInfo, LgsxMetadataPtr *pMetadata, uint64_t **ppRefAssetIds)
Retrieve the info for a given Model.

Detailed Description

Function Documentation

Lgsx_ReaderGetModelInfo()

```
int Lgsx_ReaderGetModelInfo ( LgsxReaderPtr      reader,
                              uint64_t           modelId,
                              CYMODELINFO *      pModelInfo,
                              LgsxMetadataPtr *  pMetadata,
                              uint64_t **        ppRefAssetIds
                            )
```

Retrieve the info for a given Model.

Each model object references two Assets: Source Rep and Scene Rep. The Source Rep represents the actual source model file (e.g., an IFC file). The Scene Rep Asset represents the model file converted into a form that is optimized for rendering (it is OpenInventor format).

Parameters

reader	[in] Reader handle.
modelId	[in] ID of the model to query.
pModelInfo	[out] Info for the requested Model.
pMetadata	[out] Model Metadata.
ppRefAssetIds	[out] Model Asset IDs. Number of Assets is contained in the **CYMODELINFO**. These are the Source and Scene Assets mentioned above.

Returns

Returns 0 for success.

Lgsx_ReaderGetModelNodeInfo()

```
int
Lgsx_ReaderGetModelNodeInfo ( LgsxReaderPtr          reader,
                              uint64_t               nodeId,
                              CYMODELNODEINFO *      pNodeInfo,
                              uint64_t **            ppChildIds
                            )
```

Retrieve the info for a given Model Node.

A ModelNode represents an instance of a Model placed somewhere in the scene. Model Node holds a reference to the the Model object, as well as the transformation matrix which positions and orients the model within the project.

Parameters

reader	[in] Reader handle.
nodeId	[in] ID of the model node to query.
pNodeInfo	[out] Info for the requested Model Node.
ppChildIds	[out] Array of child Model Nodes. Note that the number of children in this array is contained within the Model Node info.

Returns

Returns 0 for success.

Lgsx_ReaderGetModelNodes()

```
int
Lgsx_ReaderGetModelNodes    ( LgsxReaderPtr    reader,
                              int *            pNumModelNode,
                              uint64_t **      ppModelNodeIds
```

)

Retrieve the Model Nodes for this project.

Model nodes can form a hierarchical scene graph to combine multiple model "parts" into a single model.

Parameters

reader	[in] Reader handle.
pNumModelNode	[out] Number of Model Nodes returned.
ppModelNodeIds	[out] Array of Model Node IDs.

Returns

Returns 0 for success.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Point Cloud Functions

Reader Functions

Functions

int **Lgsx_ReaderQueryPropertyTypes** (**LgsxReaderPtr** reader)
Returns the Property Types available for points in the project's Point Cloud.

int **Lgsx_ReaderGetPropertyTypeInfo** (**LgsxReaderPtr** reader, int typeMask, const wchar_t *name, int *pnBytes)
Retrieve Property Type information from the Point Cloud.

int **Lgsx_ReaderEnumPoints** (**LgsxReaderPtr** reader, int desiredTypes, int subSample, bool visible)
Collects points in the Point Cloud.

int **Lgsx_ReaderMoveNextPoints** (**LgsxReaderPtr** reader, int chunkSize, int bmpFormat, double *points, float *intens, uchar *colors, float *normals)
Advances to the next group of points in the Point Cloud and returns the basic properties.

int **Lgsx_ReaderMoveNextFields** (**LgsxReaderPtr** reader, int chunkSize, int bmpFormat, void *ppData[])
Advances to the next group of points in the Point Cloud and returns the basic properties.

int **Lgsx_ReaderExtractPointCloud** (**LgsxReaderPtr** reader, char *hspcPathname)
Extract the entire point cloud from the LGSx file and store it to the specified path/file.

int **Lgsx_ReaderGetClassModels** (**LgsxReaderPtr** reader, wchar_t **ppClassModels)
Retrieve the classification model names for this Point Cloud.

int **Lgsx_ReaderGetClassVisibles** (**LgsxReaderPtr** reader, int modelIdx, wchar_t **ppClassNames, int **ppClassVisibles, int **ppClassColors)
Retrieve the classification properties for the given classification model index.

Detailed Description

Function Documentation

◆ Lgsx_ReaderEnumPoints()

```
int Lgsx_ReaderEnumPoints ( LgsxReaderPtr reader,
                            int           desiredTypes,
                            int           subSample,
                            bool          visible
                          )
```

Collects points in the Point Cloud.

Parameters

reader	[in] Reader handle.
desiredTypes	[in] Mask containing the properties to be retrieved during iteration.
subSample	[in] Not surrently supported and should be set to 1.
visible	[in] Limits collection to "visible" points when true. Otherwise collection includes clipped points.

Returns

Returns 0 for success.

Lgsx_ReaderExtractPointCloud()

```
int
Lgsx_ReaderExtractPointCloud ( LgsxReaderPtr  reader,
                               char *         hspcPathname
                             )
```

Extract the entire point cloud from the LGSx file and store it to the specified path/file.

File format will be packed HSPC.

Parameters

reader	[in] Reader handle.
hspcPathname	[in] Path at which to store the extracted file.

Returns

Returns 0 for success.

Lgsx_ReaderGetClassModels()

```
int
Lgsx_ReaderGetClassModels ( LgsxReaderPtr  reader,
                            wchar_t **     ppClassModels
                          )
```

Retrieve the classification model names for this Point Cloud.

Returns the number of model names.

Model name buffer is created by the SDK. Caller must free this buffer via **LgsxUtil_FreeMem()**.

Parameters

reader	[in] Reader handle.
ppClassModels	[out] Classification model names. Names are null-terminated strings packed into a single buffer.

Returns

Returns the number of model names.

◆ Lgsx_ReaderGetClassVisibles()

```
int
Lgsx_ReaderGetClassVisibles ( LgsxReaderPtr  reader,
                              int            modelIdx,
                              wchar_t **     ppClassNames,
                              int **         ppClassVisibles,
                              int **         ppClassColors
                            )
```

Retrieve the classification properties for the given classification model index.

Classification properties are returned as arrays in buffers created by the SDK. Caller must free these buffers via **LgsxUtil_FreeMem()**.

Parameters

reader	[in] Reader handle.
modelIdx	[in] Index of the model to retrieve.
ppClassNames	[out] Array of classification name strings. Strings are null-terminated and packed into the returned buffer.
ppClassVisibles	[out] Array of flags indicating whether the classification is visible by default.

ppClassColors [out] Array of colors for each classification. Each color is 4-byte RGBA format.

Returns

RReturns the number of classifications in the requested model.

Lgsx_ReaderGetPropertyTypeInfo()

```
int
Lgsx_ReaderGetPropertyTypeInfo ( LgsxReaderPtr reader,
                                 int           typeMask,
                                 const wchar_t * name,
                                 int *         pnBytes
                               )
```

Retrieve Property Type information from the Point Cloud.

Specifically, it returns the number of bytes used for the property being queried. This function fails if the requested property is not present in the Point Cloud. Only one property can be queried at a time.

Each Point Cloud contains a set of Data Properties. Basic properties include position, color, intensity, and point normal, etc. Point Clouds can also contain optional types: SetupIndex and Classification. Finally, a point cloud can have user-defined properties. This function allows the application to query for which optional and user-defined properties are present in the Point Cloud.

Parameters

reader [in] Reader handle.

typeMask [in] Property type to query (see Property type

masks).

name [in] Property name being queried. Only used for user-defined types.

pnBytes [out] Number of bytes used for values of this property type.

Returns

Returns 0 for success.

◆ Lgsx_ReaderMoveNextFields()

```
int Lgsx_ReaderMoveNextFields ( LgsxReaderPtr  reader,
                                int            chunkSize,
                                int            bmpFormat,
                                void *         ppData[]
                              )
```

Advances to the next group of points in the Point Cloud and returns the basic properties.

Caller passes in an array of buffer pointers. Each array index corresponds to one of the property masks. Individual property types can be skipped by setting the corresponding buffer pointer to null. Buffers need to be sized to the number of values specified by **chunkSize**.

Parameters

reader [in] Reader handle.

chunkSize [in] Number of points to read per call.

bmpFormat [in] Bitmap format to use for returned colors (BitmapFormat__RGB or BitmapFormat__RGBA)

ppData [out] Array of buffers to be filled with point data.

Returns
Returns 0 for success.

Lgsx_ReaderMoveNextPoints()

```
int Lgsx_ReaderMoveNextPoints ( LgsxReaderPtr reader,
                                int           chunkSize,
                                int           bmpFormat,
                                double *      points,
                                float *       intens,
                                uchar *       colors,
                                float *       normals
                              )
```

Advances to the next group of points in the Point Cloud and returns the basic properties.

Parameters

reader	[in] Reader handle.
chunkSize	[in] Number of points to read per call.
bmpFormat	[in] Bitmap format to use for returned colors (BitmapFormat__RGB or BitmapFormat__RGBA)
points	[out] XYZ data.
intens	[out] Scan Intensity data.
colors	[out] Color data.
normals	[out] Normal vectors.

Returns
Returns 0 for success.

Lgsx_ReaderQueryPropertyTypes()

int Lgsx_ReaderQueryPropertyTypes (**LgsxReaderPtr** reader)

Returns the Property Types available for points in the project's Point Cloud.

Property types are defined in LgsxClient.h, and are things like XYZ, Intensity, Color, Normal, etc.

Parameters

reader [in] Reader handle.

Returns

Property type mask for this project.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

User Coordinate System Functions

Reader Functions

Functions

int **Lgsx_ReaderEnumCoordSystems** (**LgsxReaderPtr** reader, int *pNumCs)
Collects User Coordinate Systems.

int **Lgsx_ReaderMoveNextCoordSystems** (**LgsxReaderPtr** reader, bool *pActive, wchar_t *pName, int maxName, wchar_t *pWkt, int maxWkt, **SCyRgdBdyTxf** *pTxf)
Advances to the next User Coordinate System (UCS) object and returns its properties.

Detailed Description

Function Documentation

Lgsx_ReaderEnumCoordSystems()

```
int Lgsx_ReaderEnumCoordSystems ( LgsxReaderPtr  reader,
                                  int *          pNumCs
                                )
```

Collects User Coordinate Systems.

Parameters

reader [in] Reader handle.
pNumCs [out] Number of User Coordinate Systems.

Returns

Returns 0 for success.

Lgsx_ReaderMoveNextCoordSystems()

```
int
Lgsx_ReaderMoveNextCoordSystems ( LgsxReaderPtr     reader,
                                  bool *            pActive,
                                  wchar_t *         pName,
                                  int               maxName,
                                  wchar_t *         pWkt,
                                  int               maxWkt,
                                  SCyRgdBdyTxf *    pTxf
```

)

Advances to the next User Coordinate System (UCS) object and returns its properties.

Well Known Text (WKT) is a standard format for specifying Coordincate References Systems (e.g, state planes, UTM, etc). You would include a WKT string in a UCS to give a project real-world position information. If the WKT string is empty, then the coordinates are localized Cartesian coordinates.

Parameters

reader	[in] Reader handle.
pActive	[out] Active flag, set to true if this UCS is active.
pName	[out] UCS name.
maxName	[in] Max size of name buffer.
pWkt	[out] WKT string for this UCS.
maxWkt	[in] RMax size of WKT buffer.
pTxf	[in] Transform/rotation for this UCS.

Returns

Returns 0 for success.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Structures

Here are the data structures with brief descriptions:

CYASSET	Structure for Asset (used by JetStream reader/writer)
CYBBOX	Structure for bounding box (min, max) corners
CYGEOTAG	Structure for geotag (used by JetStream reader/writer) Note, that position is specified in project coordinates regardless of if it is defined within Setup
CYMODELINFO	Struct of returned Model info
CYMODELNODEINFO	Struct of returned ModelNode info
CYRANGECUBE	Structure for an arbitrary range cube (a box)
CYRECT	Structure for rectangle (left, top, right, bottom)
CYSETUPCAMERAINFO	Structure for setup camera info (used by JetStream reader/writer)
CYSETUPINFO	Structure for setup info (used by JetStream reader/writer) This has been extended to support both TLS setup and mobile setup (WayPoint along track of run)

CYTARGETINFO	Structure for target info (used by JetStream reader/writer) Note that origin and normal are specified in Setup coordinate system
CYTRAJECTORYINFO	Structure for track (a Run, Walk or Fly) header info of mobile scanning (such as BLK2GO)
CYTRAJECTORYVERTEX	Structure for trajectory (a Run, Walk or Fly) node of mobile scanning (such as BLK2GO)
M3DDUALFISHEYE	Structure for M3D dual fish eye camera model
M3DFLAT	Structure for M3D flat camera model
M3DLUTCAMERA	Structure for M3D LUT camera model
M3DPINHOLE	Structure for M3D pinhole camera model
PNT2D	Structure for 2D point (or vector, defined as 2 doubles)
PNT3D	Structure for 3D point (or vector, defined as 3 doubles)
QUA4D	Structure for Quaternion (or vector, defined as 4 doubles)
SCyRgdBdyTxf	Rigid transformation represented by a translation and a quaternion

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYASSET Struct Reference

Structure for Asset (used by JetStream reader/writer) More...

```
#include <LgsxClient.h>
```

Data Fields

wchar_t	**name** [80] Name of the asset.
wchar_t	**type** [40] Asset types of following names.
wchar_t	**filename** [256] A temporary file name in which the asset data is stored.

Detailed Description

Structure for Asset (used by JetStream reader/writer)

Field Documentation

filename

wchar_t CYASSET::filename[256]

A temporary file name in which the asset data is stored.

You need to process it immediately as next call to get asset will cause the file unlinked.

name

wchar_t CYASSET::name[80]

Name of the asset.

In case of file asset, the name may imply the original file name, which contains file extension about the type of data stored. For example, a model file asset may be of DXF, COE, OBJ, IFC types. Document file asset can be PDF or other types. A file asset can also be of image types.

type

wchar_t CYASSET::type[40]

Asset types of following names.

Value	Meaning
"Image"	The blob is an image that you may use Lgsx_JsGetAssetAsImage (may fail of unsupported image).
"BackgroundImage"	Same as Image but it is being used by one or more Sitemaps as a background image.
"Slice"	Same as Image, but it was made from a snapshot of a slice.
"View"	Same as Image, but it was made from a snapshot (screen copy).
"File"	The blob can be any file type (PDF, geometry model types, or image, implied by its file extension).

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYBBOX Struct Reference

Structure for bounding box (min, max) corners. More...

```
#include <LgsxClient.h>
```

Data Fields

PNT3D	**minCorner** Corner with smaller coordinate values.
PNT3D	**maxCorner** corner with bigger coordinate values

Detailed Description

Structure for bounding box (min, max) corners.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYGEOTAG Struct Reference

Structure for geotag (used by JetStream reader/writer) Note, that position is specified in project coordinates regardless of if it is defined within Setup. More...

```
#include <LgsxClient.h>
```

Data Fields

PNT3D	**position** location is specified in project coordinates
wchar_t	**name** [40] name of asset (does not need to be unique)
uint64_t	**assetId** when specified non-zero, this Geotag contains Asset.
uint64_t	**geotagId** unique ID for the geotag.
uint64_t	**setupId** when specified non-zero, this Geotag belongs to the Setup

Detailed Description

Structure for geotag (used by JetStream reader/writer) Note, that position is specified in project coordinates regardless of if it is defined within Setup.

When Geotag is added outside of Setup context, it belongs to the project level.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYMODELINFO Struct Reference

struct of returned Model info More...

```
#include <LgsxClient.h>
```

Data Fields

uint64_t	**id** ID of this object for direct access.
uint64_t	**sourceId** Asset ID for source model file (e.g.
uint64_t	**sceneRepId** Asset ID for scene rep file (OpenInventor).
wchar_t	**name** [80] Model name (contains file type)
double	**scale** Scale from meters (e.g., mm = 1000)
SCyRgdBdyTxf	**txf** Translation from Model center to model CS.

PNT3D	**sceneRepOffset** Translation from Model CS to scene rep origin.
uint32_t	**numRef** When non-zero, there are more files referenced by the model file.

Detailed Description

struct of returned Model info

Field Documentation

sceneRepId

uint64_t CYMODELINFO::sceneRepId

Asset ID for scene rep file (OpenInventor).

This Asset is a version of the source model file that's been optimized for rendering.

sourceId

uint64_t CYMODELINFO::sourceId

Asset ID for source model file (e.g.

IFC file, etc.)

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYMODELNODEINFO Struct Reference

struct of returned ModelNode info More...

```
#include <LgsxClient.h>
```

Data Fields

uint64_t	**id** ID of this object for direct access.
uint64_t	**modelId** ID of model (which is referenced not owned)
wchar_t	**name** [80] Name of this model instance.
SCyRgdBdyTxf	**txf** Transformation from Point Cloud render offset to Model center.
uint32_t	**numChild** When non-zero, this is a group of models.

Detailed Description

struct of returned ModelNode info

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYRANGECUBE Struct Reference

Structure for an arbitrary range cube (a box) More...

```
#include <LgsxClient.h>
```

Data Fields

PNT3D	**corners** [4] Min corner point and near corners at x, y, z directions.

Detailed Description

Structure for an arbitrary range cube (a box)

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYRECT Struct Reference

Structure for rectangle (left, top, right, bottom) More...

```
#include <LgsxClient.h>
```

Data Fields

int **left**
Left.

int **top**
Top.

int **right**
Right.

int **bottom**
Bottom.

Detailed Description

Structure for rectangle (left, top, right, bottom)

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYSETUPCAMERAINFO Struct Reference

Structure for setup camera info (used by JetStream reader/writer) More...

```
#include <LgsxClient.h>
```

Data Fields

uint64_t	**setupCameraId** Setup Camera ID.
uint64_t	**setupId** Parent Setup ID.
wchar_t	**name** [40] Setup Camera name.
M3DPOSETYPE	**m_type** Camera model type.
int	**m_widthImage** Image width.

int **m_heightImage**
Image height.

int **padding**
Padding to 8-byte alignment.

union {

M3DPINHOLE **m_Pinhole**

M3DDUALFISHEYE **m_DualFisheye**

M3DLUTCAMERA **m_Lut**

M3DFLAT **m_Flat**

};

Camera model parameters (varies by camera model type)

Detailed Description

Structure for setup camera info (used by JetStream reader/writer)

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYSETUPINFO Struct Reference

Structure for setup info (used by JetStream reader/writer) This has been extended to support both TLS setup and mobile setup (WayPoint along track of run) More...

```
#include <LgsxClient.h>
```

Data Fields

SCyRgdBdyTxf	**pose** Pose translation(xyz) + quaternion(uvws) in project coordinates.
wchar_t	**name** [80] Name needs to be unique in all TLS and mobile setups.
uint64_t	**time** Mobile only: timestamp of mobile setup in UTC seconds (this should be specified for mobile setups).
int	**vertexIndex** Mobile only: index of the closest trajectory vertex.

int	**setupIndex** SETUPIDX that is used in point cloud property PM_SETUPIDX.

Detailed Description

Structure for setup info (used by JetStream reader/writer) This has been extended to support both TLS setup and mobile setup (WayPoint along track of run)

Field Documentation

time

uint64_t CYSETUPINFO::time

Mobile only: timestamp of mobile setup in UTC seconds (this should be specified for mobile setups).

UTC time is the number of seconds since 1/1/1970.

vertexIndex

int CYSETUPINFO::vertexIndex

Mobile only: index of the closest trajectory vertex.

If -1 is specified it will be determined by time.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYTARGETINFO Struct Reference

Structure for target info (used by JetStream reader/writer) Note that origin and normal are specified in Setup coordinate system. More...

```
#include <LgsxClient.h>
```

Data Fields

int	**type** Target type.
PNT3D	**origin** Location specified in Setup coordinate system.
PNT3D	**normal** Direction specified in Setup coordinate system.
double	**radius** 0.0 means none, sphere target needs it
wchar_t	**label** [40] User-facing target name.

Detailed Description

Structure for target info (used by JetStream reader/writer) Note that origin and normal are specified in Setup coordinate system.

A target always belongs to a Setup.

Field Documentation

label

wchar_t CYTARGETINFO::label[40]

User-facing target name.

Does not need to be unique.

normal

PNT3D CYTARGETINFO::normal

Direction specified in Setup coordinate system.

0,0,0 means no normal

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYTRAJECTORYINFO Struct Reference

Structure for track (a Run, Walk or Fly) header info of mobile scanning (such as BLK2GO) More...

```
#include <LgsxClient.h>
```

Data Fields

wchar_t	**jobName** [40] Job name (this could be empty if there is not multiple jobs)
wchar_t	**runName** [40] Track (run) name: needs to be unique within job.
uint64_t	**startTime** Start timestamp in UTC seconds (accurate start time should be computed from first vertex) UTC time is the number of seconds since 1/1/1970.
uint64_t	**stopTime** End timestamp in UTC seconds (accurate end time should be computed from last vertex) UTC time is the number of seconds since 1/1/1970.

double	**timeScale** Scale from the ticks stored in each vertex to actual time lapse.
int	**startSetupIndex** SETUPIDX of the first setup in track/run.
double	**scaleFactor** Scale from vertex index to the SETUPIDX.

Detailed Description

Structure for track (a Run, Walk or Fly) header info of mobile scanning (such as BLK2GO)

Send comments on this topic to LGSx SDK Support.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

CYTRAJECTORYVER TEX Struct Reference

Structure for trajectory (a Run, Walk or Fly) node of mobile scanning (such as BLK2GO) More...

```
#include <LgsxClient.h>
```

Data Fields

uint64_t	**time** Number of ticks since the start of trajectory (this times a scale to get actual lapse from trajectory's startTime)
PNT3D	**trans** Position of vertex in project coordinates.
QUA4D	**orientation** Orientation (quaternion UVW + Scalar) of vertex in project coordinates.
int	**setupIndex** SETUPIDX identify points that belong (or are closest) to this vertex.
double	**quality** quality (0.0 means none)

Detailed Description

Structure for trajectory (a Run, Walk or Fly) node of mobile scanning (such as BLK2GO)

Field Documentation

setupIndex

int CYTRAJECTORYVERTEX::setupIndex

SETUPIDX identify points that belong (or are closest) to this vertex.

Note that the storage model does not currently have this property for each vertex. Because of that, when you read trajectory vertexs you may notice that they do not match the values used to build it. This is not a problem. The value from the reader is a computed value from CWTRAJECTORYINFO parameters startSetupIndex and scaleFactor.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

M3DDUALFISHEYE Struct Reference

Structure for M3D dual fish eye camera model. More...

```
#include <LgsxClient.h>
```

Data Fields

double	**m_nStartLimit** start angle of the first position center
double	**m_nEndLimit** end angle of the first position center
double	**m_nRadius** in meters: radius of the sphere
double	**m_ptCamPosition_1** [3] position 3D (double x,y,z) of the first center of half sphere in project coordinates
double	**m_ptCamPosition_2** [3] position 3D (double x,y,z) of the second center of half sphere in project coordinates

double	**m_orientation** [4] Quaternion [w, x, y, z] in project coordinates.

Detailed Description

Structure for M3D dual fish eye camera model.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

M3DFLAT Struct Reference

Structure for M3D flat camera model. More...

```
#include <LgsxClient.h>
```

Data Fields

double	**m_Flat** [20]

Detailed Description

Structure for M3D flat camera model.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

M3DLUTCAMERA Struct Reference

Structure for M3D LUT camera model. More...

```
#include <LgsxClient.h>
```

Data Fields

double	**m_scale** angle of every pixel
double	**m_radius** in meters: radius used for generation of the panorama
double	**m_sphereOrigin** [3] origin of the sphere in project coordinates
int	**m_roi** [4] region of interest of the panorama in pixel
double	**m_orientation** [4] Quaternion [w, x, y, z] in project coordinates.

Detailed Description

Structure for M3D LUT camera model.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Public Types | Data Fields

M3DPINHOLE Struct Reference

Structure for M3D pinhole camera model. More...

```
#include <LgsxClient.h>
```

Public Types

enum {
distortionK1 = 0 ,
distortionK2 = 1 ,
distortionP1 = 2 ,
distortionP2 = 3 ,
distortionK3 = 4 ,
distortionK4 = 5 ,
distortionK5 = 6 ,
distortionK6 = 7 ,
distortionElements
}

Data Fields

double **m_xPrincipalPoint**
In meters: principal point x.

double **m_yPrincipalPoint**

	In meters: principal point y.
double	**m_xFocal** In meters: focal length x.
double	**m_yFocal** In meters: focal length y.
double	**m_coeffDistortions** [distortionElements] distortion coefficients
double	**m_position** [3] position of the camera in project coordinates
double	**m_orientation** [4] Quaternion [w, x, y, z] in project coordinates.

Detailed Description

Structure for M3D pinhole camera model.

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

PNT2D Struct Reference

Structure for 2D point (or vector, defined as 2 doubles) More...

```
#include <LgsxClient.h>
```

Data Fields

double **x**
double **y**

Detailed Description

Structure for 2D point (or vector, defined as 2 doubles)

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

PNT3D Struct Reference

Structure for 3D point (or vector, defined as 3 doubles) More...

```
#include <LgsxClient.h>
```

Data Fields

double	**x**
double	**y**
double	**z**

Detailed Description

Structure for 3D point (or vector, defined as 3 doubles)

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

QUA4D Struct Reference

Structure for Quaternion (or vector, defined as 4 doubles) More...

```
#include <LgsxClient.h>
```

Data Fields

double	**u**
double	**v**
double	**w**
double	**s**

Detailed Description

Structure for Quaternion (or vector, defined as 4 doubles)

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Data Fields

SCyRgdBdyTxf Struct Reference

Rigid transformation represented by a translation and a quaternion. More...

```
#include <LgsxClient.h>
```

Data Fields

double **mDx**
X translation.

double **mDy**
Y translation.

double **mDz**
Z translation.

double **mU**
Quaternion U.

double **mV**
Quaternion V.

double **mW**

	Quaternion W.
double	**mS** Quaternion S.

Detailed Description

Rigid transformation represented by a translation and a quaternion.

(mDx, mDy, mDz) is the translation component. (mU, mV, mW, mS) is the rotation as quaternion in which the first 3 relates to rotation axis. Quaternion is represented as rotating around given axis a given angle and the values are normalized: So (U,V,W) = axis * sin(theta/2)), S = cos(theta/2).

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Here is a list of all documented struct and union fields with links to the struct/union documentation for each field:

- a -

- assetId : **CYGEOTAG**

- b -

- bottom : **CYRECT**

- c -

- corners : **CYRANGECUBE**

- f -

- filename : **CYASSET**

- g -

- geotagId : **CYGEOTAG**

- i -

- id : **CYMODELINFO**, **CYMODELNODEINFO**

- j -

- jobName : **CYTRAJECTORYINFO**

- l -

- label : **CYTARGETINFO**
- left : **CYRECT**

- m -

- m_coeffDistortions : **M3DPINHOLE**
- m_heightImage : **CYSETUPCAMERAINFO**
- m_nEndLimit : **M3DDUALFISHEYE**
- m_nRadius : **M3DDUALFISHEYE**
- m_nStartLimit : **M3DDUALFISHEYE**
- m_orientation : **M3DDUALFISHEYE**, **M3DLUTCAMERA**, **M3DPINHOLE**
- m_position : **M3DPINHOLE**
- m_ptCamPosition_1 : **M3DDUALFISHEYE**
- m_ptCamPosition_2 : **M3DDUALFISHEYE**
- m_radius : **M3DLUTCAMERA**
- m_roi : **M3DLUTCAMERA**
- m_scale : **M3DLUTCAMERA**
- m_sphereOrigin : **M3DLUTCAMERA**
- m_type : **CYSETUPCAMERAINFO**
- m_widthImage : **CYSETUPCAMERAINFO**
- m_xFocal : **M3DPINHOLE**
- m_xPrincipalPoint : **M3DPINHOLE**
- m_yFocal : **M3DPINHOLE**
- m_yPrincipalPoint : **M3DPINHOLE**
- maxCorner : **CYBBOX**
- mDx : **SCyRgdBdyTxf**
- mDy : **SCyRgdBdyTxf**
- mDz : **SCyRgdBdyTxf**
- minCorner : **CYBBOX**
- modelId : **CYMODELNODEINFO**
- mS : **SCyRgdBdyTxf**

- mU : **SCyRgdBdyTxf**
- mV : **SCyRgdBdyTxf**
- mW : **SCyRgdBdyTxf**

- n -

- name : **CYASSET**, **CYGEOTAG**, **CYMODELINFO**, **CYMODELNODEINFO**, **CYSETUPCAMERAINFO**, **CYSETUPINFO**
- normal : **CYTARGETINFO**
- numChild : **CYMODELNODEINFO**
- numRef : **CYMODELINFO**

- o -

- orientation : **CYTRAJECTORYVERTEX**
- origin : **CYTARGETINFO**

- p -

- padding : **CYSETUPCAMERAINFO**
- pose : **CYSETUPINFO**
- position : **CYGEOTAG**

- q -

- quality : **CYTRAJECTORYVERTEX**

- r -

- radius : **CYTARGETINFO**
- right : **CYRECT**
- runName : **CYTRAJECTORYINFO**

- s -

- scale : **CYMODELINFO**

- scaleFactor : **CYTRAJECTORYINFO**
- sceneRepId : **CYMODELINFO**
- sceneRepOffset : **CYMODELINFO**
- setupCameraId : **CYSETUPCAMERAINFO**
- setupId : **CYGEOTAG**, **CYSETUPCAMERAINFO**
- setupIndex : **CYSETUPINFO**, **CYTRAJECTORYVERTEX**
- sourceId : **CYMODELINFO**
- startSetupIndex : **CYTRAJECTORYINFO**
- startTime : **CYTRAJECTORYINFO**
- stopTime : **CYTRAJECTORYINFO**

- t -

- time : **CYSETUPINFO**, **CYTRAJECTORYVERTEX**
- timeScale : **CYTRAJECTORYINFO**
- top : **CYRECT**
- trans : **CYTRAJECTORYVERTEX**
- txf : **CYMODELINFO**, **CYMODELNODEINFO**
- type : **CYASSET**, **CYTARGETINFO**

- v -

- vertexIndex : **CYSETUPINFO**

LGSx SDK API Reference (Version 2024.0.0) 2024.0.0

Here is a list of all documented variables with links to the struct/union documentation for each field:

- a -

- assetId : **CYGEOTAG**

- b -

- bottom : **CYRECT**

- c -

- corners : **CYRANGECUBE**

- f -

- filename : **CYASSET**

- g -

- geotagId : **CYGEOTAG**

- i -

- id : **CYMODELINFO**, **CYMODELNODEINFO**

- j -

- jobName : **CYTRAJECTORYINFO**

- l -

- label : **CYTARGETINFO**
- left : **CYRECT**

- m -

- m_coeffDistortions : **M3DPINHOLE**
- m_heightImage : **CYSETUPCAMERAINFO**
- m_nEndLimit : **M3DDUALFISHEYE**
- m_nRadius : **M3DDUALFISHEYE**
- m_nStartLimit : **M3DDUALFISHEYE**
- m_orientation : **M3DDUALFISHEYE**, **M3DLUTCAMERA**, **M3DPINHOLE**
- m_position : **M3DPINHOLE**
- m_ptCamPosition_1 : **M3DDUALFISHEYE**
- m_ptCamPosition_2 : **M3DDUALFISHEYE**
- m_radius : **M3DLUTCAMERA**
- m_roi : **M3DLUTCAMERA**
- m_scale : **M3DLUTCAMERA**
- m_sphereOrigin : **M3DLUTCAMERA**
- m_type : **CYSETUPCAMERAINFO**
- m_widthImage : **CYSETUPCAMERAINFO**
- m_xFocal : **M3DPINHOLE**
- m_xPrincipalPoint : **M3DPINHOLE**
- m_yFocal : **M3DPINHOLE**
- m_yPrincipalPoint : **M3DPINHOLE**
- maxCorner : **CYBBOX**
- mDx : **SCyRgdBdyTxf**
- mDy : **SCyRgdBdyTxf**
- mDz : **SCyRgdBdyTxf**
- minCorner : **CYBBOX**
- modelId : **CYMODELNODEINFO**
- mS : **SCyRgdBdyTxf**

- mU : **SCyRgdBdyTxf**
- mV : **SCyRgdBdyTxf**
- mW : **SCyRgdBdyTxf**

- n -

- name : **CYASSET**, **CYGEOTAG**, **CYMODELINFO**, **CYMODELNODEINFO**, **CYSETUPCAMERAINFO**, **CYSETUPINFO**
- normal : **CYTARGETINFO**
- numChild : **CYMODELNODEINFO**
- numRef : **CYMODELINFO**

- o -

- orientation : **CYTRAJECTORYVERTEX**
- origin : **CYTARGETINFO**

- p -

- padding : **CYSETUPCAMERAINFO**
- pose : **CYSETUPINFO**
- position : **CYGEOTAG**

- q -

- quality : **CYTRAJECTORYVERTEX**

- r -

- radius : **CYTARGETINFO**
- right : **CYRECT**
- runName : **CYTRAJECTORYINFO**

- s -

- scale : **CYMODELINFO**

- scaleFactor : **CYTRAJECTORYINFO**
- sceneRepId : **CYMODELINFO**
- sceneRepOffset : **CYMODELINFO**
- setupCameraId : **CYSETUPCAMERAINFO**
- setupId : **CYGEOTAG**, **CYSETUPCAMERAINFO**
- setupIndex : **CYSETUPINFO**, **CYTRAJECTORYVERTEX**
- sourceId : **CYMODELINFO**
- startSetupIndex : **CYTRAJECTORYINFO**
- startTime : **CYTRAJECTORYINFO**
- stopTime : **CYTRAJECTORYINFO**

- t -

- time : **CYSETUPINFO**, **CYTRAJECTORYVERTEX**
- timeScale : **CYTRAJECTORYINFO**
- top : **CYRECT**
- trans : **CYTRAJECTORYVERTEX**
- txf : **CYMODELINFO**, **CYMODELNODEINFO**
- type : **CYASSET**, **CYTARGETINFO**

- v -

- vertexIndex : **CYSETUPINFO**
