

LGSx SDK

2025.0.1 (2025.0.1+2569cb8)

Generated by Doxygen 1.13.2

Send comments on this topic to LGSx SDK Support

tls.sdksupport@leica-geosystems.com

Copyright © 2025, Leica Geosystems, AG.

1 Leica LGSx SDK Getting Started Guide	1
1.1 Overview	1
1.2 Windows Integration	1
1.3 Linux Integration	2
1.4 WSL Integration	2
1.5 SDK Know-How	2
2 Release notes	3
2.1 Release Notes 2025.0.1	3
2.1.1 Changelog	3
2.1.1.1 Added	3
2.1.1.2 Fixed	3
2.1.1.3 Other	3
2.1.2 Additional Notes	4
2.1.3 Packaging	4
2.1.4 Licensing	4
2.2 Release Notes 2025.0.0	4
2.2.1 Changelog	4
2.2.1.1 Added	4
2.2.1.2 Changed	5
2.2.1.3 Deprecated	6
2.2.1.4 Removed	6
2.2.1.5 Fixed	6
2.2.1.6 Other	6
2.2.2 Additional Notes	6
2.2.3 Packaging	6
2.2.4 Licensing	7
2.3 Release Notes 2024.0.1	7
2.3.1 What's New	7
2.3.2 Packaging	7
2.3.3 Fixed Issues	7
2.3.4 Known Issues	7
2.3.5 Licensing	8
2.4 Release Notes 2024.0.0	8
2.4.1 What's New	8
2.4.2 Packaging	8
2.4.3 Fixed Issues	8
2.4.4 Known Issues	8
2.4.5 Licensing	8
3 Deprecated List	9
4 Topic Index	11

4.1 Topics	11
5 Data Structure Index	13
5.1 Data Structures	13
6 File Index	15
6.1 File List	15
7 Topic Documentation	17
7.1 Reader Functions	17
7.1.1 Detailed Description	18
7.1.2 Basic Reader Functions	18
7.1.2.1 Detailed Description	18
7.1.2.2 Function Documentation	18
7.1.3 Project Functions	20
7.1.3.1 Detailed Description	20
7.1.3.2 Function Documentation	20
7.1.4 Asset Functions	22
7.1.4.1 Detailed Description	23
7.1.4.2 Function Documentation	23
7.1.5 Categories Functions	27
7.1.5.1 Detailed Description	28
7.1.5.2 Function Documentation	29
7.1.6 Fields Functions	36
7.1.6.1 Detailed Description	36
7.1.6.2 Function Documentation	36
7.1.7 Migrated Fields	40
7.1.7.1 Detailed Description	40
7.1.7.2 Enumeration Type Documentation	40
7.1.7.3 Function Documentation	41
7.1.8 Setup Functions	41
7.1.8.1 Detailed Description	41
7.1.8.2 Function Documentation	41
7.1.9 Target Functions	47
7.1.9.1 Detailed Description	47
7.1.9.2 Function Documentation	47
7.1.10 Run Functions	48
7.1.10.1 Detailed Description	49
7.1.10.2 Function Documentation	49
7.1.11 GeoTag Functions	53
7.1.11.1 Detailed Description	54
7.1.11.2 Function Documentation	54
7.1.12 Sitemap Functions	65

7.1.12.1 Detailed Description	66
7.1.12.2 Function Documentation	66
7.1.13 Model and Model Node Functions	70
7.1.13.1 Detailed Description	70
7.1.13.2 Function Documentation	70
7.1.14 Point Cloud Functions	73
7.1.14.1 Detailed Description	73
7.1.14.2 Function Documentation	73
7.1.15 User Coordinate System Functions	77
7.1.15.1 Detailed Description	78
7.1.15.2 Function Documentation	78
7.2 General Functions	79
7.2.1 Detailed Description	80
7.2.2 Function Documentation	80
7.2.2.1 Lgsx_FreeHandle()	80
7.2.2.2 Lgsx_GetLastError()	81
7.2.2.3 Lgsx_GetLastErrorNo()	81
7.2.2.4 Lgsx_GetProductVersion()	81
7.2.2.5 Lgsx_InitProductBuildNumber()	81
7.2.2.6 Lgsx_InitProductName()	82
7.2.2.7 Lgsx_InitProductVersion()	82
7.2.2.8 Lgsx_InitProductVersionEx()	82
7.2.2.9 LgsxUtil_A2W()	83
7.2.2.10 LgsxUtil_AllocMem()	83
7.2.2.11 LgsxUtil_FreeMem()	83
7.2.2.12 LgsxUtil_GetCurrentDateString()	84
7.2.2.13 LgsxUtil_GetCurrentTimeString()	84
7.2.2.14 LgsxUtil_Sleep()	84
7.2.2.15 LgsxUtil_StrDupA()	84
7.2.2.16 LgsxUtil_StrDupW()	85
7.2.2.17 LgsxUtil_TimeEnd()	85
7.2.2.18 LgsxUtil_TimeGetLapse()	85
7.2.2.19 LgsxUtil_TimeGetLapseStr()	86
7.2.2.20 LgsxUtil_TimeStart()	86
7.2.2.21 LgsxUtil_W2A()	86
7.3 Application Functions	87
7.3.1 Detailed Description	87
7.3.2 Function Documentation	87
7.3.2.1 Lgsx_Initialize()	87
7.3.2.2 Lgsx_Uninitialize()	87
7.4 Licensing Functions	88
7.4.1 Detailed Description	88

7.4.2 Function Documentation	88
7.4.2.1 Lgsx_InitLicense()	88
7.4.2.2 Lgsx_InitLicenseEx()	89
7.4.2.3 Lgsx_InitLicenseEx2()	89
7.4.2.4 Lgsx_IsLicensed()	90
7.4.2.5 Lgsx_IsLicensedExA()	90
7.4.2.6 Lgsx_LicenseDaysLeft()	90
7.4.2.7 Lgsx_LicenseDaysLeftExA()	90
7.4.2.8 Lgsx_LoadLicense()	91
7.4.2.9 Lgsx_LoadLicenseExA()	91
7.4.2.10 Lgsx_UnloadLicense()	91
7.4.2.11 Lgsx_UnloadLicenseExA()	91
7.5 Metadata Functions	92
7.5.1 Detailed Description	93
7.5.2 Function Documentation	93
7.5.2.1 Lgsx_MetaAddBool()	93
7.5.2.2 Lgsx_MetaAddDouble()	93
7.5.2.3 Lgsx_MetaAddDouble3()	94
7.5.2.4 Lgsx_MetaAddDouble4()	94
7.5.2.5 Lgsx_MetaAddInt()	95
7.5.2.6 Lgsx_MetaAddInt64()	95
7.5.2.7 Lgsx_MetaAddMetadata()	96
7.5.2.8 Lgsx_MetaAddString()	96
7.5.2.9 Lgsx_MetaClone()	97
7.5.2.10 Lgsx_MetaCreateInstance()	97
7.5.2.11 Lgsx_MetaGetBool()	97
7.5.2.12 Lgsx_MetaGetDouble()	98
7.5.2.13 Lgsx_MetaGetDouble3()	98
7.5.2.14 Lgsx_MetaGetDouble4()	99
7.5.2.15 Lgsx_MetaGetInt()	99
7.5.2.16 Lgsx_MetaGetInt64()	100
7.5.2.17 Lgsx_MetaGetMetadata()	100
7.5.2.18 Lgsx_MetaGetString()	101
7.5.2.19 Lgsx_MetaGetString2()	101
7.5.2.20 Lgsx_MetaHas()	102
7.5.2.21 Lgsx_MetaMoveNext()	102
7.5.2.22 Lgsx_MetaMoveNext2()	103
7.5.2.23 Lgsx_MetaRemove()	104
7.5.2.24 Lgsx_MetaReset()	104
8 Data Structure Documentation	105
8.1 CategoryEntryHandle Struct Reference	105

8.1.1 Detailed Description	105
8.2 CategoryHandle Struct Reference	105
8.2.1 Detailed Description	105
8.3 CategoryValueHandle Struct Reference	105
8.3.1 Detailed Description	105
8.4 CYASSET Struct Reference	106
8.4.1 Detailed Description	106
8.4.2 Field Documentation	106
8.4.2.1 filename	106
8.4.2.2 name	106
8.4.2.3 type	107
8.4.2.4 url	107
8.5 CYBBOX Struct Reference	107
8.5.1 Detailed Description	107
8.6 CYGEOTAG Struct Reference	107
8.6.1 Detailed Description	108
8.6.2 Field Documentation	108
8.6.2.1 setupId	108
8.7 CYMODELINFO Struct Reference	108
8.7.1 Detailed Description	109
8.7.2 Field Documentation	109
8.7.2.1 sceneRepld	109
8.8 CYMODELNODEINFO Struct Reference	109
8.8.1 Detailed Description	110
8.9 CYRANGECUBE Struct Reference	110
8.9.1 Detailed Description	110
8.10 CYRECT Struct Reference	110
8.10.1 Detailed Description	110
8.11 CYRGBA Struct Reference	111
8.11.1 Detailed Description	111
8.12 CYSETUPCAMERAINFO Struct Reference	111
8.12.1 Detailed Description	112
8.13 CYSETUPINFO Struct Reference	112
8.13.1 Detailed Description	112
8.13.2 Field Documentation	112
8.13.2.1 time	112
8.13.2.2 vertexIndex	112
8.14 CYSITEMAP Struct Reference	113
8.14.1 Detailed Description	113
8.15 CYSITEMAPIMAGE Struct Reference	113
8.15.1 Detailed Description	114
8.16 CYTARGETINFO Struct Reference	114

8.16.1 Detailed Description	114
8.17 CYTRAJECTORYINFO Struct Reference	114
8.17.1 Detailed Description	115
8.17.2 Field Documentation	115
8.17.2.1 startTime	115
8.17.2.2 stopTime	115
8.18 CYTRAJECTORYVERTEX Struct Reference	115
8.18.1 Detailed Description	115
8.18.2 Field Documentation	116
8.18.2.1 setupIndex	116
8.18.2.2 time	116
8.19 FieldEntryHandle Struct Reference	116
8.19.1 Detailed Description	116
8.20 FieldHandle Struct Reference	116
8.20.1 Detailed Description	116
8.21 GeoTagHandle Struct Reference	116
8.21.1 Detailed Description	117
8.22 M3DDUALFISHEYE Struct Reference	117
8.22.1 Detailed Description	117
8.23 M3DFLAT Struct Reference	117
8.23.1 Detailed Description	117
8.24 M3DLUTCAMERA Struct Reference	118
8.24.1 Detailed Description	118
8.25 M3DPINHOLE Struct Reference	118
8.25.1 Detailed Description	119
8.25.2 Member Enumeration Documentation	119
8.25.2.1 anonymous enum	119
8.26 PNT2D Struct Reference	119
8.26.1 Detailed Description	120
8.27 PNT3D Struct Reference	120
8.27.1 Detailed Description	120
8.28 QUA4D Struct Reference	120
8.28.1 Detailed Description	121
8.29 SCyRgdBdyTxf Struct Reference	121
8.29.1 Detailed Description	121
9 File Documentation	123
9.1 /LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/LgsxClient.h File Reference	123
9.1.1 Detailed Description	130
9.1.2 Enumeration Type Documentation	130
9.1.2.1 ImagePixelType	130
9.1.2.2 M3DPOSETYPE	130

9.1.2.3 SitemapImageType	130
9.2 /LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/LgsxReader.h File Reference	131
9.2.1 Detailed Description	137
Index	139

Chapter 1

Leica LGSx SDK Getting Started Guide

1.1 Overview

This guide is intended for Windows and Linux developers. Each LGSx SDK release includes one or more sample programs that demonstrate the available API functions.

1.2 Windows Integration

The following steps outline the process for utilizing the LGSx SDK:

1. Extract the provided .zip file containing the LGSx SDK. The extracted file structure should appear as follows:

```
\acknowledgements
\bin
\doc
\include
\lib
\samples
VC_redist.x64.exe
```

After extraction, install VC_redist.x64.

2. The full SDK documentation is located in the \doc folder.
3. A sample program executable, LgsxReaderExample.exe, is located in the \bin folder. This sample can be used to output the contents of an LGSx file. A sample LGSx file, testMobile.lgsx, is included in the \samples folder. To test the LgsxReaderExample.exe, use the following command:

```
LgsxReaderExample -i ../samples/testMobile.lgsx
```

4. The following steps describe how to build and run the provided sample programs from source code. Visual Studio 2022 and CMake 3.16 (or newer) are required.

From the command line, starting in the extracted LGSx SDK root folder:

```
cd samples
cmake -S . -B ../msvc2022_64
```

This command generates a Visual Studio solution file in a new targets folder located at samples\msvc2022_64.

```
cd msvc2022_64
LgsxSdk-Examples.sln
```

This command navigates to the new targets folder and launches Visual Studio with the newly created solution. The sample program can be built and run from within Visual Studio.

1.3 Linux Integration

1. Extract the provided LGSx SDK package:

```
sudo tar -xvzf <LGSx SDK Package>.tar.gz
```

The extracted directory structure should appear as follows:

```
/acknowledgements
/bin
/doc
/include
/lib
/samples
```

2. The full SDK documentation is located in the doc folder.
3. The LGSx package includes a sample application, LgsxReaderExample, which outputs the contents of an LGSx file. Both the source and executable for provided examples can be found in the SDK. The following steps outline how to run the executable, starting in the LGSx SDK root directory:

```
cd bin
./LgsxReaderExample -i ../samples/testMobile.lgsx
```

4. To build provided examples from source code, gcc 11 and CMake 3.16` (or newer) are required.

A. Setting up the environment

The required development tools can be installed using the following commands:

```
sudo apt-get update
sudo apt-get install --no-install-recommends -y \
    build-essential \
    gcc-multilib g++-multilib \
    cmake ninja-build
```

Install the necessary library dependencies using:

```
sudo apt-get install --no-install-recommends -y \
    libgomp1 libcurl4 libgl1 libglu1
```

B. Building and running examples

Use the following steps to build and run provided examples (starting in the LGSx SDK root directory):

```
cd samples cmake -S . -B ./build cd build cmake --build .
```

This process creates a targets directory samples/build and builds provided examples executables there.

To test LgsxReaderExample, use the same command as previously:

```
./LgsxReaderExample -i ../testMobile.lgsx
```

1.4 WSL Integration

The following steps outline the process for utilizing the LGSx SDK under Windows Subsystem for Linux (WSL):

1. Enable "Virtualization in BIOS."
2. Enable Hyper-V in Windows.
3. Install WSL. Use the following link to install the supported Ubuntu release: [Ubuntu 22.04.5 LTS - Free download and install on Windows | Microsoft Store](#)

1.5 SDK Know-How

1. Most SDK functionality is located in the include/LgsxSdk/LgsxReader.h header file. This file should be included in the application.
2. The LgsxReaderExample main function provides a standard workflow (Lgsx_Initialize, Lgsx_↵ InitLicense, ..., Lgsx_Uninitialize). Additional details can be found in the SDK documentation.

Chapter 2

Release notes

- [Release Notes 2025.0.1](#) (November 12, 2025)
- [Release Notes 2025.0.0](#) (October 1, 2025)
- [Release Notes 2024.0.1](#) (April 22, 2024)
- [Release Notes 2024.0.0](#) (February 2, 2024)

2.1 Release Notes 2025.0.1

Product	Leica LGSx SDK 2025.0.1 (patch release)
Date	November 12, 2025
Developed by	Reality Capture Software Product Management

2.1.1 Changelog

2.1.1.1 Added

- Function `Lgsx_GetMigratedFieldGuid` to return GUIDs of migrated fields `Category`, `Label`, `TagIndex`

2.1.1.2 Fixed

- Migration (automatically updates the database schema when opening files created with older software versions): Attachments now preserve their UUIDs and creation times, `LinkRef` in `GeoTags` correctly handles both external URLs and local references, and `GeoTags` modification times remain unchanged
- `LgsxReaderExample` (example application): Fixes related to non-ASCII characters handling

2.1.1.3 Other

- The Linux flavor of library no longer exports dependency symbols. This ensures a cleaner symbol table and prevents unintentional symbol exposure to avoid dependencies conflicts.

2.1.2 Additional Notes

Existing owners of LGS files will need to convert them to the LGSx format before opening or importing the LGSx content. The Leica LGS Converter Tool is available online (free of charge without any license).

2.1.3 Packaging

1. Windows Package (ZIP) - Made for Windows 10 and 11 (64-bit) using Visual Studio 2022 (19.43.34809.0).
2. Linux Package (TGZ) - Made for Ubuntu 22.04 LTS version (available on the Microsoft App Store) using gcc11.

Both packages include all applicable libraries, header and source files, sample programs, LGSx dataset, documentation (release notes, API documentation, Getting Started Guide).

Only the new LGSx format is supported. LGSx allows access to HSPC point clouds and other datasets.

The latest documentation is available on the [Reality Capture SDK site](#).

2.1.4 Licensing

The LGSx SDK is available only to partners who have signed up for the Geosystems Partners Network (GPN).

A developer license key will be provided to enable the API, allowing Licensee developers to access the available functionality. The developer key must not be distributed with the integrated product.

A deployment license key will be provided to enable the API in the distributed integrated product.

The Licensee must have the LGSx SDK installed to start coding against the API.

2.2 Release Notes 2025.0.0

Product	Leica LGSx SDK 2025.0.0 (added support for the updated GeoTag data scheme)
Date	October 1, 2025
Developed by	Reality Capture Software Product Management

2.2.1 Changelog

2.2.1.1 Added

- Additional documentation for multiple API elements, including data types and functions.
- `LgsxUtil_StrDupA()`/`LgsxUtil_StrDupW()` functions for duplicating strings.
- **Experimental** support for sitemap images (status: work in progress), including: `SitemapImageType`, `CYSITEMAP`, `CYSITEMAPIMAGE`, `Lgsx_ReaderMoveNextSitemap()`, `Lgsx_ReaderGetSitemap()`, `Lgsx_ReaderGetSitemapImage()`, `Lgsx_ReaderEnumSetupsForSitemap()`, `Lgsx_ReaderMoveNextSetupForSitemap()`.

- Versions of multiple functions that duplicate strings to avoid truncating the output:
`Lgsx_ReaderGetLastError2()`, `Lgsx_ReaderGetProjectDescription2()`,
`Lgsx_ReaderGetImageLayerNames2()`, `Lgsx_GetAssetAsImage2()`,
`Lgsx_ReaderMoveNextCoordSystems2()`.
- Function to check if a file is password-protected: `Lgsx_ReaderHasPassword()`.
- Support for retrieving point cloud points in strict order (if possible) via `Lgsx_ReaderEnumPointsEx()`.
- Project Assets extended with mime-type, url and id attributes.
- New API for reading the Assets: `Lgsx_AssetReadImage()`, `Lgsx_AssetReadBinary()`,
`Lgsx_ReaderAssetGetGuid()`, `Lgsx_AssetHasType()`.
- New API for reading the Setups: `Lgsx_ReaderGetSetupGuid()`.
- New API for GeoTags: `Lgsx_ReaderGetGeoTag()`, `Lgsx_GeoTagRelease()`,
`Lgsx_GeoTagGetGuid()`, `Lgsx_GeoTagGetName()`, `Lgsx_GeoTagHasDescription()`,
`Lgsx_GeoTagGetDescription()`, `Lgsx_GeoTagGetPosition()`,
`Lgsx_GeoTagGetCameraPosition()`, `Lgsx_GeoTagGetSetupIndex()`,
`Lgsx_GeoTagGetMetadata()`, `Lgsx_GeoTagGetThumbnail()`,
`Lgsx_GeoTagEnumAssets()`, `Lgsx_ReaderGetGeoTagAsset()`,
`Lgsx_ReaderGetGeoTagAssetThumbnail()`, `Lgsx_GeoTagHasThumbnail()`,
`Lgsx_GeoTagEnumCategoryEntries()`, `Lgsx_GeoTagGetCategoryEntry()`,
`Lgsx_GeoTagEnumFieldEntries()`, `Lgsx_GeoTagGetFieldEntry()`,
`Lgsx_GeoTagGetRelativePosition()`, `Lgsx_GeoTagGetCameraRelativePosition()`.
- Deprecated API for Geotags: `Lgsx_ReaderGetGeoTagName()`,
`Lgsx_ReaderHasGeoTagDescription()`, `Lgsx_ReaderGetGeoTagDescription()`.
- New API for Categories: `Lgsx_ReaderEnumCategories()`, `Lgsx_ReaderGetCategory()`,
`Lgsx_CategoryRelease()`, `Lgsx_CategoryGetGuid()`, `Lgsx_CategoryGetName()`,
`Lgsx_CategoryGetPriority()`, `Lgsx_CategoryEnumValues()`,
`Lgsx_CategoryGetValue()`, `Lgsx_CategoryValueGetValue()`,
`Lgsx_CategoryValueGetPriority()`, `Lgsx_CategoryValueHasColor()`,
`Lgsx_CategoryValueGetColor()`, `Lgsx_CategoryEntryGetCategory()`,
`Lgsx_CategoryEntryHasValue()`, `Lgsx_CategoryEntryGetValue()`,
`Lgsx_CategoryValueGetGuid()`.
- New API for Fields: `Lgsx_ReaderEnumFields()`, `Lgsx_ReaderGetField()`,
`Lgsx_FieldRelease()`, `Lgsx_FieldGetGuid()`, `Lgsx_FieldGetName()`,
`Lgsx_FieldGetPriority()`, `Lgsx_FieldEntryGetField()`,
`Lgsx_FieldEntryGetValue()`.
- New API for metadata: enumerate without name truncation `Lgsx_MetaMoveNext2()`, getting string value without truncation `Lgsx_MetaGetString2()`.
- Fields: id, creation and modification timestamp to [CYASSET](#).
- Fields: cameraPosition, mimeType, url, id, creation and modification timestamp to [CYGEOTAG](#).
- New API for getting creation/modification timestamps: `Lgsx_GeoTagGetCreationTimestamp()`,
`Lgsx_GeoTagGetModificationTimestamp()`, `Lgsx_FieldGetCreationTimestamp()`,
`Lgsx_FieldGetModificationTimestamp()`,
`Lgsx_CategoryGetCreationTimestamp()`,
`Lgsx_CategoryGetModificationTimestamp()`,
`Lgsx_CategoryValueGetCreationTimestamp()`,
`Lgsx_CategoryValueGetModificationTimestamp()`.

2.2.1.2 Changed

- Functions for metadata (`Lgsx_Meta*`) accepts also longer names. No API changes required due to C array decay.

2.2.1.3 Deprecated

- `Lgsx_ReaderEnumSiteMaps()` renamed to `Lgsx_ReaderEnumSitemaps()`.
- `Lgsx_ReaderMoveNextSiteMapImage()` renamed to `Lgsx_ReaderMoveNextSitemapImage()`.
- `CYGEOTAG` replaced with the new API for GeoTags.
- `Lgsx_ReaderMoveNextGeoTag()` replaced with `Lgsx_ReaderGetGeoTag()`.
- `Lgsx_MetaMoveNext()` replaced with `Lgsx_MetaMoveNext2()`.

2.2.1.4 Removed

- Multiple unused type definitions: `AnyHandle`, `GeomPtr`, `FilterPtr`, `QueryPtr`, `ClusterPtr`, `ObjectListPtr`.

2.2.1.5 Fixed

- Implemented missing functions in Linux packages: `Lgsx_GetProductVersion()`, `Lgsx_InitProductVersion()`, `Lgsx_InitProductVersionEx()`, `Lgsx_InitProductName()`, `Lgsx_InitProductBuildNumber()`.
- Issues with handling files without point clouds.

2.2.1.6 Other

- Headers now include Doxygen-formatted documentation.
- Various cleanup and fixes related to UTF character handling.

2.2.2 Additional Notes

Existing owners of LGS files will need to convert them to the LGSx format before opening or importing the LGSx content. The Leica LGS Converter Tool is available online (free of charge without any license).

2.2.3 Packaging

1. Windows Package (ZIP) - Made for Windows 10 and 11 (64-bit) using Visual Studio 2022 (19.43.34809.0).
2. Linux Package (TGZ) - Made for Ubuntu 22.04 LTS version (available on the Microsoft App Store) using gcc11.

Both packages include all applicable libraries, header and source files, sample programs, LGSx dataset, documentation (release notes, API documentation, Getting Started Guide).

Only the new LGSx format is supported. LGSx allows access to HSPC point clouds and other datasets.

The latest documentation is available on the [Reality Capture SDK site](#).

2.2.4 Licensing

The LGSx SDK is available only to partners who have signed up for the Geosystems Partners Network (GPN).

A developer license key will be provided to enable the API, allowing Licensee developers to access the available functionality. The developer key must not be distributed with the integrated product.

A deployment license key will be provided to enable the API in the distributed integrated product.

The Licensee must have the LGSx SDK installed to start coding against the API.

2.3 Release Notes 2024.0.1

Product	Leica LGSx SDK 2024.0.1 (Minor patch release)
Date	April 22, 2024
Developed by	Reality Capture Software Product Management

2.3.1 What's New

- Evaluation Read-Only license key is valid until July 1, 2024.
- The 'Samples' folder was updated with another example program.
- The release is targeting integrators who are interested in open/import LGSx files only. This release is provided in two distributions:

2.3.2 Packaging

1. Windows InstallShield
2. Linux Package (tgz) - Made for Ubuntu 20.04.6 LTS version (Available on Microsoft App Store).

Both include all the applicable libs, header and source files, sample programs, LGSx dataset, release notes, API documentation, and Getting Started Guide.

Only the new LGSx format is supported. LGSx allows access to HSPC point clouds and other datasets.

Latest documentations are available on [Reality Capture SDK site](#).

2.3.3 Fixed Issues

Several small fixes were made to allow more LGSx file types to be opened/imported.

2.3.4 Known Issues

Existing owners of LGS files will need to convert to LGSx format prior to opening/importing the LGSx content. Leica LGS Converter Tool is available online (Free of charge w/o any license).

2.3.5 Licensing

The LGSx SDK is available only to partners who signed up for the Geosystems Partners Network (GPN).

A developer license key will be provided to enable the API so that Licensee developers can access the available functionality in the API. The developer key shall not be distributed with the Integrated Product.

A deployment license key will be provided to enable the API in the distributed Integrated Product.

The Licensee will need to have the LGSx SDK installed to start coding against the API.

2.4 Release Notes 2024.0.0

Product	Leica LGSx SDK 2024.0 (First Release to 3rd Party Partners)
Date	February 2, 2024
Developed by	Reality Capture Software Product Management

2.4.1 What's New

The first release is targeting integrators who are interested in open/import LGSx files only.

2.4.2 Packaging

This release is provided in two distributions:

1. Windows InstallShield
2. Linux Package (tgz) - Made for Ubuntu 20.04.6 LTS version (Available on Microsoft App Store).

Both include all the applicable libs, header and source files, sample programs, LGSx dataset, release notes, API documentation, and Getting Started Guide.

Only the new LGSx format is supported. LGSx allows access to HSPC point clouds and other datasets.

Latest documentations are available on [Reality Capture SDK site](#).

2.4.3 Fixed Issues

None

2.4.4 Known Issues

Existing owners of LGS files will need to convert to LGSx format prior to opening/importing the LGSx content. Leica LGS Converter Tool is available online (Free of charge w/o any license).

2.4.5 Licensing

The LGSx SDK is available only to partners who signed up for the Geosystems Partners Network (GPN).

A developer license key will be provided to enable the API so that Licensee developers can access the available functionality in the API. The developer key shall not be distributed with the Integrated Product.

A deployment license key will be provided to enable the API in the distributed Integrated Product.

The Licensee will need to have the LGSx SDK installed to start coding against the API.

Chapter 3

Deprecated List

Struct [CYGEOTAG](#)

26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

Global [Lgsx_MetaMoveNext](#) ([LgsxMetadataPtr](#) pMetadata, [LgsxPropertyName](#) pName, int *pType)

09.07.2025 - [Lgsx_MetaMoveNext2](#) introduced that does not truncate the name.

Global [Lgsx_ReaderEnumSiteMaps](#) ([LgsxReaderPtr](#) reader, int *pNumSitemap)

18.04.2025 - Renamed to [Lgsx_ReaderEnumSitemaps\(\)](#)

Global [Lgsx_ReaderGetGeoTagDescription](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, wchar_t **pDescPtr)

26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

Global [Lgsx_ReaderGetGeoTagName](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, wchar_t **pNamePtr)

26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

Global [Lgsx_ReaderHasGeoTagDescription](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, bool *pStatus)

26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

Global [Lgsx_ReaderMoveNextGeoTag](#) ([LgsxReaderPtr](#) reader, uint64_t *geoTagId, [CYGEOTAG](#) *pGeoTag, [LgsxMetadataPtr](#) *pMetadata)

26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

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

18.04.2025 - Renamed to [Lgsx_ReaderMoveNextSitemapImage\(\)](#)

Chapter 4

Topic Index

4.1 Topics

Here is a list of all topics with brief descriptions:

Reader Functions	17
Basic Reader Functions	18
Project Functions	20
Asset Functions	22
Categories Functions	27
Fields Functions	36
Migrated Fields	40
Setup Functions	41
Target Functions	47
Run Functions	48
GeoTag Functions	53
Sitemap Functions	65
Model and Model Node Functions	70
Point Cloud Functions	73
User Coordinate System Functions	77
General Functions	79
Application Functions	87
Licensing Functions	88
Metadata Functions	92

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

CategoryEntryHandle	Descriptor for a Category Entry object	105
CategoryHandle	Descriptor for a Category object	105
CategoryValueHandle	Descriptor for a Category Value object	105
CYASSET	Structure for Asset	106
CYBBOX	Structure for bounding box (min, max) corners	107
CYGEOTAG	Structure for GeoTag	107
CYMODELINFO	Struct of returned Model info	108
CYMODELNODEINFO	Struct of returned ModelNode info	109
CYRANGECUBE	Structure for an arbitrary range cube (a box)	110
CYRECT	Structure for rectangle (left, top, right, bottom)	110
CYRGBA	Structure for holding the RGBA color channels	111
CYSETUPCAMERAINFO	Structure for setup camera info	111
CYSETUPINFO	Structure for setup info	112
CYSITEMAP	Structure for Sitemap	113
CYSITEMAPIMAGE	Structure for sitemap image	113
CYTARGETINFO	Structure for target info	114
CYTRAJECTORYINFO	Structure for track (a Run, Walk or Fly) header info of mobile scanning (such as BLK2GO) . . .	114
CYTRAJECTORYVERTEX	Structure for Trajectory (a Run, Walk or Fly) node of mobile scanning (such as BLK2GO) . . .	115

FieldEntryHandle	
Descriptor for a Field Entry object	116
FieldHandle	
Descriptor for a Field object	116
GeoTagHandle	
GeoTag data handle for accessing the GeoTag data	116
M3DDUALFISHEYE	
Structure for M3D dual fish eye camera model	117
M3DFLAT	
Structure for M3D flat camera model	117
M3DLUTCAMERA	
Structure for M3D LUT camera model	118
M3DPINHOLE	
Structure for M3D pinhole camera model	118
PNT2D	
Structure for 2D point (or vector, defined as 2 doubles)	119
PNT3D	
Structure for 3D point (or vector, defined as 3 doubles)	120
QUA4D	
Structure for Quaternion (or vector, defined as 4 doubles)	120
SCyRgdBdyTxf	
Rigid transformation represented by a translation and a quaternion	121

Chapter 6

File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

/LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/ LgsxClient.h	123
/LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/ LgsxReader.h	131

Chapter 7

Topic Documentation

7.1 Reader Functions

Topics

- [Basic Reader Functions](#)
- [Project Functions](#)
- [Asset Functions](#)
- [Categories Functions](#)
Categories.
- [Fields Functions](#)
Fields.
- [Migrated Fields](#)
Old Geotag's metadata "Category", "Label" and "TagIndex" are now automatically migrated to new Fields.
- [Setup Functions](#)
- [Target Functions](#)
- [Run Functions](#)
- [GeoTag Functions](#)
- [Sitemap Functions](#)
- [Model and Model Node Functions](#)
- [Point Cloud Functions](#)
- [User Coordinate System Functions](#)

Data Structures

- struct [CategoryHandle](#)
Descriptor for a Category object.
- struct [CategoryValueHandle](#)
Descriptor for a Category Value object.
- struct [CategoryEntryHandle](#)
Descriptor for a Category Entry object.
- struct [FieldHandle](#)
Descriptor for a Field object.
- struct [FieldEntryHandle](#)
Descriptor for a Field Entry object.
- struct [GeoTagHandle](#)
GeoTag data handle for accessing the GeoTag data.

Typedefs

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

7.1.1 Detailed Description

7.1.2 Basic 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.
- int [Lgsx_ReaderGetLastError2](#) ([LgsxReaderPtr](#) reader, char **pErrorPtr)
Returns a string description of the last error that occurred with the given Reader.
- int [Lgsx_ReaderHasPassword](#) (const wchar_t *pFilePath, bool *hasPassword)
Checks if file is password protected.

7.1.2.1 Detailed Description

7.1.2.2 Function Documentation

7.1.2.2.1 Lgsx_ReaderClose()

```
int Lgsx_ReaderClose (
    LgsxReaderPtr reader)
```

Closes the LGSx file for this Reader.

Parameters

in	<i>reader</i>	Reader handle.
----	---------------	----------------

Returns

0 for success.

7.1.2.2.2 Lgsx_ReaderCreate()

```
LgsxReaderPtr Lgsx_ReaderCreate (
    int * rError)
```

Create a Reader and return its handle.

Parameters

out	<i>rError</i>	Error code.
-----	---------------	-------------

Returns

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

7.1.2.2.3 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.

See also

[Lgsx_ReaderGetLastError2\(\)](#) is a version that always return complete data without truncation.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pError</i>	Error string buffer.
in	<i>maxLen</i>	Size of error string buffer.

Returns

0 for success.

7.1.2.2.4 Lgsx_ReaderGetLastError2()

```
int Lgsx_ReaderGetLastError2 (
    LgsxReaderPtr reader,
    char ** pErrorPtr)
```

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

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pErrorPtr</i>	Pointer to store pointer to error string. Buffer must be freed usign LgsxUtil_FreeMem() .

Returns

0 for success.

7.1.2.2.5 Lgsx_ReaderHasPassword()

```
int Lgsx_ReaderHasPassword (
    const wchar_t * pFilePath,
    bool * hasPassword)
```

Checks if file is password protected.

Parameters

in	<i>pFilePath</i>	Full path to the LGSx file to be opened.
out	<i>hasPassword</i>	Set to true if the file is password protected.

Returns

0 for success.

7.1.2.2.6 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

in	<i>reader</i>	Reader handle.
in	<i>pFilePath</i>	Full path to the LGSx file to be opened.
in	<i>password</i>	Password needed to open the LGSx file. Set to null if there is no password.

Returns

0 for success.

7.1.3 Project 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.
- int [Lgsx_ReaderGetProjectDescription2](#) ([LgsxReaderPtr](#) reader, wchar_t **pDescPtr)
Returns the project description string for the opened project.

7.1.3.1 Detailed Description**7.1.3.2 Function Documentation****7.1.3.2.1 Lgsx_ReaderGetMetadata()**

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

Returns the project metadata for the opened project.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pPointCount</i>	Number of points in the point cloud.
out	<i>pBbox</i>	Bounding box that contains the project.
out	<i>pMetadata</i>	Metadata object that contains additional project metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

0 for success.

7.1.3.2.2 Lgsx_ReaderGetProjectDescription()

```
int Lgsx_ReaderGetProjectDescription (
    LgsxReaderPtr reader,
    wchar_t * pDesc,
    int maxLen)
```

Returns the project description string for the opened project.

See also

[Lgsx_ReaderGetProjectDescription2\(\)](#) is a version that always return complete data without truncation.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pDesc</i>	Description string buffer.
in	<i>maxLen</i>	Size of error string buffer.

Returns

0 for success.

7.1.3.2.3 Lgsx_ReaderGetProjectDescription2()

```
int Lgsx_ReaderGetProjectDescription2 (
    LgsxReaderPtr reader,
    wchar_t ** pDescPtr)
```

Returns the project description string for the opened project.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pDescPtr</i>	Pointer to store pointer to description. Buffer must be freed usign LgsxUtil_FreeMem() .

Returns

0 for success.

7.1.3.2.4 Lgsx_ReaderGetProjectInfo()

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

Returns the project metadata and thumbnail for the opened project.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pOwnerInfo</i>	Project metadata. Metadata must be freed usign Lgsx_FreeHandle() .
out	<i>width</i>	Thumbnail image width.
out	<i>height</i>	Thumbnail image height.
out	<i>widthInBytes</i>	Thumbnail image byte-width.
out	<i>pixelType</i>	Thumbnail image pixel type.
out	<i>pThumbImage</i>	Thumbnail image data.

Returns

0 for success.

7.1.4 Asset Functions

Functions

- int [Lgsx_ReaderEnumAssets](#) (LgsxReaderPtr reader, int *pNumAsset, bool bGetAll)
Collects assets 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 current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_ReaderAssetGetGuid](#) (LgsxReaderPtr reader, const CYASSET *pAsset, char **pGuidPtr)
Returns the Asset GUID for the asset passed as parameter.
- int [Lgsx_GetAssetAsImage](#) (LgsxReaderPtr reader, uint64_t assetId, wchar_t *pName, int maxLenName, int *width, int *height, int *widthInBytes, ImagePixelFormat *pixelType, unsigned char **pImage)
Get the "payload" image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_GetAssetAsImage2](#) (LgsxReaderPtr reader, uint64_t assetId, wchar_t **pNamePtr, int *width, int *height, int *widthInBytes, ImagePixelFormat *pixelType, unsigned char **pImage)
Get the "payload" image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_AssetReadImage](#) (const CYASSET *pAsset, int *pWidth, int *pHeight, int *pWidthInBytes, ImagePixelFormat *pPixelFormat, unsigned char **pImage)
Get the "payload" as image for the given Asset.
- int [Lgsx_AssetReadBinary](#) (const CYASSET *pAsset, unsigned char **pData, int *pSize)
Get the "payload" as binary data for the given Asset.
- int [Lgsx_GetAssetThumbImage](#) (LgsxReaderPtr reader, uint64_t assetId, int *width, int *height, int *widthInBytes, ImagePixelFormat *pixelType, unsigned char **pImage)
Get the thumbnail image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_AssetHasType](#) (LgsxReaderPtr reader, uint64_t assetId, bool *pHasAssetType)
Check if an Asset has a type set.

7.1.4.1 Detailed Description

7.1.4.2 Function Documentation

7.1.4.2.1 Lgsx_AssetHasType()

```
int Lgsx_AssetHasType (
    LgsxReaderPtr reader,
    uint64_t assetId,
    bool * pHasAssetType)
```

Check if an Asset has a type set.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>assetId</i>	Asset ID.
out	<i>value</i>	that tells if an asset has the type set.

Returns

Returns 0 for success.

7.1.4.2.2 Lgsx_AssetReadBinary()

```
int Lgsx_AssetReadBinary (
    const CYASSET * pAsset,
    unsigned char ** pData,
    int * pSize)
```

Get the "payload" as binary data for the given Asset.

Parameters

in	<i>pAsset</i>	Asset info.
out	<i>pData</i>	Pointer to store pointer to binary data. Buffer is read-only.
out	<i>pSize</i>	Number of bytes in the binary data.

Returns

Returns 0 for success.

7.1.4.2.3 Lgsx_AssetReadImage()

```
int Lgsx_AssetReadImage (
    const CYASSET * pAsset,
    int * pWidth,
    int * pHeight,
    int * pWidthInBytes,
    ImagePixelType * pPixelType,
    unsigned char ** pImage)
```

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

Parameters

in	<i>pAsset</i>	Asset info.
out	<i>pWidth</i>	Image width.
out	<i>pHeight</i>	Image height.
out	<i>pWidthInBytes</i>	Image width in bytes.
out	<i>pPixelFormat</i>	Pixel type.
out	<i>pImage</i>	Asset image.

Returns

Returns 0 for success.

7.1.4.2.4 Lgsx_GetAssetAsImage()

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

Get the "payload" image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.

See also

[Lgsx_GetAssetAsImage2\(\)](#) is a version that always return complete data without truncation.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>assetId</i>	Asset ID.
out	<i>pName</i>	Asset filename buffer.
in	<i>maxLenName</i>	Asset filename buffer length.
out	<i>width</i>	Image width.
out	<i>height</i>	Image height.
out	<i>widthInBytes</i>	Image width in bytes.
out	<i>pixelType</i>	Pixel type.
out	<i>pImage</i>	Asset image.

Returns

Returns 0 for success.

7.1.4.2.5 Lgsx_GetAssetAsImage2()

```
int Lgsx_GetAssetAsImage2 (
    LgsxReaderPtr reader,
    uint64_t assetId,
    wchar_t ** pNamePtr,
    int * width,
    int * height,
    int * widthInBytes,
    ImagePixelFormat * pixelType,
    unsigned char ** pImage)
```

Get the "payload" image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>assetId</i>	Asset ID.
out	<i>pNamePtr</i>	Pointer to store pointer to asset filename. Buffer must be freed using LgsxUtil_FreeMem() .
out	<i>width</i>	Image width.
out	<i>height</i>	Image height.
out	<i>widthInBytes</i>	Image width in bytes.
out	<i>pixelType</i>	Pixel type.
out	<i>pImage</i>	Asset image.

Returns

Returns 0 for success.

7.1.4.2.6 Lgsx_GetAssetThumbImage()

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

Get the thumbnail image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>assetId</i>	Asset ID.
out	<i>width</i>	Image width.
out	<i>height</i>	Image height.
out	<i>widthInBytes</i>	Image width in bytes.
out	<i>pixelType</i>	Pixel type.
out	<i>pImage</i>	Asset thumbnail image.

Returns

Returns 0 for success.

7.1.4.2.7 Lgsx_ReaderAssetGetGuid()

```
int Lgsx_ReaderAssetGetGuid (
    LgsxReaderPtr reader,
    const CYASSET * pAsset,
    char ** pGuidPtr)
```

Returns the Asset GUID for the asset passed as parameter.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>pAsset</i>	Pointer to the asset information structure.
out	<i>pGuidPtr</i>	Pointer to store the GUID of the asset. Buffer must be freed using LgsxUtil_FreeMem() .

Returns

Returns 0 for success. Returns -1 if an error occurs.

7.1.4.2.8 Lgsx_ReaderEnumAssets()

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

Collects assets in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumAsset</i>	Number of collected Assets.
in	<i>bGetAll</i>	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.

7.1.4.2.9 Lgsx_ReaderGetAsset()

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

Get Info and metadata for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>assetId</i>	Asset ID
out	<i>pAsset</i>	Asset info.
out	<i>pMetadata</i>	Asset metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

Returns 0 for success.

7.1.4.2.10 Lgsx_ReaderMoveNextAsset()

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

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

Parameters

in	<i>reader</i>	Reader handle.
out	<i>assetId</i>	Asset ID.

Returns

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

7.1.5 Categories Functions

Categories.

Data Structures

- struct [CategoryHandle](#)
Descriptor for a Category object.
- struct [CategoryValueHandle](#)
Descriptor for a Category Value object.
- struct [CategoryEntryHandle](#)
Descriptor for a Category Entry object.

Functions

- int [Lgsx_ReaderEnumCategories](#) ([LgsxReaderPtr](#) reader, int *pNumCategories)
Collects Categories in the project and returns the count.
- int [Lgsx_ReaderGetCategory](#) ([LgsxReaderPtr](#) reader, int pos, [CategoryHandle](#) *pHandle)
Read Category data at the given position in the active collection.
- int [Lgsx_CategoryRelease](#) ([CategoryHandle](#) *pHandle)
Release the Category data pointer returned by [Lgsx_ReaderGetCategory\(\)](#).
- int [Lgsx_CategoryGetGuid](#) (const [CategoryHandle](#) handle, const char **pId)
Get the GUID of the Category.
- int [Lgsx_CategoryGetName](#) (const [CategoryHandle](#) handle, const wchar_t **pName)
Get the name of the Category.
- int [Lgsx_CategoryGetPriority](#) (const [CategoryHandle](#) handle, int *pPriority)
Get the priority of the Category.
- int [Lgsx_CategoryGetCreationTimestamp](#) (const [CategoryHandle](#) handle, uint64_t *pTimestamp)
Get the creation timestamp of the Category.
- int [Lgsx_CategoryGetModificationTimestamp](#) (const [CategoryHandle](#) handle, uint64_t *pTimestamp)
Get the modification timestamp of the Category.
- int [Lgsx_CategoryEnumValues](#) (const [CategoryHandle](#) handle, int *pNumValues)
Enumerate the values within a Category and return the count.
- int [Lgsx_CategoryGetValue](#) (const [CategoryHandle](#) handle, int pos, [CategoryValueHandle](#) *pValueHandle)
Get a Category Value at the specified position.
- int [Lgsx_CategoryValueGetGuid](#) (const [CategoryValueHandle](#) handle, const char **pId)
Get the GUID of a Category Value.
- int [Lgsx_CategoryValueGetValue](#) (const [CategoryValueHandle](#) handle, const wchar_t **pValue)
Get the value of a Category Value.
- int [Lgsx_CategoryValueGetPriority](#) (const [CategoryValueHandle](#) handle, int *pPriority)
Get the priority of a Category Value.
- int [Lgsx_CategoryValueGetCreationTimestamp](#) (const [CategoryValueHandle](#) handle, uint64_t *pTimestamp)
Get the creation timestamp of a Category Value.
- int [Lgsx_CategoryValueGetModificationTimestamp](#) (const [CategoryValueHandle](#) handle, uint64_t *pTimestamp)
Get the modification timestamp of a Category Value.
- int [Lgsx_CategoryValueHasColor](#) (const [CategoryValueHandle](#) handle, bool *pStatus)
Check if a Category Value has a color associated with it.
- int [Lgsx_CategoryValueGetColor](#) (const [CategoryValueHandle](#) handle, [CYRGBA](#) *pColor)
Get the color of a Category Value.
- int [Lgsx_CategoryEntryGetCategory](#) (const [CategoryEntryHandle](#) handle, [CategoryHandle](#) *pCategoryHandle)
Get the Category from a Category Entry.
- int [Lgsx_CategoryEntryHasValue](#) (const [CategoryEntryHandle](#) handle, bool *pStatus)
Check if a Category Entry has a value associated with it.
- int [Lgsx_CategoryEntryGetValue](#) (const [CategoryEntryHandle](#) handle, [CategoryValueHandle](#) *pValueHandle)
Get the Category Value from a Category Entry.

7.1.5.1 Detailed Description

Categories.

7.1.5.2 Function Documentation

7.1.5.2.1 Lgsx_CategoryEntryGetCategory()

```
int Lgsx_CategoryEntryGetCategory (
    const CategoryEntryHandle handle,
    CategoryHandle * pCategoryHandle)
```

Get the Category from a Category Entry.

Parameters

in	<i>handle</i>	Category Entry handle.
out	<i>pCategoryHandle</i>	Pointer to store the Category handle. Returns read-only pointer bound to lifetime of CategoryEntryHandle , shall not be released manually.

Returns

Returns 0 for success.

7.1.5.2.2 Lgsx_CategoryEntryGetValue()

```
int Lgsx_CategoryEntryGetValue (
    const CategoryEntryHandle handle,
    CategoryValueHandle * pValueHandle)
```

Get the Category Value from a Category Entry.

Parameters

in	<i>handle</i>	Category Entry handle.
out	<i>pValueHandle</i>	Pointer to store the Category Value handle. Returns read-only pointer bound to lifetime of CategoryEntryHandle , shall not be released manually. Might return null pointer if the Category Entry does not have a value associated with it.

Returns

Returns 0 for success.

7.1.5.2.3 Lgsx_CategoryEntryHasValue()

```
int Lgsx_CategoryEntryHasValue (
    const CategoryEntryHandle handle,
    bool * pStatus)
```

Check if a Category Entry has a value associated with it.

Parameters

in	<i>handle</i>	Category Entry handle.
out	<i>pStatus</i>	Pointer to store the status.

Returns

Returns 0 for success.

7.1.5.2.4 Lgsx_CategoryEnumValues()

```
int Lgsx_CategoryEnumValues (
    const CategoryHandle handle,
    int * pNumValues)
```

Enumerate the values within a Category and return the count.

Parameters

in	<i>handle</i>	Category handle.
out	<i>pNumValues</i>	Pointer to store the number of values in the Category.

Returns

Returns 0 for success.

7.1.5.2.5 Lgsx_CategoryGetCreationTimestamp()

```
int Lgsx_CategoryGetCreationTimestamp (
    const CategoryHandle handle,
    uint64_t * pTimestamp)
```

Get the creation timestamp of the Category.

Parameters

in	<i>handle</i>	Category handle.
out	<i>pTimestamp</i>	Pointer to store the creation timestamp of the Category.

Returns

Returns 0 for success.

7.1.5.2.6 Lgsx_CategoryGetGuid()

```
int Lgsx_CategoryGetGuid (
    const CategoryHandle handle,
    const char ** pId)
```

Get the GUID of the Category.

Parameters

in	<i>handle</i>	Category handle.
out	<i>pId</i>	Pointer to store the GUID string of the Category.

Returns

Returns 0 for success.

7.1.5.2.7 Lgsx_CategoryGetModificationTimestamp()

```
int Lgsx_CategoryGetModificationTimestamp (  
    const CategoryHandle handle,  
    uint64_t * pTimestamp)
```

Get the modification timestamp of the Category.

Parameters

in	<i>handle</i>	Category handle.
out	<i>pTimestamp</i>	Pointer to store the modification timestamp of the Category.

Returns

Returns 0 for success.

7.1.5.2.8 Lgsx_CategoryGetName()

```
int Lgsx_CategoryGetName (  
    const CategoryHandle handle,  
    const wchar_t ** pName)
```

Get the name of the Category.

Parameters

in	<i>handle</i>	Category handle.
out	<i>pName</i>	Pointer to store the name of the Category.

Returns

Returns 0 for success.

7.1.5.2.9 Lgsx_CategoryGetPriority()

```
int Lgsx_CategoryGetPriority (  
    const CategoryHandle handle,  
    int * pPriority)
```

Get the priority of the Category.

Parameters

in	<i>handle</i>	Category handle.
out	<i>pPriority</i>	Pointer to store the priority value of the Category.

Returns

Returns 0 for success.

7.1.5.2.10 Lgsx_CategoryGetValue()

```
int Lgsx_CategoryGetValue (
    const CategoryHandle handle,
    int pos,
    CategoryValueHandle * pValueHandle)
```

Get a Category Value at the specified position.

Parameters

in	<i>handle</i>	Category handle.
in	<i>pos</i>	Position in the Category's value collection.
out	<i>pValueHandle</i>	Pointer to store the Category Value handle. Returns read-only pointer bound to lifetime of CategoryHandle , shall not be released manually.

Returns

Returns 0 for success. Returns -1 when position is out of range.

7.1.5.2.11 Lgsx_CategoryRelease()

```
int Lgsx_CategoryRelease (
    CategoryHandle * pHandle)
```

Release the Category data pointer returned by [Lgsx_ReaderGetCategory\(\)](#).

Parameters

in	<i>pHandle</i>	Pointer to Category data
----	----------------	--------------------------

Returns

Returns 0 for success.

7.1.5.2.12 Lgsx_CategoryValueGetColor()

```
int Lgsx_CategoryValueGetColor (
    const CategoryValueHandle handle,
    CYRGBA * pColor)
```

Get the color of a Category Value.

Parameters

in	<i>handle</i>	Category Value handle.
out	<i>pColor</i>	Pointer to store the RGBA color of the Category Value.

Returns

Returns 0 for success.

7.1.5.2.13 Lgsx_CategoryValueGetCreationTimestamp()

```
int Lgsx_CategoryValueGetCreationTimestamp (  
    const CategoryValueHandle handle,  
    uint64_t * pTimestamp)
```

Get the creation timestamp of a Category Value.

Parameters

in	<i>handle</i>	Category Value handle.
out	<i>pTimestamp</i>	Pointer to store the creation timestamp of the Category Value.

Returns

Returns 0 for success.

7.1.5.2.14 Lgsx_CategoryValueGetGuid()

```
int Lgsx_CategoryValueGetGuid (  
    const CategoryValueHandle handle,  
    const char ** pId)
```

Get the GUID of a Category Value.

Parameters

in	<i>handle</i>	Category Value handle.
out	<i>pId</i>	Pointer to store the ID string of the Category Value.

Returns

Returns 0 for success.

7.1.5.2.15 Lgsx_CategoryValueGetModificationTimestamp()

```
int Lgsx_CategoryValueGetModificationTimestamp (  
    const CategoryValueHandle handle,  
    uint64_t * pTimestamp)
```

Get the modification timestamp of a Category Value.

Parameters

in	<i>handle</i>	Category Value handle.
out	<i>pTimestamp</i>	Pointer to store the modification timestamp of the Category Value.

Returns

Returns 0 for success.

7.1.5.2.16 Lgsx_CategoryValueGetPriority()

```
int Lgsx_CategoryValueGetPriority (  
    const CategoryValueHandle handle,  
    int * pPriority)
```

Get the priority of a Category Value.

Parameters

in	<i>handle</i>	Category Value handle.
out	<i>pPriority</i>	Pointer to store the priority value of the Category Value.

Returns

Returns 0 for success.

7.1.5.2.17 Lgsx_CategoryValueGetValue()

```
int Lgsx_CategoryValueGetValue (  
    const CategoryValueHandle handle,  
    const wchar_t ** pValue)
```

Get the value of a Category Value.

Parameters

in	<i>handle</i>	Category Value handle.
out	<i>pValue</i>	Pointer to store the value string of the Category Value.

Returns

Returns 0 for success.

7.1.5.2.18 Lgsx_CategoryValueHasColor()

```
int Lgsx_CategoryValueHasColor (  
    const CategoryValueHandle handle,  
    bool * pStatus)
```

Check if a Category Value has a color associated with it.

Parameters

in	<i>handle</i>	Category Value handle.
out	<i>pStatus</i>	Pointer to store the status. Set to true if the Category Value has a color associated with it.

Returns

Returns 0 for success.

7.1.5.2.19 Lgsx_ReaderEnumCategories()

```
int Lgsx_ReaderEnumCategories (  
    LgsxReaderPtr reader,  
    int * pNumCategories)
```

Collects Categories in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumCategories</i>	Number of collected Categories.

Returns

Returns 0 for success.

7.1.5.2.20 Lgsx_ReaderGetCategory()

```
int Lgsx_ReaderGetCategory (  
    LgsxReaderPtr reader,  
    int pos,  
    CategoryHandle * pHandle)
```

Read Category data at the given position in the active collection.

Active collection is set by [Lgsx_ReaderEnumCategories\(\)](#)

Parameters

in	<i>reader</i>	Reader handle.
in	<i>pos</i>	Position in the active collection.
out	<i>pHandle</i>	Pointer to store pointer to Category data. The pointer has to be released using Lgsx_CategoryRelease()

Returns

Returns 0 for success. Returns -1 when position is out of range.

7.1.6 Fields Functions

Fields.

Data Structures

- struct [FieldHandle](#)
Descriptor for a Field object.
- struct [FieldEntryHandle](#)
Descriptor for a Field Entry object.

Functions

- int [Lgsx_ReaderEnumFields](#) ([LgsxReaderPtr](#) reader, int *pNumFields)
Collects Fields in the project and returns the count.
- int [Lgsx_ReaderGetField](#) ([LgsxReaderPtr](#) reader, int pos, [FieldHandle](#) *pHandle)
Read Field data at the given position in the active collection.
- int [Lgsx_FieldRelease](#) ([FieldHandle](#) *pHandle)
Release the Field data pointer returned by [Lgsx_ReaderGetField\(\)](#).
- int [Lgsx_FieldGetGuid](#) (const [FieldHandle](#) handle, const char **pId)
Get the GUID of the Field.
- int [Lgsx_FieldGetName](#) (const [FieldHandle](#) handle, const wchar_t **pName)
Get the name of the Field.
- int [Lgsx_FieldGetPriority](#) (const [FieldHandle](#) handle, int *pPriority)
Get the priority of the Field.
- int [Lgsx_FieldGetCreationTimestamp](#) (const [FieldHandle](#) handle, uint64_t *pTimestamp)
Get the creation timestamp of the Field.
- int [Lgsx_FieldGetModificationTimestamp](#) (const [FieldHandle](#) handle, uint64_t *pTimestamp)
Get the modification timestamp of the Field.
- int [Lgsx_FieldEntryGetField](#) (const [FieldEntryHandle](#) handle, [FieldHandle](#) *pFieldHandle)
Get the Field from a Field Entry.
- int [Lgsx_FieldEntryGetValue](#) (const [FieldEntryHandle](#) handle, const wchar_t **pValue)
Get the value from a Field Entry.

7.1.6.1 Detailed Description

Fields.

7.1.6.2 Function Documentation

7.1.6.2.1 Lgsx_FieldEntryGetField()

```
int Lgsx_FieldEntryGetField (
    const FieldEntryHandle handle,
    FieldHandle * pFieldHandle)
```

Get the Field from a Field Entry.

Parameters

in	<i>handle</i>	Field Entry handle.
out	<i>pFieldHandle</i>	Pointer to store the Field handle. Returns read-only pointer bound to lifetime of FieldEntryHandle , shall not be released manually.

Returns

Returns 0 for success.

7.1.6.2.2 Lgsx_FieldEntryGetValue()

```
int Lgsx_FieldEntryGetValue (  
    const FieldEntryHandle handle,  
    const wchar_t ** pValue)
```

Get the value from a Field Entry.

Parameters

in	<i>handle</i>	Field Entry handle.
out	<i>pValue</i>	Pointer to store the value string of the Field Entry.

Returns

Returns 0 for success.

7.1.6.2.3 Lgsx_FieldGetCreationTimestamp()

```
int Lgsx_FieldGetCreationTimestamp (  
    const FieldHandle handle,  
    uint64_t * pTimestamp)
```

Get the creation timestamp of the Field.

Parameters

in	<i>handle</i>	Field handle.
out	<i>pTimestamp</i>	Pointer to store the creation timestamp of the Field.

Returns

Returns 0 for success.

7.1.6.2.4 Lgsx_FieldGetGuid()

```
int Lgsx_FieldGetGuid (  
    const FieldHandle handle,  
    const char ** pId)
```

Get the GUID of the Field.

Parameters

in	<i>handle</i>	Field handle.
out	<i>pId</i>	Pointer to store the GUID string of the Field.

Returns

Returns 0 for success.

7.1.6.2.5 Lgsx_FieldGetModificationTimestamp()

```
int Lgsx_FieldGetModificationTimestamp (  
    const FieldHandle handle,  
    uint64_t * pTimestamp)
```

Get the modification timestamp of the Field.

Parameters

in	<i>handle</i>	Field handle.
out	<i>pTimestamp</i>	Pointer to store the modification timestamp of the Field.

Returns

Returns 0 for success.

7.1.6.2.6 Lgsx_FieldGetName()

```
int Lgsx_FieldGetName (  
    const FieldHandle handle,  
    const wchar_t ** pName)
```

Get the name of the Field.

Parameters

in	<i>handle</i>	Field handle.
out	<i>pName</i>	Pointer to store the name of the Field.

Returns

Returns 0 for success.

7.1.6.2.7 Lgsx_FieldGetPriority()

```
int Lgsx_FieldGetPriority (  
    const FieldHandle handle,  
    int * pPriority)
```

Get the priority of the Field.

Parameters

in	<i>handle</i>	Field handle.
out	<i>pPriority</i>	Pointer to store the priority value of the Field.

Returns

Returns 0 for success.

7.1.6.2.8 Lgsx_FieldRelease()

```
int Lgsx_FieldRelease (  
    FieldHandle * pHandle)
```

Release the Field data pointer returned by [Lgsx_ReaderGetField\(\)](#).

Parameters

in	<i>pHandle</i>	Pointer to Field data
----	----------------	-----------------------

Returns

Returns 0 for success.

7.1.6.2.9 Lgsx_ReaderEnumFields()

```
int Lgsx_ReaderEnumFields (  
    LgsxReaderPtr reader,  
    int * pNumFields)
```

Collects Fields in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumFields</i>	Number of collected Fields.

Returns

Returns 0 for success.

7.1.6.2.10 Lgsx_ReaderGetField()

```
int Lgsx_ReaderGetField (  
    LgsxReaderPtr reader,  
    int pos,  
    FieldHandle * pHandle)
```

Read Field data at the given position in the active collection.

Active collection is set by [Lgsx_ReaderEnumFields\(\)](#)

Parameters

in	<i>reader</i>	Reader handle.
in	<i>pos</i>	Position in the active collection.
out	<i>pHandle</i>	Pointer to store pointer to Field data. The pointer has to be released using Lgsx_FieldRelease()

Returns

Returns 0 for success. Returns -1 when position is out of range.

7.1.7 Migrated Fields

Old Geotag's metadata "Category", "Label" and "TagIndex" are now automatically migrated to new Fields.

Enumerations

- enum [LgsxMigratedMetaKeyField](#) {
[LgsxMigratedMetaKeyField_Category](#) = 0 ,
[LgsxMigratedMetaKeyField_Label](#) ,
[LgsxMigratedMetaKeyField_TagIndex](#) }

Enum for the migrated metadata.

Functions

- int [Lgsx_GetMigratedFieldGuid](#) ([LgsxMigratedMetaKeyField](#) field, const char **pId)

Get the GUID of the migrated Field specified by "category" argument.

7.1.7.1 Detailed Description

Old Geotag's metadata "Category", "Label" and "TagIndex" are now automatically migrated to new Fields.

Those new fields have predefined GUIDs that can be retrieved using the function below.

7.1.7.2 Enumeration Type Documentation**7.1.7.2.1 LgsxMigratedMetaKeyField**

enum [LgsxMigratedMetaKeyField](#)

Enum for the migrated metadata.

Each value corresponds to specific migrated field.

Enumerator

LgsxMigratedMetaKeyField_Category	Field "Category".
LgsxMigratedMetaKeyField_Label	Field "Label".
LgsxMigratedMetaKeyField_TagIndex	Field "TagIndex".

7.1.7.3 Function Documentation

7.1.7.3.1 Lgsx_GetMigratedFieldGuid()

```
int Lgsx_GetMigratedFieldGuid (
    LgsxMigratedMetaKeyField field,
    const char ** pId)
```

Get the GUID of the migrated Field specified by "category" argument.

Parameters

in	category	Field specifier.
out	pId	Pointer to store the read-only GUID string.

Returns

Returns 0 for success.

7.1.8 Setup 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_ReaderGetSetupGuid](#) ([LgsxReaderPtr](#) reader, uint64_t setupId, char **pGuidPtr)
Get GUID of the setup for current setup as per [Lgsx_ReaderMoveNextSetup\(\)](#) context.
- 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_ReaderGetImageLayerNames2](#) ([LgsxReaderPtr](#) reader, wchar_t **pLayersPtr)
Retrieve the Image Layer names in the active Setup.
- int [Lgsx_ReaderGetPanolImage](#) ([LgsxReaderPtr](#) reader, int *width, int *height, int *widthInBytes, [ImagePixelType](#) *pixelType, unsigned char **panolImage)
Get the pano image for the active setup.
- int [Lgsx_ReaderGetCubeImage](#) ([LgsxReaderPtr](#) reader, const wchar_t *pLayer, [SCyRgdBdyTxf](#) *txfFrom↵ Setup, 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.

7.1.8.1 Detailed Description

7.1.8.2 Function Documentation

7.1.8.2.1 Lgsx_ReaderEndSetups()

```
int Lgsx_ReaderEndSetups (
    LgsxReaderPtr reader)
```

Frees the active collection.

Parameters

in	<i>reader</i>	Reader handle.
----	---------------	----------------

Returns

Returns 0 for success.

7.1.8.2.2 Lgsx_ReaderEnumSetups()

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

Collects Setups in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumSetup</i>	Number of collected setups.

Returns

0 for success.

7.1.8.2.3 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

in	<i>reader</i>	Reader handle.
in	<i>pLocation</i>	Position of range filter.
in	<i>range</i>	Distance from position for range filter.
out	<i>pNumSetup</i>	Number of collected setups.

Returns

0 for success.

7.1.8.2.4 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

in	<i>reader</i>	Reader handle.
in	<i>pLayer</i>	Cubemap layer to get, see Lgsx_ReaderGetImageLayerNames() .
out	<i>txfFromSetup</i>	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.
out	<i>width</i>	Cubemap image width.
out	<i>height</i>	Cubemap image height.
out	<i>widthInBytes</i>	Cubemap image byte-width.
out	<i>pixelType</i>	Cubemap image pixel type.
out	<i>cubeImages</i>	Array of cubemap images, one for each cube face.

Returns

Returns 0 for success.

7.1.8.2.5 Lgsx_ReaderGetImageLayerNames()

```
int Lgsx_ReaderGetImageLayerNames (
    LgsxReaderPtr reader,
    wchar_t * pLayers,
    int maxLen)
```

Retrieve the Image Layer names in the active Setup.

See also

[Lgsx_ReaderGetImageLayerNames2\(\)](#) is a version that always return complete data without truncation.

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

in	<i>reader</i>	Reader handle.
out	<i>pLayers</i>	Comma-separated list of layer names present in Setup object.
in	<i>maxLen</i>	Length of the given layer string buffer.

Returns

Returns number of layers (≥ 0) on success.

7.1.8.2.6 Lgsx_ReaderGetImageLayerNames2()

```
int Lgsx_ReaderGetImageLayerNames2 (  
    LgsxReaderPtr reader,  
    wchar_t ** pLayersPtr)
```

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

in	<i>reader</i>	Reader handle.
out	<i>pLayersPtr</i>	Pointer to store pointer to comma-separated list of layer names present in Setup object. Buffer must be freed using LgsxUtil_FreeMem() .

Returns

Returns number of layers (≥ 0) on success.

7.1.8.2.7 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.

Remarks

[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

in	<i>reader</i>	Reader handle.
out	<i>width</i>	Pano image width.
out	<i>height</i>	Pano image height.
out	<i>widthInBytes</i>	Pano image byte-width.
out	<i>pixelType</i>	Pano image pixel type.
out	<i>panoImage</i>	Pano image.

Returns

Returns 0 for success.

7.1.8.2.8 Lgsx_ReaderGetSetupGuid()

```
int Lgsx_ReaderGetSetupGuid (
    LgsxReaderPtr reader,
    uint64_t setupId,
    char ** pGuidPtr)
```

Get GUID of the setup for current setup as per [Lgsx_ReaderMoveNextSetup\(\)](#) context.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>setupId</i>	Setup ID to match last setup returned by Lgsx_ReaderMoveNextSetup() .
out	<i>pGuidPtr</i>	Pointer to store pointer to GUID string. Buffer must be freed usign LgsxUtil_FreeMem() .

Returns

Returns 0 for success. Returns -1 if an error occurs.

7.1.8.2.9 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

in	<i>reader</i>	Reader handle.
out	<i>setupId</i>	Setup ID.
out	<i>pSetup</i>	Setup info.
out	<i>pMetadata</i>	Setup metadata. Metadata must be freed usign Lgsx_FreeHandle() . If NULL is given as input, metadata is not returned.

Returns

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

7.1.9 Target 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.

7.1.9.1 Detailed Description

7.1.9.2 Function Documentation

7.1.9.2.1 Lgsx_ReaderEnumTarget()

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

Collects Targets in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumTarget</i>	Number of collected Targets.

Returns

0 for success.

7.1.9.2.2 Lgsx_ReaderEnumTargetOfSetup()

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

Collects Targets in the given Setup and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>setupId</i>	Parent Setup ID.
out	<i>pNumTarget</i>	Number of collected targets.

Returns

0 for success.

7.1.9.2.3 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

in	<i>reader</i>	Reader handle.
out	<i>targetId</i>	Target ID.
out	<i>pTarget</i>	Target info.
out	<i>pMetadata</i>	Target metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

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

7.1.10 Run 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.

7.1.10.1 Detailed Description

7.1.10.2 Function Documentation

7.1.10.2.1 Lgsx_ReaderEndRuns()

```
int Lgsx_ReaderEndRuns (  
    LgsxReaderPtr reader)
```

Frees the active Run collection.

Parameters

in	<i>reader</i>	Reader handle.
----	---------------	----------------

Returns

Returns 0 for success.

7.1.10.2.2 Lgsx_ReaderEnumRuns()

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

Collects Runs in the project and returns the count.

Remarks

Runs are also known as "Tracks" or "Walks" to some integrators.

Runs are generated by kinematic scanners.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumRun</i>	Number of collected Runs.

Returns

0 for success.

7.1.10.2.3 Lgsx_ReaderEnumSetupCamera()

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

Collects Setup Cameras in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumSetupCamera</i>	Number of collected Setup Cameras.

Returns

0 for success.

7.1.10.2.4 Lgsx_ReaderEnumSetupCameraOfSetup()

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

Collects Setup Cameras for the given Setup returns the count.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>setupId</i>	Parent Setup ID.
out	<i>pNumSetupCamera</i>	Number of collected Setup Cameras.

Returns

0 for success.

7.1.10.2.5 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.

Warning

Known issue - this function may return truncated typeName. There is no workaround for this issue at this time.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>typeName</i>	Desired lookup table type.
out	<i>nBytes</i>	Number of bytes in lookup table.
out	<i>lut</i>	Lookup table.

Returns

0 for success.

7.1.10.2.6 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.

Warning

Known issue - this function may return truncated typeName. There is no workaround for this issue at this time.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>setupId</i>	Setup that owns the lookup table.
in	<i>typeName</i>	Desired lookup table type.
out	<i>nBytes</i>	Number of bytes in lookup table.
out	<i>lut</i>	Lookup table.

Returns

0 for success.

7.1.10.2.7 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

in	<i>reader</i>	Reader handle.
in	<i>typeName</i>	Lookup table type.
out	<i>nBytes</i>	Number of bytes in Setup Camera image.
out	<i>blob</i>	Setup Camera image.
out	<i>nThumbBytes</i>	Number of bytes in Setup Camera thumbnail image.
out	<i>thumbBlob</i>	Setup Camera thumbnail image.

Returns

Returns 0 for success.

7.1.10.2.8 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

in	<i>reader</i>	Reader handle.
out	<i>runId</i>	Run ID.
out	<i>info</i>	Target info.
out	<i>nVertex</i>	Number of vertices in the Run's trajectory.
out	<i>ppPath</i>	Array of Trajectory Vertex info structs.
out	<i>pMetadata</i>	Run metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

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

7.1.10.2.9 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

in	<i>reader</i>	Reader handle.
out	<i>pSetupCameraId</i>	Run ID.
out	<i>pSetupCamera</i>	Setup Camera info.
out	<i>pMetadata</i>	Setup Camera metadata. Metadata must be freed usign Lgsx_FreeHandle() . If NULL is given as input, metadata is not returned.

Returns

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

7.1.11 GeoTag Functions

Data Structures

- struct [GeoTagHandle](#)
GeoTag data handle for accessing the GeoTag data.

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.
- int [Lgsx_ReaderGetGeoTagName](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, wchar_t **pNamePtr)
Get name attribute of the GeoTag with the given ID.
- int [Lgsx_ReaderHasGeoTagDescription](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, bool *pStatus)
Check if the GeoTag with the given ID has a description attribute.
- int [Lgsx_ReaderGetGeoTagDescription](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, wchar_t **pDescPtr)
Get description attribute of the GeoTag with the given ID.
- int [Lgsx_ReaderGetGeoTag](#) ([LgsxReaderPtr](#) reader, int pos, [GeoTagHandle](#) *pHandle)
Read GeoTag data at the given position in the active collection.
- int [Lgsx_GeoTagRelease](#) ([GeoTagHandle](#) *pHandle)
Release the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetGuid](#) (const [GeoTagHandle](#) handle, const char **pGuidPtr)
Get the GUID of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetName](#) (const [GeoTagHandle](#) handle, const wchar_t **pNamePtr)
Get name attribute of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagHasDescription](#) (const [GeoTagHandle](#) handle, bool *pStatus)
Check if the GeoTag has a description attribute.
- int [Lgsx_GeoTagGetDescription](#) (const [GeoTagHandle](#) handle, const wchar_t **pDescPtr)
Get description attribute of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetPosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pGeoTagPosition)
Get the position of the GeoTag.
- int [Lgsx_GeoTagGetRelativePosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pGeoTagPosition)
Get the relative position of the GeoTag.
- int [Lgsx_GeoTagHasCameraPosition](#) (const [GeoTagHandle](#) handle, bool *pStatus)
Check if the GeoTag has a camera position.
- int [Lgsx_GeoTagGetCameraPosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pCameraPosition)
Get the camera position of the GeoTag.
- int [Lgsx_GeoTagGetCameraRelativePosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pCameraPosition)
Get the relative camera position of the GeoTag.
- int [Lgsx_GeoTagGetSetupIndex](#) (const [GeoTagHandle](#) handle, uint64_t *pSetupIndex)
Get the Setup Index of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetCreationTimestamp](#) (const [GeoTagHandle](#) handle, uint64_t *pCreationTimestamp)
Get the creation timestamp of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetModificationTimestamp](#) (const [GeoTagHandle](#) handle, uint64_t *pModificationTimestamp)
Get the modification timestamp of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

- Get the modification timestamp of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).*

 - int [Lgsx_GeoTagGetMetadata](#) (const [GeoTagHandle](#) handle, [LgsxMetadataPtr](#) *pMetadata)

Get the metadata of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

 - int [Lgsx_GeoTagEnumCategoryEntries](#) (const [GeoTagHandle](#) handle, int *pNumEntries)

Enumerate the Category Entries associated with the GeoTag and return the count.

 - int [Lgsx_GeoTagGetCategoryEntry](#) (const [GeoTagHandle](#) handle, int pos, [CategoryEntryHandle](#) *pEntry↵
Handle)

Get a Category Entry at the specified position for the GeoTag.

 - int [Lgsx_GeoTagEnumFieldEntries](#) (const [GeoTagHandle](#) handle, int *pNumEntries)

Enumerate the Field Entries associated with the GeoTag and return the count.

 - int [Lgsx_GeoTagGetFieldEntry](#) (const [GeoTagHandle](#) handle, int pos, [FieldEntryHandle](#) *pEntryHandle)

Get a Field Entry at the specified position for the GeoTag.

 - int [Lgsx_GeoTagEnumAssets](#) (const [GeoTagHandle](#) handle, int *pNumAssets)

Collects Assets associated with the given GeoTag and returns the count.

 - int [Lgsx_ReaderGetGeoTagAsset](#) ([LgsxReaderPtr](#) reader, const [GeoTagHandle](#) handle, int pos, [CYASSET](#) *pAsset, [LgsxMetadataPtr](#) *pMetadata)

Get Info and metadata for the GeoTag Asset at the given position.

 - int [Lgsx_ReaderGetGeoTagAssetThumbnail](#) ([LgsxReaderPtr](#) reader, const [GeoTagHandle](#) handle, int pos, int *pWidth, int *pHeight, int *pWidthInBytes, [ImagePixelType](#) *pPixelType, unsigned char **pImage)

Get the thumbnail image for GeoTag Asset at the given position.

 - int [Lgsx_GeoTagHasThumbnail](#) (const [GeoTagHandle](#) handle, bool *pHasThumbnail)

Checks if GeoTag has a thumbnail.

 - int [Lgsx_GeoTagGetThumbnail](#) (const [GeoTagHandle](#) handle, int *pWidth, int *pHeight, int *pWidthInBytes, [ImagePixelType](#) *pPixelType, unsigned char **pImage)

Get the thumbnail image for the GeoTag itself.

7.1.11.1 Detailed Description

7.1.11.2 Function Documentation

7.1.11.2.1 Lgsx_GeoTagEnumAssets()

```
int Lgsx_GeoTagEnumAssets (
    const GeoTagHandle handle,
    int * pNumAssets)
```

Collects Assets associated with the given GeoTag and returns the count.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pNumAssets</i>	Number of collected Assets.

Returns

Returns 0 for success.

7.1.11.2.2 Lgsx_GeoTagEnumCategoryEntries()

```
int Lgsx_GeoTagEnumCategoryEntries (
    const GeoTagHandle handle,
    int * pNumEntries)
```

Enumerate the Category Entries associated with the GeoTag and return the count.

Parameters

in	<i>handle</i>	GeoTag handle.
out	<i>pNumEntries</i>	Pointer to store the number of Category Entries in the GeoTag.

Returns

Returns 0 for success.

7.1.11.2.3 Lgsx_GeoTagEnumFieldEntries()

```
int Lgsx_GeoTagEnumFieldEntries (
    const GeoTagHandle handle,
    int * pNumEntries)
```

Enumerate the Field Entries associated with the GeoTag and return the count.

Parameters

in	<i>handle</i>	GeoTag handle.
out	<i>pNumEntries</i>	Pointer to store the number of Field Entries in the GeoTag.

Returns

Returns 0 for success.

7.1.11.2.4 Lgsx_GeoTagGetCameraPosition()

```
int Lgsx_GeoTagGetCameraPosition (
    const GeoTagHandle handle,
    PNT3D * pCameraPosition)
```

Get the camera position of the GeoTag.

Camera position is specified in the project coordinates.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pCameraPosition</i>	Pointer to a structure where the function will store the camera position.

Returns

Returns 0 for success.

7.1.11.2.5 Lgsx_GeoTagGetCameraRelativePosition()

```
int Lgsx_GeoTagGetCameraRelativePosition (
    const GeoTagHandle handle,
    PNT3D * pCameraPosition)
```

Get the relative camera position of the GeoTag.

Camera position is specified in the coordinates relative to setup.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pCameraPosition</i>	Pointer to a structure where the function will store the relative camera position.

Returns

Returns 0 for success.

7.1.11.2.6 Lgsx_GeoTagGetCategoryEntry()

```
int Lgsx_GeoTagGetCategoryEntry (
    const GeoTagHandle handle,
    int pos,
    CategoryEntryHandle * pEntryHandle)
```

Get a Category Entry at the specified position for the GeoTag.

Parameters

in	<i>handle</i>	GeoTag handle.
in	<i>pos</i>	Position in the GeoTag's Category Entry collection.
out	<i>pEntryHandle</i>	Pointer to store the Category Entry handle. Returns read-only pointer bound to lifetime of GeoTagHandle , shall not be released manually.

Returns

Returns 0 for success. Returns -1 when position is out of range.

7.1.11.2.7 Lgsx_GeoTagGetCreationTimestamp()

```
int Lgsx_GeoTagGetCreationTimestamp (
    const GeoTagHandle handle,
    uint64_t * pCreationTimestamp)
```

Get the creation timestamp of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pCreationTimestamp</i>	Pointer to store creation timestamp of the GeoTag.

Returns

Returns 0 for success.

7.1.11.2.8 Lgsx_GeoTagGetDescription()

```
int Lgsx_GeoTagGetDescription (
    const GeoTagHandle handle,
    const wchar_t ** pDescPtr)
```

Get description attribute of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pDescPtr</i>	Pointer to store description buffer. Buffer is read-only, bound to the lifetime of the GeoTag data. Must not be freed.

Returns

Returns 0 for success.

7.1.11.2.9 Lgsx_GeoTagGetFieldEntry()

```
int Lgsx_GeoTagGetFieldEntry (
    const GeoTagHandle handle,
    int pos,
    FieldEntryHandle * pEntryHandle)
```

Get a Field Entry at the specified position for the GeoTag.

Parameters

in	<i>handle</i>	GeoTag handle.
in	<i>pos</i>	Position in the GeoTag's Field Entry collection.
out	<i>pEntryHandle</i>	Pointer to store the Field Entry handle. Returns read-only pointer bound to lifetime of GeoTagHandle , shall not be released manually.

Returns

Returns 0 for success. Returns -1 when position is out of range.

7.1.11.2.10 Lgsx_GeoTagGetGuid()

```
int Lgsx_GeoTagGetGuid (
    const GeoTagHandle handle,
    const char ** pGuidPtr)
```

Get the GUID of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pGuidPtr</i>	Pointer to store GUID buffer of the GeoTag. Buffer is read-only, bound to the lifetime of the GeoTag data. Must not be freed.

Returns

Returns 0 for success.

7.1.11.2.11 Lgsx_GeoTagGetMetadata()

```
int Lgsx_GeoTagGetMetadata (
    const GeoTagHandle handle,
    LgsxMetadataPtr * pMetadata)
```

Get the metadata of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pMetadata</i>	GeoTag metadata. Metadata is bound to the lifetime of the GeoTagHandle data. Must not be freed.

Returns

Returns 0 for success.

7.1.11.2.12 Lgsx_GeoTagGetModificationTimestamp()

```
int Lgsx_GeoTagGetModificationTimestamp (
    const GeoTagHandle handle,
    uint64_t * pModificationTimestamp)
```

Get the modification timestamp of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pModificationTimestamp</i>	Pointer to store modification timestamp of the GeoTag.

Returns

Returns 0 for success.

7.1.11.2.13 Lgsx_GeoTagGetName()

```
int Lgsx_GeoTagGetName (
    const GeoTagHandle handle,
    const wchar_t ** pNamePtr)
```

Get name attribute of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pNamePtr</i>	Pointer to store name buffer. Buffer is read-only, bound to the lifetime of the GeoTag data. Must not be freed.

Returns

Returns 0 for success.

7.1.11.2.14 Lgsx_GeoTagGetPosition()

```
int Lgsx_GeoTagGetPosition (
    const GeoTagHandle handle,
    PNT3D * pGeoTagPosition)
```

Get the position of the GeoTag.

Position is specified in the project coordinates.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pGeoTagPosition</i>	Pointer to a structure where the function will store the position of the GeoTag.

Returns

Returns 0 for success.

7.1.11.2.15 Lgsx_GeoTagGetRelativePosition()

```
int Lgsx_GeoTagGetRelativePosition (
    const GeoTagHandle handle,
    PNT3D * pGeoTagPosition)
```

Get the relative position of the GeoTag.

Position is specified in the the coordinates relative to setup.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pGeoTagPosition</i>	Pointer to a structure where the function will store the relative position of the GeoTag.

Returns

Returns 0 for success.

7.1.11.2.16 Lgsx_GeoTagGetSetupIndex()

```
int Lgsx_GeoTagGetSetupIndex (
    const GeoTagHandle handle,
    uint64\_t * pSetupIndex)
```

Get the Setup Index of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pSetupIndex</i>	Pointer to store Setup Index of the GeoTag.

Returns

Returns 0 for success.

7.1.11.2.17 Lgsx_GeoTagGetThumbnail()

```
int Lgsx_GeoTagGetThumbnail (
    const GeoTagHandle handle,
    int * pWidth,
    int * pHeight,
    int * pWidthInBytes,
    ImagePixelFormat * pPixelFormat,
    unsigned char ** pImage)
```

Get the thumbnail image for the GeoTag itself.

Note

If the thumbnail does not exist, an error is returned; please check that GeoTag has thumbnail using [Lgsx_GeoTagHasThumbnail\(\)](#).

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pWidth</i>	Image width.
out	<i>pHeight</i>	Image height.
out	<i>pWidthInBytes</i>	Image byte-width.
out	<i>pPixelFormat</i>	Pixel type.
out	<i>pImage</i>	GeoTag thumbnail image.

Returns

Returns 0 for success.

7.1.11.2.18 Lgsx_GeoTagHasCameraPosition()

```
int Lgsx_GeoTagHasCameraPosition (
    const GeoTagHandle handle,
    bool * pStatus)
```

Check if the GeoTag has a camera position.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pStatus</i>	Pointer to flag where the function will store whether geotag has camera position.

7.1.11.2.19 Lgsx_GeoTagHasDescription()

```
int Lgsx_GeoTagHasDescription (
    const GeoTagHandle handle,
    bool * pStatus)
```

Check if the GeoTag has a description attribute.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pStatus</i>	Pointer to store the status.

Returns

Returns 0 for success.

7.1.11.2.20 Lgsx_GeoTagHasThumbnail()

```
int Lgsx_GeoTagHasThumbnail (
    const GeoTagHandle handle,
    bool * pHasThumbnail)
```

Checks if GeoTag has a thumbnail.

Parameters

in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
out	<i>pHasThumbnail</i>	Set to true if the GeoTag has thumbnail.

Returns

Returns 0 for success.

7.1.11.2.21 Lgsx_GeoTagRelease()

```
int Lgsx_GeoTagRelease (
    GeoTagHandle * pHandle)
```

Release the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).

Parameters

in	<i>pHandle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
----	----------------	---

Returns

Returns 0 for success.

7.1.11.2.22 Lgsx_ReaderEnumGeoTags()

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

Collects GeoTags in the project and returns the count.

Warning

Known issue - this function may require a prior call to [Lgsx_ReaderEnumSetups\(\)](#).

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumGeoTag</i>	Number of collected GeoTags.

Returns

0 for success.

7.1.11.2.23 Lgsx_ReaderEnumGeoTagsOfSetup()

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

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

Parameters

in	<i>reader</i>	Reader handle.
in	<i>setupId</i>	Setup associated with GeoTags.
out	<i>pNumGeoTag</i>	Number of collected GeoTags.

Returns

0 for success.

7.1.11.2.24 Lgsx_ReaderGetGeoTag()

```
int Lgsx_ReaderGetGeoTag (
    LgsxReaderPtr reader,
    int pos,
    GeoTagHandle * pHandle)
```

Read GeoTag data at the given position in the active collection.

Active collection is set by [Lgsx_ReaderEnumGeoTags\(\)](#) or [Lgsx_ReaderEnumGeoTagsOfSetup\(\)](#).

Parameters

in	<i>reader</i>	Reader handle.
in	<i>pos</i>	Position in the active collection.
out	<i>pHandle</i>	Pointer to store pointer to GeoTag data. Buffer must be freed using Lgsx_GeoTagRelease() .

Returns

Returns 0 for success. Returns -1 when position is out of range.

7.1.11.2.25 Lgsx_ReaderGetGeoTagAsset()

```
int Lgsx_ReaderGetGeoTagAsset (
    LgsxReaderPtr reader,
    const GeoTagHandle handle,
    int pos,
    CYASSET * pAsset,
    LgsxMetadataPtr * pMetadata)
```

Get Info and metadata for the GeoTag Asset at the given position.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
in	<i>pos</i>	Position in the GeoTag's Asset collection.
out	<i>pAsset</i>	Asset info.
out	<i>pMetadata</i>	Asset metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

Returns 0 for success.

7.1.11.2.26 Lgsx_ReaderGetGeoTagAssetThumbnail()

```
int Lgsx_ReaderGetGeoTagAssetThumbnail (
    LgsxReaderPtr reader,
    const GeoTagHandle handle,
    int pos,
    int * pWidth,
    int * pHeight,
    int * pWidthInBytes,
    ImagePixelType * pPixelType,
    unsigned char ** pImage)
```

Get the thumbnail image for GeoTag Asset at the given position.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>handle</i>	Pointer to GeoTag data returned by Lgsx_ReaderGetGeoTag() .
in	<i>pos</i>	Position in the GeoTag's Asset collection.
out	<i>pWidth</i>	Image width.
out	<i>pHeight</i>	Image height.
out	<i>pWidthInBytes</i>	Thumbnail image byte-width.
out	<i>pPixelType</i>	Pixel type.
out	<i>pImage</i>	Asset thumbnail image.

Returns

Returns 0 for success.

7.1.11.2.27 Lgsx_ReaderGetGeoTagDescription()

```
int Lgsx_ReaderGetGeoTagDescription (
    LgsxReaderPtr reader,
    uint64_t geoTagId,
    wchar_t ** pDescPtr)
```

Get description attribute of the GeoTag with the given ID.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>geoTagId</i>	GeoTag ID.
out	<i>pDescPtr</i>	Pointer to store description buffer. Buffer must be freed using LgsxUtil_FreeMem() .

Returns

Returns 0 for success.

Deprecated 26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

7.1.11.2.28 Lgsx_ReaderGetGeoTagName()

```
int Lgsx_ReaderGetGeoTagName (
    LgsxReaderPtr reader,
    uint64_t geoTagId,
    wchar_t ** pNamePtr)
```

Get name attribute of the GeoTag with the given ID.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>geoTagId</i>	GeoTag ID.
out	<i>pNamePtr</i>	Pointer to store name buffer. Buffer must be freed using LgsxUtil_FreeMem() .

Returns

Returns 0 for success.

Deprecated 26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

7.1.11.2.29 Lgsx_ReaderHasGeoTagDescription()

```
int Lgsx_ReaderHasGeoTagDescription (
    LgsxReaderPtr reader,
    uint64_t geoTagId,
    bool * pStatus)
```

Check if the GeoTag with the given ID has a description attribute.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>geo↔ TagId</i>	GeoTag ID.
out	<i>pStatus</i>	Pointer to store the status.

Returns

Returns 0 for success.

Deprecated 26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

7.1.11.2.30 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

in	<i>reader</i>	Reader handle.
out	<i>geoTagId</i>	GeoTag ID.
out	<i>pGeoTag</i>	GeoTag info.
out	<i>pMetadata</i>	GeoTag metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

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

Deprecated 26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

7.1.12 Sitemap Functions

Functions

- int [Lgsx_ReaderEnumSitemaps](#) ([LgsxReaderPtr](#) reader, int *pNumSitemap)
Collects Sitemaps in the project and returns the count.
- int [Lgsx_ReaderEnumSiteMaps](#) ([LgsxReaderPtr](#) reader, int *pNumSitemap)
Collects Sitemaps in the project and returns the count.
- int [Lgsx_ReaderMoveNextSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t *sitemapId, [CYSITEMAP](#) *pSitemap↔
Info, [LgsxMetadataPtr](#) *pMetadata)
Advance to the next Sitemap in the active collection and return its ID, info, and metadata.
- int [Lgsx_ReaderGetSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, [CYSITEMAP](#) *pSitemapInfo)

Get Info and metadata for the given Sitemap ID.

- int [Lgsx_ReaderGetSitemapImage](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, [CYSITEMAPIMAGE](#) *p↔ SitemapImageInfo)

Get the image for the given Sitemap ID.

- 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.

- 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.

- int [Lgsx_ReaderEnumSetupsForSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, int *pNumSetups)

Collects Setups in the project that are associated with the given Sitemap and returns the count.

- int [Lgsx_ReaderMoveNextSetupForSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, uint64_t *pSetupId, [CYSETUPINFO](#) *pSetup, [LgsxMetadataPtr](#) *pMetadata)

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

7.1.12.1 Detailed Description

7.1.12.2 Function Documentation

7.1.12.2.1 Lgsx_ReaderEnumSetupsForSitemap()

```
int Lgsx_ReaderEnumSetupsForSitemap (
    LgsxReaderPtr reader,
    uint64_t sitemapId,
    int * pNumSetups)
```

Collects Setups in the project that are associated with the given Sitemap and returns the count.

Warning

This is experimental API, subject to change (WIP)

Parameters

in	<i>reader</i>	Reader handle.
in	<i>sitemapId</i>	Sitemap ID.
out	<i>pNumSetups</i>	Number of collected Setups.

Returns

0 for success.

7.1.12.2.2 Lgsx_ReaderEnumSiteMaps()

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

Collects Sitemaps in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumSitemap</i>	Number of collected Sitemaps.

Returns

0 for success.

Deprecated 18.04.2025 - Renamed to [Lgsx_ReaderEnumSitemaps\(\)](#)

7.1.12.2.3 Lgsx_ReaderEnumSitemaps()

```
int Lgsx_ReaderEnumSitemaps (  
    LgsxReaderPtr reader,  
    int * pNumSitemap)
```

Collects Sitemaps in the project and returns the count.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumSitemap</i>	Number of collected Sitemaps.

Returns

0 for success.

7.1.12.2.4 Lgsx_ReaderGetSitemap()

```
int Lgsx_ReaderGetSitemap (  
    LgsxReaderPtr reader,  
    uint64_t sitemapId,  
    CYSITEMAP * pSitemapInfo)
```

Get Info and metadata for the given Sitemap ID.

Warning

This is experimental API, subject to change (WIP)

Parameters

in	<i>reader</i>	Reader handle.
in	<i>sitemapId</i>	Sitemap ID.
out	<i>pSitemapInfo</i>	Sitemap info.

Returns

Returns 0 for success.

7.1.12.2.5 Lgsx_ReaderGetSitemapImage()

```
int Lgsx_ReaderGetSitemapImage (  
    LgsxReaderPtr reader,  
    uint64_t sitemapId,  
    CYSITEMAPIMAGE * pSitemapImageInfo)
```

Get the image for the given Sitemap ID.

Warning

This is experimental API, subject to change (WIP)

Parameters

in	<i>reader</i>	Reader handle.
in	<i>sitemapId</i>	Sitemap ID.
in, out	<i>pSitemapImageInfo</i>	Sitemap image info. Use <i>imageType</i> to specify the type of image to get.

Returns

Returns 0 for success.

7.1.12.2.6 Lgsx_ReaderMoveNextSetupForSitemap()

```
int Lgsx_ReaderMoveNextSetupForSitemap (  
    LgsxReaderPtr reader,  
    uint64_t sitemapId,  
    uint64_t * pSetupId,  
    CYSETUPINFO * pSetup,  
    LgsxMetadataPtr * pMetadata)
```

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

Warning

This is experimental API, subject to change (WIP)

Parameters

in	<i>reader</i>	Reader handle.
in	<i>sitemapId</i>	Sitemap ID.
out	<i>pSetupId</i>	Setup ID.
out	<i>pSetup</i>	Setup info.
out	<i>pMetadata</i>	Setup metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

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

7.1.12.2.7 Lgsx_ReaderMoveNextSitemap()

```
int Lgsx_ReaderMoveNextSitemap (  
    LgsxReaderPtr reader,  
    uint64_t * sitemapId,  
    CYSITEMAP * pSitemapInfo,  
    LgsxMetadataPtr * pMetadata)
```

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

Warning

This is experimental API, subject to change (WIP)

Parameters

in	<i>reader</i>	Reader handle.
out	<i>sitemapId</i>	Sitemap ID.
out	<i>pSitemapInfo</i>	Sitemap info.
out	<i>pMetadata</i>	Sitemap metadata. Metadata must be freed usign Lgsx_FreeHandle() .

Returns

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

7.1.12.2.8 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

in	<i>reader</i>	Reader handle.
out	<i>width</i>	Background image width.
out	<i>height</i>	Background image height.
out	<i>widthInBytes</i>	Background image width in bytes.
out	<i>pixelType</i>	Background image pixel type.
out	<i>pImage</i>	Background image.
out	<i>world2screen</i>	Background image world to screen matrix.

Returns

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

Deprecated 18.04.2025 - Renamed to [Lgsx_ReaderMoveNextSitemapImage\(\)](#)

7.1.12.2.9 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

in	<i>reader</i>	Reader handle.
out	<i>width</i>	Background image width.
out	<i>height</i>	Background image height.
out	<i>widthInBytes</i>	Background image width in bytes.
out	<i>pixelType</i>	Background image pixel type.
out	<i>pImage</i>	Background image.
out	<i>world2screen</i>	Background image world to screen matrix.

Returns

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

7.1.13 Model and Model Node 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.

7.1.13.1 Detailed Description

7.1.13.2 Function Documentation

7.1.13.2.1 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

in	<i>reader</i>	Reader handle.
in	<i>modelId</i>	ID of the model to query.
out	<i>pModelInfo</i>	Info for the requested Model.
out	<i>pMetadata</i>	Model Metadata. Metadata must be freed usign Lgsx_FreeHandle() .
out	<i>ppRefAssetIds</i>	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.

7.1.13.2.2 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

in	<i>reader</i>	Reader handle.
in	<i>nodeId</i>	ID of the model node to query.
out	<i>pNodeInfo</i>	Info for the requested Model Node.
out	<i>ppChildIds</i>	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.

7.1.13.2.3 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

in	<i>reader</i>	Reader handle.
out	<i>pNumModelNode</i>	Number of Model Nodes returned.
out	<i>ppModelNodeIds</i>	Array of Model Node IDs.

Returns

Returns 0 for success.

7.1.14 Point Cloud 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 *pn↔Bytes)
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_ReaderEnumPointsEx](#) ([LgsxReaderPtr](#) reader, int desiredTypes, int subSample, bool visible, bool scriptOrder)
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 desired 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.

7.1.14.1 Detailed Description

7.1.14.2 Function Documentation

7.1.14.2.1 Lgsx_ReaderEnumPoints()

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

Collects points in the Point Cloud.

Parameters

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

Returns

Returns 0 for success.

7.1.14.2.2 Lgsx_ReaderEnumPointsEx()

```
int Lgsx_ReaderEnumPointsEx (
    LgsxReaderPtr reader,
    int desiredTypes,
    int subSample,
    bool visible,
    bool scriptOrder)
```

Collects points in the Point Cloud.

Parameters

in	<i>reader</i>	Reader handle.
in	<i>desiredTypes</i>	Mask containing the properties to be retrieved during iteration.
in	<i>subSample</i>	Not currently supported and should be set to 1.
in	<i>visible</i>	Limits collection to "visible" points when true. Otherwise collection includes clipped points.
in	<i>scriptOrder</i>	If true, the points are returned in the stable order (if possible).

Returns

Returns 0 for success.

7.1.14.2.3 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.

Warning

When hspcPathname is empty it will be used to store auto-generated name. The buffer must be at least 280 characters long.

Parameters

in	<i>reader</i>	Reader handle.
in, out	<i>hspcPathname</i>	Path at which to store the extracted file if not empty. Buffer to store path (if empty on function call).

Returns

Returns 0 for success.

7.1.14.2.4 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

in	<i>reader</i>	Reader handle.
out	<i>ppClassModels</i>	Classification model names. Names are null-terminated strings packed into a single buffer.

Returns

Returns the number of model names.

7.1.14.2.5 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

in	<i>reader</i>	Reader handle.
in	<i>modelIdx</i>	Index of the model to retrieve.
out	<i>ppClassNames</i>	Array of classification name strings. Strings are null-terminated and packed into the returned buffer.
out	<i>ppClassVisibles</i>	Array of flags indicating whether the classification is visible by default.
out	<i>ppClassColors</i>	Array of colors for each classification. Each color is 4-byte RGBA format.

Returns

Returns the number of classifications in the requested model.

7.1.14.2.6 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

in	<i>reader</i>	Reader handle.
in	<i>typeMask</i>	Property type to query (see Property type masks).
in	<i>name</i>	Property name being queried. Only used for user-defined types.
out	<i>pnBytes</i>	Number of bytes used for values of this property type.

Returns

Returns 0 for success.

7.1.14.2.7 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 desired properties.

The desired properties are specified in the call to [Lgsx_ReaderEnumPoints\(\)](#). 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

in	<i>reader</i>	Reader handle.
in	<i>chunkSize</i>	Number of points to read per call.
in	<i>bmpFormat</i>	Bitmap format to use for returned colors (BitmapFormat__RGB or BitmapFormat__RGBA)
out	<i>ppData</i>	Array of buffers to be filled with point data.

Returns

Returns number of points retrieved. Note that this can be less than the number of points requested. Return value of 0 indicates there are no more points to read.

7.1.14.2.8 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

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

Returns

Returns number of points retrieved. Note that this can be less than the number of points requested. Return value of 0 indicates there are no more points to read.

7.1.14.2.9 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

in	<i>reader</i>	Reader handle.
----	---------------	----------------

Returns

Property type mask for this project.

7.1.15 User Coordinate System 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.
- int [Lgsx_ReaderMoveNextCoordSystems2](#) ([LgsxReaderPtr](#) reader, bool *pActive, wchar_t **pNamePtr, wchar_t **pWktPtr, [SCyRgdBdyTxf](#) *pTxf)
Advances to the next User Coordinate System (UCS) object and returns its properties.

7.1.15.1 Detailed Description

7.1.15.2 Function Documentation

7.1.15.2.1 Lgsx_ReaderEnumCoordSystems()

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

Collects User Coordinate Systems.

Parameters

in	<i>reader</i>	Reader handle.
out	<i>pNumCs</i>	Number of User Coordinate Systems.

Returns

Returns 0 for success.

7.1.15.2.2 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 Coordinate Reference 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.

See also

[Lgsx_ReaderMoveNextCoordSystems2\(\)](#) is a version that always return complete data without truncation.

Parameters

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

Returns

Returns 0 for success.

7.1.15.2.3 Lgsx_ReaderMoveNextCoordSystems2()

```
int Lgsx_ReaderMoveNextCoordSystems2 (
    LgsxReaderPtr reader,
    bool * pActive,
    wchar_t ** pNamePtr,
    wchar_t ** pWktPtr,
    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 Coordinate 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

in	<i>reader</i>	Reader handle.
out	<i>pActive</i>	Active flag, set to true if this UCS is active.
out	<i>pNamePtr</i>	Pointer to store pointer to UCS name. Buffer must be freed using LgsxUtil_FreeMem() .
out	<i>pWktPtr</i>	Pointer to store pointer WKT string for this UCS. Buffer must be freed using LgsxUtil_FreeMem() .
in	<i>pTxf</i>	Transform/rotation for this UCS.

Returns

Returns 0 for success.

7.2 General Functions

General convenience functions.

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_StrDupA](#) (const char *value, char **ptrDupValue)
Duplicate string.
- int [LgsxUtil_StrDupW](#) (const wchar_t *value, wchar_t **ptrDupValue)
Duplicate string.
- 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.

7.2.1 Detailed Description

General convenience functions.

Methods in this class are actually global C APIs.

7.2.2 Function Documentation

7.2.2.1 Lgsx_FreeHandle()

```
void Lgsx_FreeHandle (
    void * handle)
```

Frees the SDK object referenced by the given handle.

Parameters

in	<i>handle</i>	Handle to free.
----	---------------	-----------------

7.2.2.2 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.

7.2.2.3 Lgsx_GetLastErrorNo()

```
int Lgsx_GetLastErrorNo ()
```

Returns error number for most recent SDK error.

Returns

Error number.

7.2.2.4 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

out	<i>version</i>	Returned version string. Need to pre-allocate at least 40 characters.
out	<i>buildNum</i>	Returned build number.

Returns

0 for success.

7.2.2.5 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

in	<i>build</i>	Build number.
----	--------------	---------------

Returns

0 for success.

7.2.2.6 Lgsx_InitProductName()

```
int Lgsx_InitProductName (  
    const wchar_t * prodName)
```

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

Parameters

in	<i>prodName</i>	Product name
----	-----------------	--------------

Returns

0 for success.

7.2.2.7 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

in	<i>customVersion</i>	Custom version string, such as "2.1A".
----	----------------------	--

Returns

0 for success.

7.2.2.8 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

in	<i>customVersion</i>	Custom version string, such as "2.1A".
in	<i>fileVersion</i>	Version string to be displayed in module/dll.

Returns

0 for success.

7.2.2.9 LgsxUtil_A2W()

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

Convert UTF8 encoded text to UTF16.

Parameters

in	<i>value</i>	Source string.
out	<i>tValue</i>	Converted string.

Returns

0 for success.

7.2.2.10 LgsxUtil_AllocMem()

```
void * LgsxUtil_AllocMem (  
    int size)
```

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

Parameters

in	<i>size</i>	Size of memory to create, in bytes.
----	-------------	-------------------------------------

Returns

Pointer to allocated memory.

7.2.2.11 LgsxUtil_FreeMem()

```
void LgsxUtil_FreeMem (  
    void ** ptr)
```

Free a block of memory previously created by the SDK.

Parameters

in	<i>ptr</i>	Pointer to memory to be freed.
----	------------	--------------------------------

7.2.2.12 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.

7.2.2.13 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.

7.2.2.14 LgsxUtil_Sleep()

```
int LgsxUtil_Sleep (
    int millisec)
```

Suspends execution for the given number of milliseconds.

Parameters

in	<i>millisec</i>	Number of milliseconds to sleep.
----	-----------------	----------------------------------

Returns

0 for success.

7.2.2.15 LgsxUtil_StrDupA()

```
int LgsxUtil_StrDupA (
    const char * value,
    char ** ptrDupValue)
```

Duplicate string.

The duplicated string must be freed using [LgsxUtil_FreeMem\(\)](#).

Parameters

in	<i>value</i>	Source string. If NULL, the function will return error.
out	<i>ptrDupValue</i>	Pointer to store pointer to duplicated string.

Returns

0 for success.

7.2.2.16 LgsxUtil_StrDupW()

```
int LgsxUtil_StrDupW (  
    const wchar_t * value,  
    wchar_t ** ptrDupValue)
```

Duplicate string.

The duplicated string must be freed using [LgsxUtil_FreeMem\(\)](#).

Parameters

in	<i>value</i>	Source string. If NULL, the function will return error.
out	<i>ptrDupValue</i>	Pointer to store pointer to duplicated string.

Returns

0 for success.

7.2.2.17 LgsxUtil_TimeEnd()

```
int LgsxUtil_TimeEnd (  
    void ** pTime)
```

Frees the timer object.

Parameters

in	<i>pTime</i>	Timer object handle.
----	--------------	----------------------

Returns

0 for success.

7.2.2.18 LgsxUtil_TimeGetLapse()

```
double LgsxUtil_TimeGetLapse (  
    void * pTime)
```

Returns elapsed time for the given timer in seconds.

Parameters

in	<i>pTime</i>	Timer object handle.
----	--------------	----------------------

Returns

Elapsed time in seconds.

7.2.2.19 LgsxUtil_TimeGetLapseStr()

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

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

Parameters

in	<i>pTime</i>	Timer object handle.
in	<i>format</i>	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.

7.2.2.20 LgsxUtil_TimeStart()

```
void * LgsxUtil_TimeStart ()
```

Create a new timer and starts it.

Returns

Handle for the new Timer object.

7.2.2.21 LgsxUtil_W2A()

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

Convert UTF16 encoded text to UTF8.

Parameters

in	<i>value</i>	Source string.
out	<i>sValue</i>	Converted string.

Returns

0 for success.

7.3 Application Functions

Functions that all SDK Applications will need to use.

Functions

- int [Lgsx_Initialize](#) (const wchar_t *pTempPath, bool enableLog)
Initialize the LGSx SDK APIs.
- int [Lgsx_Uninitialize](#) ()
Unintialize LGSx SDK APIs.

7.3.1 Detailed Description

Functions that all SDK Applications will need to use.

7.3.2 Function Documentation

7.3.2.1 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 a project.

Parameters

in	<i>pTempPath</i>	Path to folder to be used for temporary files used by the SDK.
in	<i>enableLog</i>	Set to TRUE to enable the SDK's logging/log file. Logging is disabled if set to FALSE.

Returns

0 for success.

7.3.2.2 Lgsx_Uninitialize()

```
int Lgsx_Uninitialize ()
```

Unintialize LGSx SDK APIs.

This should always be called before exiting from your application. It is not recommended to call the pair of [Lgsx_Initialize](#) and [Lgsx_Uninitialize](#) multiple times during a session.

Returns

0 for success.

7.4 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.

7.4.1 Detailed Description

7.4.2 Function Documentation

7.4.2.1 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:

- `Lgsx_ReaderOpen`

Parameters

in	<i>lic</i>	The name string of the license.
in	<i>ver</i>	The version string of the license.

Returns

0 for success.

7.4.2.2 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

in	<i>lic</i>	The name string of the license.
in	<i>ver</i>	The version string of the license.
in	<i>product</i>	The product name.

Returns

0 for success.

7.4.2.3 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

in	<i>lic</i>	The name string of the license.
in	<i>ult</i>	The name string of ultimate license (or empty to not support ultimate).
in	<i>ver</i>	The version string of the license.
in	<i>product</i>	The product name.

Returns

0 for success.

7.4.2.4 Lgsx_IsLicensed()

```
int Lgsx_IsLicensed ()
```

Check if the license is valid.

Returns

1 means it is licensed.

7.4.2.5 Lgsx_IsLicensedExA()

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

Check if a named license is valid.

Parameters

in	<i>lic</i>	The name string of the license.
in	<i>ver</i>	The version string of the license.

Returns

1 means it is licensed.

7.4.2.6 Lgsx_LicenseDaysLeft()

```
int Lgsx_LicenseDaysLeft ()
```

Check the number of days left for the license.

Returns

The number of days.

7.4.2.7 Lgsx_LicenseDaysLeftExA()

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

Check the number of days left for given license.

Parameters

in	<i>lic</i>	The name string of the license.
in	<i>ver</i>	The version string of the license.

Returns

The number of days.

7.4.2.8 Lgsx_LoadLicense()

```
int Lgsx_LoadLicense ()
```

Load license and hold the usage.

Returns

0 for success.

7.4.2.9 Lgsx_LoadLicenseExA()

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

Load given license and hold usage until UnloadLicense.

Parameters

in	<i>lic</i>	The name string of the license.
in	<i>ver</i>	The version string of the license.

Returns

0 for success.

7.4.2.10 Lgsx_UnloadLicense()

```
int Lgsx_UnloadLicense ()
```

Unload license.

Returns

0 for success.

7.4.2.11 Lgsx_UnloadLicenseExA()

```
int Lgsx_UnloadLicenseExA (  
    const char * lic)
```

Unload a named license.

Parameters

in	<i>lic</i>	The name string of the license.
----	------------	---------------------------------

Returns

0 for success.

7.5 Metadata Functions

Functions for reading, writing, and parsing metadata objects.

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_MetaMoveNext2](#) (LgsxMetadataPtr pMetadata, const char **ppName, 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_MetaGetString2](#) (LgsxMetadataPtr pMetadata, const [LgsxPropertyName](#) pName, wchar_t **ppValue)
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.

7.5.1 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.

7.5.2 Function Documentation

7.5.2.1 Lgsx_MetaAddBool()

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

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>value</i>	Value to add. Be careful that the type of value is important.

Note

The pName may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.2 Lgsx_MetaAddDouble()

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

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>value</i>	Value to add. Be careful that the type of value is important.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.3 Lgsx_MetaAddDouble3()

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

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>pValue</i>	Value to add. Be careful that the type of value is important.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.4 Lgsx_MetaAddDouble4()

```
int Lgsx_MetaAddDouble4 (  
    LgsxMetadataPtr pMetadata,  
    const LgsxPropertyName pName,  
    const double * pValue)
```

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>pValue</i>	Value to add. Be careful that the type of value is important.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.5 Lgsx_MetaAddInt()

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

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>value</i>	Value to add. Be careful that the type of value is important.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.6 Lgsx_MetaAddInt64()

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

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>value</i>	Value to add. Be careful that the type of value is important.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.7 Lgsx_MetaAddMetadata()

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

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>p↔ MetadataIn</i>	Value to add. Be careful that the type of value is important.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.8 Lgsx_MetaAddString()

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

Add name value pair to metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
in	<i>pValue</i>	Value to add. Be careful that the type of value is important.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.9 Lgsx_MetaClone()

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

Clones a metadata container.

Parameters

in	<i>pMetadata</i>	Metadata object to be cloned.
----	------------------	-------------------------------

Returns

Metadata object. You need to delete it using [Lgsx_FreeHandle](#).

7.5.2.10 Lgsx_MetaCreateInstance()

```
LgsxMetadataPtr Lgsx_MetaCreateInstance ()
```

Creates an instance of metadata container.

Returns

Metadata object. You need to delete it using [Lgsx_FreeHandle](#).

7.5.2.11 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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pValue</i>	Returned value.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.12 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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pValue</i>	Returned value.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.13 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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pValue</i>	Returned value. Need to pre-allocate 3 doubles for returning data.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.14 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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pValue</i>	Returned value. Need to pre-allocate 4 doubles for returning data.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.15 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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pValue</i>	Returned value.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.16 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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pValue</i>	Returned value.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.17 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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pMetadataOut</i>	Returned value which is of metadata type.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, otherwise failed.

7.5.2.18 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 least the actual length plus 1).

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>pValue</i>	Buffer to store value.
in	<i>maxLength</i>	Size of buffer.

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

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

7.5.2.19 Lgsx_MetaGetString2()

```
int Lgsx_MetaGetString2 (
    LgsxMetadataPtr pMetadata,
    const LgsxPropertyName pName,
    wchar_t ** ppValue)
```

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

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.
out	<i>ppValue</i>	Pointer to store pointer to buffer with value. Buffer must be freed usign LgsxUtil_FreeMem() .

Note

The *pName* may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 as success, negative failed.

7.5.2.20 Lgsx_MetaHas()

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

TRUE if the key exists in the metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.

Note

The *pName* may be of any length, [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

1 as true and 0 as false.

7.5.2.21 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

in	<i>pMetadata</i>	Metadata container object.
out	<i>pName</i>	Returned key name. Please reserve 40 chars for returning string.
out	<i>pType</i>	Returned value type. See table for more detail of the returned type.

Deprecated 09.07.2025 - Lgsx_MetaMoveNext2 introduced that does not truncate the name.

Returns

0 for success. None-zero means end.

7.5.2.22 Lgsx_MetaMoveNext2()

```
int Lgsx_MetaMoveNext2 (
    LgsxMetadataPtr pMetadata,
    const char ** ppName,
    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

in	<i>pMetadata</i>	Metadata container object.
out	<i>ppName</i>	Returned key name. The pointer remains valid until next call to <code>Lgsx_MetaMoveNext2</code> or <code>Lgsx_MetaReset</code> .
out	<i>pType</i>	Returned value type. See table for more detail of the returned type.

Returns

0 for success. None-zero means end.

7.5.2.23 Lgsx_MetaRemove()

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

Remove a key pair from metadata.

Parameters

in	<i>pMetadata</i>	Metadata container object.
in	<i>pName</i>	Key name.

Note

The `pName` may be of any length (const char *), [Lgsx_MetaMoveNext2\(\)](#) allows getting non-truncated names.

Returns

0 for success.

7.5.2.24 Lgsx_MetaReset()

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

Resets traverse pointer in the metadata.

This is called before calling `Lgsx_MetaMoveNext`/`Lgsx_MetaMoveNext2`.

Parameters

in	<i>pMetadata</i>	Metadata container object.
----	------------------	----------------------------

Returns

0 for success.

Chapter 8

Data Structure Documentation

8.1 CategoryEntryHandle Struct Reference

Descriptor for a Category Entry object.

```
#include <LgsxReader.h>
```

8.1.1 Detailed Description

Descriptor for a Category Entry object.

8.2 CategoryHandle Struct Reference

Descriptor for a Category object.

```
#include <LgsxReader.h>
```

8.2.1 Detailed Description

Descriptor for a Category object.

8.3 CategoryValueHandle Struct Reference

Descriptor for a Category Value object.

```
#include <LgsxReader.h>
```

8.3.1 Detailed Description

Descriptor for a Category Value object.

8.4 CYASSET Struct Reference

Structure for Asset.

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

Data Fields

- `wchar_t name` [80]
User-facing name of the Asset.
- `wchar_t type` [40]
Asset type.
- `wchar_t mimeType` [60]
A mime type of the Asset.
- `wchar_t filename` [256]
A temporary file name in which the asset data is stored.
- `wchar_t url` [256]
The url address to the external resource.
- `uint64_t id`
The unique identifier of the Asset.
- `uint64_t creationTime`
Creation timestamp (UTC seconds since 1/1/1970).
- `uint64_t modificationTime`
Modification timestamp (UTC seconds since 1/1/1970).
- `bool isExternalUrl`
Marks an asset as external url.

8.4.1 Detailed Description

Structure for Asset.

8.4.2 Field Documentation

8.4.2.1 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.

8.4.2.2 name

```
wchar_t CYASSET::name[80]
```

User-facing 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.

8.4.2.3 type

```
wchar_t CYASSET::type[40]
```

Asset type.

One of the 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).

8.4.2.4 url

```
wchar_t CYASSET::url[256]
```

The url address to the external resource.

Should be filled only when isExternalUrl is true.

8.5 CYBBOX Struct Reference

Structure for bounding box (min, max) corners.

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

Data Fields

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

8.5.1 Detailed Description

Structure for bounding box (min, max) corners.

8.6 CYGEOTAG Struct Reference

Structure for GeoTag.

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

Data Fields

- **PNT3D position**
Location is specified in project coordinates.
- bool **hasCameraPosition**
True if camera position is specified.
- **PNT3D cameraPosition**
Camera position in project coordinates.
- wchar_t **name** [40]
Name of geotag (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.

8.6.1 Detailed Description

Structure for GeoTag.

Note: a GeoTag's position is specified in project coordinates.

Deprecated 26.06.2025 - New GeoTag API available via [GeoTagHandle](#)

8.6.2 Field Documentation

8.6.2.1 setupId

```
uint64_t CYGEOTAG::setupId
```

When specified non-zero, this GeoTag belongs to the Setup.

Note: setupId is expected to be 0 for LGSx files.

8.7 CYMODELINFO Struct Reference

Struct of returned Model info.

```
#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. IFC file, etc.)
- `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.

8.7.1 Detailed Description

Struct of returned Model info.

8.7.2 Field Documentation

8.7.2.1 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.

8.8 CYMODELNODEINFO Struct Reference

Struct of returned ModelNode info.

```
#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.

8.8.1 Detailed Description

Struct of returned ModelNode info.

8.9 CYRANGECUBE Struct Reference

Structure for an arbitrary range cube (a box)

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

Data Fields

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

8.9.1 Detailed Description

Structure for an arbitrary range cube (a box)

8.10 CYRECT Struct Reference

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

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

Data Fields

- int **left**
Left.
- int **top**
Top.
- int **right**
Right.
- int **bottom**
Bottom.

8.10.1 Detailed Description

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

8.11 CYRGBA Struct Reference

Structure for holding the RGBA color channels.

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

Data Fields

- `uint8_t r`
Red channel.
- `uint8_t g`
Green channel.
- `uint8_t b`
Blue channel.
- `uint8_t a`
Alpha channel.

8.11.1 Detailed Description

Structure for holding the RGBA color channels.

8.12 CYSETUPCAMERAINFO Struct Reference

Structure for setup camera info.

```
#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`
Parameters for PINHOLE_CAMERA.
 - `M3DDUALFISHEYE m_DualFisheye`
Parameters for DUAL_FISHEYE_CAMERA.
 - `M3DLUTCAMERA m_Lut`
Parameters for LUT_CAMERA.
 - `M3DFLAT m_Flat`
Parameters for FLAT.
- };
- Camera model parameters (varies by camera model type)*

8.12.1 Detailed Description

Structure for setup camera info.

8.13 CYSETUPINFO Struct Reference

Structure for setup info.

```
#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.

8.13.1 Detailed Description

Structure for setup info.

This has been extended to support both TLS setup and mobile setup (WayPoint along track of run)

8.13.2 Field Documentation

8.13.2.1 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.

8.13.2.2 vertexIndex

```
int CYSETUPINFO::vertexIndex
```

Mobile only: index of the closest trajectory vertex.

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

8.14 CYSITEMAP Struct Reference

Structure for Sitemap.

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

Data Fields

- **uint64_t sitemapId**
ID of the sitemap.
- **wchar_t name** [40]
Sitemap name.
- **uint64_t numSetups**
Number of setups in the sitemap.
- **uint64_t numAssets**
Number of assets referenced by the sitemap.
- **uint64_t backgroundImageId**
Asset ID of the background image (if not zero)
- **uint64_t acceptanceImageId**
Asset ID of the acceptance image (if not zero)
- **uint64_t overviewImageId**
Asset ID of the overview image (if not zero)
- [LgsxMetadataPtr](#) **pMetaData**
Collection of sitemap metadata.

8.14.1 Detailed Description

Structure for Sitemap.

8.15 CYSITEMAPIMAGE Struct Reference

Structure for sitemap image.

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

Data Fields

- **int width**
Number of columns aka the image-width.
- **int height**
Number of rows aka the image-height.
- **int widthInBytes**
Number of bytes that constitute a row.
- [ImagePixelType](#) **pixelType**
Pixel type (image format)
- **unsigned char * pImageBytes**
Image raw data.
- **double world2screen** [16]
World to screen transformation matrix (4x4)
- [SitemapImageType](#) **imageType**
Type of the image.

8.15.1 Detailed Description

Structure for sitemap image.

8.16 CYTARGETINFO Struct Reference

Structure for target info.

```
#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. 0,0,0 means no normal.
- **double radius**
0.0 means none, sphere target needs it
- **wchar_t label [40]**
User-facing target name. Does not need to be unique.

8.16.1 Detailed Description

Structure for target info.

Note that origin and normal are specified in Setup coordinate system. A target always belongs to a Setup.

8.17 CYTRAJECTORYINFO Struct Reference

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

```
#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)
- **uint64_t stopTime**
End timestamp in UTC seconds (accurate end time should be computed from last vertex)
- **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.

8.17.1 Detailed Description

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

8.17.2 Field Documentation

8.17.2.1 startTime

```
uint64_t CYTRAJECTORYINFO::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.

8.17.2.2 stopTime

```
uint64_t CYTRAJECTORYINFO::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.

8.18 CYTRAJECTORYVERTEX Struct Reference

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

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

Data Fields

- `uint64_t time`
Number of ticks since the start of trajectory.
- `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)

8.18.1 Detailed Description

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

8.18.2 Field Documentation

8.18.2.1 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 vertices 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.

8.18.2.2 time

```
uint64_t CYTRAJECTORYVERTEX::time
```

Number of ticks since the start of trajectory.

This times a scale to get actual lapse from trajectory's startTime.

8.19 FieldEntryHandle Struct Reference

Descriptor for a Field Entry object.

```
#include <LgsxReader.h>
```

8.19.1 Detailed Description

Descriptor for a Field Entry object.

8.20 FieldHandle Struct Reference

Descriptor for a Field object.

```
#include <LgsxReader.h>
```

8.20.1 Detailed Description

Descriptor for a Field object.

8.21 GeoTagHandle Struct Reference

GeoTag data handle for accessing the GeoTag data.

```
#include <LgsxReader.h>
```

8.21.1 Detailed Description

GeoTag data handle for accessing the GeoTag data.

8.22 M3DDUALFISHEYE Struct Reference

Structure for M3D dual fish eye camera model.

```
#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**
Radius of the sphere in meters.
- 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.

8.22.1 Detailed Description

Structure for M3D dual fish eye camera model.

8.23 M3DFLAT Struct Reference

Structure for M3D flat camera model.

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

Data Fields

- double **m_Flat** [20]
Flat camera model parameters.

8.23.1 Detailed Description

Structure for M3D flat camera model.

8.24 M3DLUTCAMERA Struct Reference

Structure for M3D LUT camera model.

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

Data Fields

- double **m_scale**
Angle of every pixel.
- double **m_radius**
Radius used for generation of the panorama in meters.
- 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.

8.24.1 Detailed Description

Structure for M3D LUT camera model.

8.25 M3DPINHOLE Struct Reference

Structure for M3D pinhole camera model.

```
#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.

8.25.1 Detailed Description

Structure for M3D pinhole camera model.

8.25.2 Member Enumeration Documentation

8.25.2.1 anonymous enum

anonymous enum

Enumerator

distortionK1	Distortion coefficient K1.
distortionK2	Distortion coefficient K2.
distortionP1	Distortion coefficient P1.
distortionP2	Distortion coefficient P2.
distortionK3	Distortion coefficient K3.
distortionK4	Distortion coefficient K4.
distortionK5	Distortion coefficient K5.
distortionK6	Distortion coefficient K6.
distortionElements	Number of distortion coefficients.

8.26 PNT2D Struct Reference

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

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

Data Fields

- double **x**
X coordinate.
- double **y**
Y coordinate.

8.26.1 Detailed Description

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

8.27 PNT3D Struct Reference

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

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

Data Fields

- double **x**
X coordinate.
- double **y**
Y coordinate.
- double **z**
Z coordinate.

8.27.1 Detailed Description

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

8.28 QUA4D Struct Reference

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

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

Data Fields

- double **u**
Quaternion U component.
- double **v**
Quaternion V component.
- double **w**
Quaternion W component.
- double **s**
Quaternion S component.

8.28.1 Detailed Description

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

8.29 SCyRgdBdyTxf Struct Reference

Rigid transformation represented by a translation and a quaternion.

```
#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.

8.29.1 Detailed Description

Rigid transformation represented by a translation and a quaternion.

Quaternion is represented as rotating around given axis a given angle: (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. The values are normalized: So (mU,mV,mW) = axis * sin(theta/2)), mS = cos(theta/2).

Chapter 9

File Documentation

9.1 /LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/LgsxClient.h File Reference

Data Structures

- struct [CYRGBA](#)
Structure for holding the RGBA color channels.
- struct [PNT3D](#)
Structure for 3D point (or vector, defined as 3 doubles)
- struct [PNT2D](#)
Structure for 2D point (or vector, defined as 2 doubles)
- struct [QUA4D](#)
Structure for Quaternion (or vector, defined as 4 doubles)
- struct [SCyRgdBdyTx](#)
Rigid transformation represented by a translation and a quaternion.
- struct [CYBBOX](#)
Structure for bounding box (min, max) corners.
- struct [CYRECT](#)
Structure for rectangle (left, top, right, bottom)
- struct [CYRANGECUBE](#)
Structure for an arbitrary range cube (a box)
- struct [CYSETUPINFO](#)
Structure for setup info.
- struct [CYTARGETINFO](#)
Structure for target info.
- struct [M3DFLAT](#)
Structure for M3D flat camera model.
- struct [M3DPINHOLE](#)
Structure for M3D pinhole camera model.
- struct [M3DDUALFISHEYE](#)
Structure for M3D dual fish eye camera model.
- struct [M3DLUTCAMERA](#)
Structure for M3D LUT camera model.
- struct [CYSETUPCAMERAINFO](#)

- Structure for setup camera info.*
 - struct [CYGEOTAG](#)
- Structure for GeoTag.*
 - struct [CYSITEMAP](#)
- Structure for Sitemap.*
 - struct [CYSITEMAPIMAGE](#)
- Structure for sitemap image.*
 - struct [CYASSET](#)
- Structure for Asset.*
 - struct [CYTRAJECTORYVERTEX](#)
- Structure for Trajectory (a Run, Walk or Fly) node of mobile scanning (such as BLK2GO)*
 - struct [CYTRAJECTORYINFO](#)
- Structure for track (a Run, Walk or Fly) header info of mobile scanning (such as BLK2GO)*
 - struct [CYMODELNODEINFO](#)
- Struct of returned ModelNode info.*
 - struct [CYMODELINFO](#)
- Struct of returned Model info.*

Macros

- #define **BitmapFormat__NoBitmap** 0
Bitmap format - No bitmap.
- #define **BitmapFormat__RGB** 1
Bitmap format - RGB.
- #define **BitmapFormat__RGBA** 2
Bitmap format - RGBA.
- #define **BitmapFormat__RRGGBBScanlines** 3
Bitmap format - RRGGBB with scanlines.
- #define **DrawPurpose__Summary** 0
Draw purpose - Summary.
- #define **DrawPurpose__Visible** 1
Draw purpose - Visible.
- #define **DrawPurpose__Dynamic** 2
Draw purpose - Dynamic.
- #define **DrawPurpose__Plot** 3
Draw purpose - Plot.
- #define **ScalingPolicy__ByWidth** 0
Scaling policy - By width.
- #define **ScalingPolicy__ByHeight** 1
Scaling policy - By height.
- #define **ScalingPolicy__ByMinimal** 2
Scaling policy - By minimal dimension.
- #define **ScalingPolicy__ByBoth** 3
Scaling policy - By both width and height.
- #define **ColorMappingMode__Unspecified** 0
Color mapping mode - Unspecified.
- #define **ColorMappingMode__ColorFromScanner** 1
Color mapping mode - Color from scanner.
- #define **ColorMappingMode__IntensityMap** 2
Color mapping mode - Intensity.

- **#define ColorMappingMode__ElevationMap 3**
Color mapping mode - Elevation.
- **#define ColorMappingMode__DistanceMap 4**
Color mapping mode - Distance map.
- **#define ColorMappingMode__SingleColor 5**
Color mapping mode - Single color.
- **#define ColorMappingMode__ClassMap 6**
Color mapping mode - Class map.
- **#define ColorMappingMode__MaxValue 6**
Color mapping mode - Maximum mode value.
- **#define ColorMappingScheme__Unspecified 0**
Color mapping scheme - Unspecified.
- **#define ColorMappingScheme__MultiHue 1**
Color mapping scheme - Multihue (multicolor)
- **#define ColorMappingScheme__GrayScale 2**
Color mapping scheme - Gray scale range.
- **#define ColorMappingScheme__Red 3**
Color mapping scheme - Red color range.
- **#define ColorMappingScheme__Green 4**
Color mapping scheme - Green color range.
- **#define ColorMappingScheme__Blue 5**
Color mapping scheme - Blue color range.
- **#define ColorMappingScheme__MaxValue 5**
Color mapping scheme - Maximum scheme value.
- **#define PatchResultShape__ConvexHull 1**
Patch result shape - Convex hull.
- **#define PatchResultShape__Rectangle 2**
Patch result shape - Rectangle.
- **#define SteelSectionType__I 0**
Steel section type - Wide flange (I-beam)
- **#define SteelSectionType__T 1**
Steel section type - Tee.
- **#define SteelSectionType__C 2**
Steel section type - Channel (U-shape)
- **#define SteelSectionType__O 3**
Steel section type - Tubular (rectangular tube)
- **#define SteelSectionType__L 4**
Steel section type - Angle (L-shape)
- **#define PM_XYZ_DATA 1**
Property type mask - XYZ.
- **#define PM_INTENSITY_DATA 2**
Property type mask - Intensity.
- **#define PM_COLOR_DATA 4**
Property type mask - Color.
- **#define PM_NORMAL_DATA 8**
Property type mask - Normal.
- **#define PM_CLASSIFICATION 16**
Property type mask - Classification.
- **#define PM_SETUPIDX 32**
Property type mask - Setup index.
- **#define PM_UserType1 256**

- Property type mask - User type 1.*
 - **#define PM_UserType2** 512
- Property type mask - User type 2.*
 - **#define PM_UserType3** 1024
- Property type mask - User type 3.*
 - **#define PM_UserType4** 2048
- Property type mask - User type 4.*
 - **#define SYNC_STATE** 0
- State of global sync.*
 - **#define SYNC_DATAPOINT** 1
- State of Sync - 3D Point.*
 - **#define SYNC_PICKPOINT** 2
- State of Sync - Pick On Cloud.*
 - **#define SYNC_VIEWPOINT** 3
- State of Sync - View Point.*
 - **#define SYNC_QLIMITBOX** 4
- State of Sync - QLimitBox, QLimitBox range.*
 - **#define targetType_BLACK_AND_WHITE** 0
- Target type - Black and white.*
 - **#define targetType_SPHERE** 1
- Target type - Sphere.*
 - **#define targetType_POINT_3D** 2
- Target type - 3D point.*
 - **#define targetType_BLUE_CIRCULAR** 3
- Target type - Blue circular.*
 - **#define targetType_BLUE_SQUARE** 4
- Target type - Blue square.*
 - **#define targetType_IMAGE_ARUCO** 5
- Target type - ArUco marker.*
 - **#define targetType_UNKNOWN** 6
- Target type - Unknown.*
 - **#define MetadataType_BOOL** 0
- Metadata type - Bool.*
 - **#define MetadataType_INT** 1
- Metadata type - Integer.*
 - **#define MetadataType_INT64** 2
- Metadata type - 64-bit integer.*
 - **#define MetadataType_DOUBLE** 3
- Metadata type - Double.*
 - **#define MetadataType_STRING** 4
- Metadata type - String.*
 - **#define MetadataType_DOUBLE3D** 5
- Metadata type - 3D double (3 double precision numbers)*
 - **#define MetadataType_QUAT4D** 6
- Metadata type - Quaternion (4 double precision numbers)*
 - **#define MetadataType_METADATA** 7
- Metadata type - Metadata (another metadata object)*
 - **#define MetadataType_COMPLEX** 8
- Metadata type - Complex (arbitrary dimensions)*

Typedefs

- typedef void * **LgsxMetadataPtr**
Metadata container object.
- typedef char **LgsxPropertyName**[[PROPERTY_NAME_LENGTH](#)]
Array to store property name.

Enumerations

- enum **ImagePixelType** {
 [ImagePixelType_GRAY](#) ,
 [ImagePixelType_RGB](#) ,
 [ImagePixelType_RGBA](#) ,
 [ImagePixelType_BGR](#) ,
 [ImagePixelType_BGRA](#) ,
 [ImagePixelType_RGBE](#) }
Constants of type [ImagePixelType](#) used for images.
- enum **SitemapImageType** {
 [SitemapImageType_Unknown](#) = 0 ,
 [SitemapImageType_Background](#) ,
 [SitemapImageType_Overview](#) ,
 [SitemapImageType_Acceptance](#) }
Sitemap image type.
- enum **M3DPOSETYPE** {
 [PINHOLE_CAMERA](#) ,
 [LUT_CAMERA](#) ,
 [DUAL_FISHEYE_CAMERA](#) ,
 [FLAT](#) }
M3D camera model type (see [CYSETUPCAMERAINFO.m_type](#))

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_StrDupA** (const char *value, char **ptrDupValue)
Duplicate string.

- `int LgsxUtil_StrDupW (const wchar_t *value, wchar_t **ptrDupValue)`
Duplicate string.
- `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_Initialize (const wchar_t *pTempPath, bool enableLog)`
Initialize the LGSx SDK APIs.
- `int Lgsx_Uninitialize ()`
Unintialize LGSx SDK APIs.
- `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.
- `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.
- `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_MetaMoveNext2](#) ([LgsxMetadataPtr](#) pMetadata, const char **ppName, 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_MetaGetString2](#) ([LgsxMetadataPtr](#) pMetadata, const [LgsxPropertyName](#) pName, wchar_t **ppValue)
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.

Variables

- const unsigned int **PROPERTY_NAME_LENGTH** = 40
Maximum length of property name (including null terminator)

9.1.1 Detailed Description

Copyright

2025, Leica Geosystems, AG

9.1.2 Enumeration Type Documentation

9.1.2.1 ImagePixelFormat

enum [ImagePixelFormat](#)

Constants of type [ImagePixelFormat](#) used for images.

Enumerator

ImagePixelFormat_GRAY	1 byte
ImagePixelFormat_RGB	3 bytes (R is low byte)
ImagePixelFormat_RGBA	4 bytes (R is low byte)
ImagePixelFormat_BGR	3 bytes (B is low byte)
ImagePixelFormat_BGRA	4 bytes (B is low byte)
ImagePixelFormat_RGBE	4 bytes (R is low byte)

9.1.2.2 M3DPOSETYPE

enum [M3DPOSETYPE](#)

M3D camera model type (see [CYSETUPCAMERAINFO.m_type](#))

Enumerator

PINHOLE_CAMERA	Pinhole camera.
LUT_CAMERA	LUT camera.
DUAL_FISHEYE_CAMERA	Dual fishaye camera.
FLAT	Flat.

9.1.2.3 SitemapImageType

enum [SitemapImageType](#)

Sitemap image type.

Enumerator

SitemapImageType_Unknown	Sitemap image type - Unknown.
SitemapImageType_Background	Sitemap image type - Background.
SitemapImageType_Overview	Sitemap image type - Overview.
SitemapImageType_Acceptance	Sitemap image type - Acceptance.

9.2 /LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/LgsxReader.h File Reference

Data Structures

- struct [CategoryHandle](#)
Descriptor for a Category object.
- struct [CategoryValueHandle](#)
Descriptor for a Category Value object.
- struct [CategoryEntryHandle](#)
Descriptor for a Category Entry object.
- struct [FieldHandle](#)
Descriptor for a Field object.
- struct [FieldEntryHandle](#)
Descriptor for a Field Entry object.
- struct [GeoTagHandle](#)
GeoTag data handle for accessing the GeoTag data.

Typedefs

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

Enumerations

- enum [LgsxMigratedMetaKeyField](#) {
 [LgsxMigratedMetaKeyField_Category](#) = 0 ,
 [LgsxMigratedMetaKeyField_Label](#) ,
 [LgsxMigratedMetaKeyField_TagIndex](#) }
Enum for the migrated metadata.

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_ReaderQueryPropertyTypes](#) (LgsxReaderPtr reader)
Returns the Property Types available for points in the project's Point Cloud.
- 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.
- int [Lgsx_ReaderGetLastError2](#) (LgsxReaderPtr reader, char **pErrorPtr)
Returns a string description of the last error that occurred with the given Reader.
- int [Lgsx_ReaderHasPassword](#) (const wchar_t *pFilePath, bool *hasPassword)
Checks if file is password protected.
- 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.
- int [Lgsx_ReaderGetProjectDescription2](#) (LgsxReaderPtr reader, wchar_t **pDescPtr)
Returns the project description string for the opened project.
- int [Lgsx_ReaderEnumAssets](#) (LgsxReaderPtr reader, int *pNumAsset, bool bGetAll)
Collects assets 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 current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_ReaderAssetGetGuid](#) (LgsxReaderPtr reader, const [CYASSET](#) *pAsset, char **pGuidPtr)
Returns the Asset GUID for the asset passed as parameter.
- 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 current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_GetAssetAsImage2](#) (LgsxReaderPtr reader, uint64_t assetId, wchar_t **pNamePtr, int *width, int *height, int *widthInBytes, [ImagePixelType](#) *pixelType, unsigned char **pImage)
Get the "payload" image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_AssetReadImage](#) (const [CYASSET](#) *pAsset, int *pWidth, int *pHeight, int *pWidthInBytes, [ImagePixelType](#) *pPixelType, unsigned char **pImage)
Get the "payload" as image for the given Asset.
- int [Lgsx_AssetReadBinary](#) (const [CYASSET](#) *pAsset, unsigned char **pData, int *pSize)
Get the "payload" as binary data for the given Asset.
- int [Lgsx_GetAssetThumbImage](#) (LgsxReaderPtr reader, uint64_t assetId, int *width, int *height, int *widthInBytes, [ImagePixelType](#) *pixelType, unsigned char **pImage)
Get the thumbnail image for current asset as per [Lgsx_ReaderMoveNextAsset\(\)](#) context.
- int [Lgsx_AssetHasType](#) (LgsxReaderPtr reader, uint64_t assetId, bool *pHasAssetType)
Check if an Asset has a type set.
- int [Lgsx_ReaderEnumCategories](#) (LgsxReaderPtr reader, int *pNumCategories)

- Collects Categories in the project and returns the count.*
- int [Lgsx_ReaderGetCategory](#) ([LgsxReaderPtr](#) reader, int pos, [CategoryHandle](#) *pHandle)
Read Category data at the given position in the active collection.
- int [Lgsx_CategoryRelease](#) ([CategoryHandle](#) *pHandle)
Release the Category data pointer returned by [Lgsx_ReaderGetCategory\(\)](#).
- int [Lgsx_CategoryGetGuid](#) (const [CategoryHandle](#) handle, const char **pId)
Get the GUID of the Category.
- int [Lgsx_CategoryGetName](#) (const [CategoryHandle](#) handle, const wchar_t **pName)
Get the name of the Category.
- int [Lgsx_CategoryGetPriority](#) (const [CategoryHandle](#) handle, int *pPriority)
Get the priority of the Category.
- int [Lgsx_CategoryGetCreationTimestamp](#) (const [CategoryHandle](#) handle, uint64_t *pTimestamp)
Get the creation timestamp of the Category.
- int [Lgsx_CategoryGetModificationTimestamp](#) (const [CategoryHandle](#) handle, uint64_t *pTimestamp)
Get the modification timestamp of the Category.
- int [Lgsx_CategoryEnumValues](#) (const [CategoryHandle](#) handle, int *pNumValues)
Enumerate the values within a Category and return the count.
- int [Lgsx_CategoryGetValue](#) (const [CategoryHandle](#) handle, int pos, [CategoryValueHandle](#) *pValueHandle)
Get a Category Value at the specified position.
- int [Lgsx_CategoryValueGetGuid](#) (const [CategoryValueHandle](#) handle, const char **pId)
Get the GUID of a Category Value.
- int [Lgsx_CategoryValueGetValue](#) (const [CategoryValueHandle](#) handle, const wchar_t **pValue)
Get the value of a Category Value.
- int [Lgsx_CategoryValueGetPriority](#) (const [CategoryValueHandle](#) handle, int *pPriority)
Get the priority of a Category Value.
- int [Lgsx_CategoryValueGetCreationTimestamp](#) (const [CategoryValueHandle](#) handle, uint64_t *pTimestamp)
Get the creation timestamp of a Category Value.
- int [Lgsx_CategoryValueGetModificationTimestamp](#) (const [CategoryValueHandle](#) handle, uint64_t *pTimestamp)
Get the modification timestamp of a Category Value.
- int [Lgsx_CategoryValueHasColor](#) (const [CategoryValueHandle](#) handle, bool *pStatus)
Check if a Category Value has a color associated with it.
- int [Lgsx_CategoryValueGetColor](#) (const [CategoryValueHandle](#) handle, [CYRGBA](#) *pColor)
Get the color of a Category Value.
- int [Lgsx_CategoryEntryGetCategory](#) (const [CategoryEntryHandle](#) handle, [CategoryHandle](#) *pCategoryHandle)
Get the Category from a Category Entry.
- int [Lgsx_CategoryEntryHasValue](#) (const [CategoryEntryHandle](#) handle, bool *pStatus)
Check if a Category Entry has a value associated with it.
- int [Lgsx_CategoryEntryGetValue](#) (const [CategoryEntryHandle](#) handle, [CategoryValueHandle](#) *pValueHandle)
Get the Category Value from a Category Entry.
- int [Lgsx_GetMigratedFieldGuid](#) ([LgsxMigratedMetaKeyField](#) field, const char **pId)
Get the GUID of the migrated Field specified by "category" argument.
- int [Lgsx_ReaderEnumFields](#) ([LgsxReaderPtr](#) reader, int *pNumFields)
Collects Fields in the project and returns the count.
- int [Lgsx_ReaderGetField](#) ([LgsxReaderPtr](#) reader, int pos, [FieldHandle](#) *pHandle)
Read Field data at the given position in the active collection.
- int [Lgsx_FieldRelease](#) ([FieldHandle](#) *pHandle)
Release the Field data pointer returned by [Lgsx_ReaderGetField\(\)](#).
- int [Lgsx_FieldGetGuid](#) (const [FieldHandle](#) handle, const char **pId)

- Get the GUID of the Field.*

 - int [Lgsx_FieldGetName](#) (const [FieldHandle](#) handle, const wchar_t **pName)

Get the name of the Field.
- int [Lgsx_FieldGetPriority](#) (const [FieldHandle](#) handle, int *pPriority)

Get the priority of the Field.
- int [Lgsx_FieldGetCreationTimestamp](#) (const [FieldHandle](#) handle, uint64_t *pTimestamp)

Get the creation timestamp of the Field.
- int [Lgsx_FieldGetModificationTimestamp](#) (const [FieldHandle](#) handle, uint64_t *pTimestamp)

Get the modification timestamp of the Field.
- int [Lgsx_FieldEntryGetField](#) (const [FieldEntryHandle](#) handle, [FieldHandle](#) *pFieldHandle)

Get the Field from a Field Entry.
- int [Lgsx_FieldEntryGetValue](#) (const [FieldEntryHandle](#) handle, const wchar_t **pValue)

Get the value from a Field Entry.
- 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_ReaderGetSetupGuid](#) ([LgsxReaderPtr](#) reader, uint64_t setupId, char **pGuidPtr)

Get GUID of the setup for current setup as per [Lgsx_ReaderMoveNextSetup\(\)](#) context.
- 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_ReaderGetImageLayerNames2](#) ([LgsxReaderPtr](#) reader, wchar_t **pLayersPtr)

Retrieve the Image Layer names in the active Setup.
- int [Lgsx_ReaderGetPanolImage](#) ([LgsxReaderPtr](#) reader, int *width, int *height, int *widthInBytes, [ImagePixelType](#) *pixelType, unsigned char **panolImage)

Get the pano image for the active setup.
- int [Lgsx_ReaderGetCubelImage](#) ([LgsxReaderPtr](#) reader, const wchar_t *pLayer, [SCyRgdBdyTxf](#) *txfFrom↔ Setup, int *width, int *height, int *widthInBytes, [ImagePixelType](#) *pixelType, unsigned char *cubelImages[6])

Get the cubemap corresponding to the given layer name for the active setup.
- 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.
- 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.
- 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.
- int [Lgsx_ReaderGetGeoTagName](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, wchar_t **pNamePtr)
Get name attribute of the GeoTag with the given ID.
- int [Lgsx_ReaderHasGeoTagDescription](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, bool *pStatus)
Check if the GeoTag with the given ID has a description attribute.
- int [Lgsx_ReaderGetGeoTagDescription](#) ([LgsxReaderPtr](#) reader, uint64_t geoTagId, wchar_t **pDescPtr)
Get description attribute of the GeoTag with the given ID.
- int [Lgsx_ReaderGetGeoTag](#) ([LgsxReaderPtr](#) reader, int pos, [GeoTagHandle](#) *pHandle)
Read GeoTag data at the given position in the active collection.
- int [Lgsx_GeoTagRelease](#) ([GeoTagHandle](#) *pHandle)
Release the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetGuid](#) (const [GeoTagHandle](#) handle, const char **pGuidPtr)
Get the GUID of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetName](#) (const [GeoTagHandle](#) handle, const wchar_t **pNamePtr)
Get name attribute of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagHasDescription](#) (const [GeoTagHandle](#) handle, bool *pStatus)
Check if the GeoTag has a description attribute.
- int [Lgsx_GeoTagGetDescription](#) (const [GeoTagHandle](#) handle, const wchar_t **pDescPtr)
Get description attribute of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetPosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pGeoTagPosition)
Get the position of the GeoTag.
- int [Lgsx_GeoTagGetRelativePosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pGeoTagPosition)
Get the relative position of the GeoTag.
- int [Lgsx_GeoTagHasCameraPosition](#) (const [GeoTagHandle](#) handle, bool *pStatus)
Check if the GeoTag has a camera position.
- int [Lgsx_GeoTagGetCameraPosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pCameraPosition)
Get the camera position of the GeoTag.
- int [Lgsx_GeoTagGetCameraRelativePosition](#) (const [GeoTagHandle](#) handle, [PNT3D](#) *pCameraPosition)
Get the relative camera position of the GeoTag.
- int [Lgsx_GeoTagGetSetupIndex](#) (const [GeoTagHandle](#) handle, uint64_t *pSetupIndex)

- Get the Setup Index of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).*

 - int [Lgsx_GeoTagGetCreationTimestamp](#) (const [GeoTagHandle](#) handle, uint64_t *pCreationTimestamp)

Get the creation timestamp of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetModificationTimestamp](#) (const [GeoTagHandle](#) handle, uint64_t *pModificationTimestamp)

Get the modification timestamp of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagGetMetadata](#) (const [GeoTagHandle](#) handle, [LgsxMetadataPtr](#) *pMetadata)

Get the metadata of the GeoTag data pointer returned by [Lgsx_ReaderGetGeoTag\(\)](#).
- int [Lgsx_GeoTagEnumCategoryEntries](#) (const [GeoTagHandle](#) handle, int *pNumEntries)

Enumerate the Category Entries associated with the GeoTag and return the count.
- int [Lgsx_GeoTagGetCategoryEntry](#) (const [GeoTagHandle](#) handle, int pos, [CategoryEntryHandle](#) *pEntryHandle)

Get a Category Entry at the specified position for the GeoTag.
- int [Lgsx_GeoTagEnumFieldEntries](#) (const [GeoTagHandle](#) handle, int *pNumEntries)

Enumerate the Field Entries associated with the GeoTag and return the count.
- int [Lgsx_GeoTagGetFieldEntry](#) (const [GeoTagHandle](#) handle, int pos, [FieldEntryHandle](#) *pEntryHandle)

Get a Field Entry at the specified position for the GeoTag.
- int [Lgsx_GeoTagEnumAssets](#) (const [GeoTagHandle](#) handle, int *pNumAssets)

Collects Assets associated with the given GeoTag and returns the count.
- int [Lgsx_ReaderGetGeoTagAsset](#) ([LgsxReaderPtr](#) reader, const [GeoTagHandle](#) handle, int pos, [CYASSET](#) *pAsset, [LgsxMetadataPtr](#) *pMetadata)

Get Info and metadata for the GeoTag Asset at the given position.
- int [Lgsx_ReaderGetGeoTagAssetThumbnail](#) ([LgsxReaderPtr](#) reader, const [GeoTagHandle](#) handle, int pos, int *pWidth, int *pHeight, int *pWidthInBytes, [ImagePixelFormat](#) *pPixelFormat, unsigned char **pImage)

Get the thumbnail image for GeoTag Asset at the given position.
- int [Lgsx_GeoTagHasThumbnail](#) (const [GeoTagHandle](#) handle, bool *pHasThumbnail)

Checks if GeoTag has a thumbnail.
- int [Lgsx_GeoTagGetThumbnail](#) (const [GeoTagHandle](#) handle, int *pWidth, int *pHeight, int *pWidthInBytes, [ImagePixelFormat](#) *pPixelFormat, unsigned char **pImage)

Get the thumbnail image for the GeoTag itself.
- int [Lgsx_ReaderEnumSitemaps](#) ([LgsxReaderPtr](#) reader, int *pNumSitemap)

Collects Sitemaps in the project and returns the count.
- int [Lgsx_ReaderEnumSiteMaps](#) ([LgsxReaderPtr](#) reader, int *pNumSitemap)

Collects Sitemaps in the project and returns the count.
- int [Lgsx_ReaderMoveNextSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t *sitemapId, [CYSITEMAP](#) *pSitemapInfo, [LgsxMetadataPtr](#) *pMetadata)

Advance to the next Sitemap in the active collection and return its ID, info, and metadata.
- int [Lgsx_ReaderGetSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, [CYSITEMAP](#) *pSitemapInfo)

Get Info and metadata for the given Sitemap ID.
- int [Lgsx_ReaderGetSitemapImage](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, [CYSITEMAPIMAGE](#) *pSitemapImageInfo)

Get the image for the given Sitemap ID.
- int [Lgsx_ReaderMoveNextSitemapImage](#) ([LgsxReaderPtr](#) reader, int *width, int *height, int *widthInBytes, [ImagePixelFormat](#) *pixelType, unsigned char **pImage, double world2screen[16])

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

Advance to the next Sitemap in the active collection and return its background image.
- int [Lgsx_ReaderEnumSetupsForSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, int *pNumSetups)

Collects Setups in the project that are associated with the given Sitemap and returns the count.
- int [Lgsx_ReaderMoveNextSetupForSitemap](#) ([LgsxReaderPtr](#) reader, uint64_t sitemapId, uint64_t *pSetupId, [CYSETUPINFO](#) *pSetup, [LgsxMetadataPtr](#) *pMetadata)

- Advance to the next Setup in the active collection and return its ID, info, and metadata.*

 - 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.
 - 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_ReaderEnumPointsEx](#) ([LgsxReaderPtr](#) reader, int desiredTypes, int subSample, bool visible, bool scriptOrder)

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 desired 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.
 - 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.
 - int [Lgsx_ReaderMoveNextCoordSystems2](#) ([LgsxReaderPtr](#) reader, bool *pActive, wchar_t **ppNamePtr, wchar_t **ppWktPtr, [SCyRgdBdyTxf](#) *pTxf)

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

9.2.1 Detailed Description

Copyright

2025, Leica Geosystems, AG

Index

- [/LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/LgsxClient.h](#), [CategoryEntryHandle](#), [105](#)
- [123](#)
- [/LeicaProjectSdk/LgsSdkClient/inc/LgsxSdk/LgsxReader.h](#), [CategoryValueHandle](#), [105](#)
- [131](#)
- Application Functions, [87](#)
 - [Lgsx_Initialize](#), [87](#)
 - [Lgsx_Uninitialize](#), [87](#)
- Asset Functions, [22](#)
 - [Lgsx_AssetHasType](#), [23](#)
 - [Lgsx_AssetReadBinary](#), [23](#)
 - [Lgsx_AssetReadImage](#), [23](#)
 - [Lgsx_GetAssetAsImage](#), [24](#)
 - [Lgsx_GetAssetAsImage2](#), [24](#)
 - [Lgsx_GetAssetThumbImage](#), [25](#)
 - [Lgsx_ReaderAssetGetGuid](#), [25](#)
 - [Lgsx_ReaderEnumAssets](#), [26](#)
 - [Lgsx_ReaderGetAsset](#), [26](#)
 - [Lgsx_ReaderMoveNextAsset](#), [27](#)
- Basic Reader Functions, [18](#)
 - [Lgsx_ReaderClose](#), [18](#)
 - [Lgsx_ReaderCreate](#), [18](#)
 - [Lgsx_ReaderGetLastError](#), [19](#)
 - [Lgsx_ReaderGetLastError2](#), [19](#)
 - [Lgsx_ReaderHasPassword](#), [19](#)
 - [Lgsx_ReaderOpen](#), [20](#)
- Categories Functions, [27](#)
 - [Lgsx_CategoryEntryGetCategory](#), [29](#)
 - [Lgsx_CategoryEntryGetValue](#), [29](#)
 - [Lgsx_CategoryEntryHasValue](#), [29](#)
 - [Lgsx_CategoryEnumValues](#), [30](#)
 - [Lgsx_CategoryGetCreationTimestamp](#), [30](#)
 - [Lgsx_CategoryGetGuid](#), [30](#)
 - [Lgsx_CategoryGetModificationTimestamp](#), [31](#)
 - [Lgsx_CategoryGetName](#), [31](#)
 - [Lgsx_CategoryGetPriority](#), [31](#)
 - [Lgsx_CategoryGetValue](#), [32](#)
 - [Lgsx_CategoryRelease](#), [32](#)
 - [Lgsx_CategoryValueGetColor](#), [32](#)
 - [Lgsx_CategoryValueGetCreationTimestamp](#), [33](#)
 - [Lgsx_CategoryValueGetGuid](#), [33](#)
 - [Lgsx_CategoryValueGetModificationTimestamp](#), [33](#)
 - [Lgsx_CategoryValueGetPriority](#), [34](#)
 - [Lgsx_CategoryValueGetValue](#), [34](#)
 - [Lgsx_CategoryValueHasColor](#), [34](#)
 - [Lgsx_ReaderEnumCategories](#), [35](#)
 - [Lgsx_ReaderGetCategory](#), [35](#)
- [CategoryEntryHandle](#), [105](#)
- [CategoryHandle](#), [105](#)
- [CategoryValueHandle](#), [105](#)
- [CYASSET](#), [106](#)
 - [filename](#), [106](#)
 - [name](#), [106](#)
 - [type](#), [106](#)
 - [url](#), [107](#)
- [CYBBOX](#), [107](#)
- [CYGEOTAG](#), [107](#)
 - [setupId](#), [108](#)
- [CYMODELINFO](#), [108](#)
 - [sceneRepId](#), [109](#)
- [CYMODELNODEINFO](#), [109](#)
- [CYRANGECUBE](#), [110](#)
- [CYRECT](#), [110](#)
- [CYRGBA](#), [111](#)
- [CYSETUPCAMERAINFO](#), [111](#)
- [CYSETUPINFO](#), [112](#)
 - [time](#), [112](#)
 - [vertexIndex](#), [112](#)
- [CYSITEMAP](#), [113](#)
- [CYSITEMAPIMAGE](#), [113](#)
- [CYTARGETINFO](#), [114](#)
- [CYTRAJECTORYINFO](#), [114](#)
 - [startTime](#), [115](#)
 - [stopTime](#), [115](#)
- [CYTRAJECTORYVERTEX](#), [115](#)
 - [setupIndex](#), [116](#)
 - [time](#), [116](#)
- Deprecated List, [9](#)
 - distortionElements
 - [M3DPINHOLE](#), [119](#)
 - distortionK1
 - [M3DPINHOLE](#), [119](#)
 - distortionK2
 - [M3DPINHOLE](#), [119](#)
 - distortionK3
 - [M3DPINHOLE](#), [119](#)
 - distortionK4
 - [M3DPINHOLE](#), [119](#)
 - distortionK5
 - [M3DPINHOLE](#), [119](#)
 - distortionK6
 - [M3DPINHOLE](#), [119](#)
 - distortionP1
 - [M3DPINHOLE](#), [119](#)
 - distortionP2
 - [M3DPINHOLE](#), [119](#)

- DUAL_FISHEYE_CAMERA
 - LgsxCliet.h, [130](#)
- FieldEntryHandle, [116](#)
- FieldHandle, [116](#)
- Fields Functions, [36](#)
 - Lgsx_FieldEntryGetField, [36](#)
 - Lgsx_FieldEntryGetValue, [37](#)
 - Lgsx_FieldGetCreationTimestamp, [37](#)
 - Lgsx_FieldGetGuid, [37](#)
 - Lgsx_FieldGetModificationTimestamp, [38](#)
 - Lgsx_FieldGetName, [38](#)
 - Lgsx_FieldGetPriority, [38](#)
 - Lgsx_FieldRelease, [39](#)
 - Lgsx_ReaderEnumFields, [39](#)
 - Lgsx_ReaderGetField, [39](#)
- filename
 - CYASSET, [106](#)
- FLAT
 - LgsxCliet.h, [130](#)
- General Functions, [79](#)
 - Lgsx_FreeHandle, [80](#)
 - Lgsx_GetLastError, [80](#)
 - Lgsx_GetLastErrorNo, [81](#)
 - Lgsx_GetProductVersion, [81](#)
 - Lgsx_InitProductBuildNumber, [81](#)
 - Lgsx_InitProductName, [82](#)
 - Lgsx_InitProductVersion, [82](#)
 - Lgsx_InitProductVersionEx, [82](#)
 - LgsxUtil_A2W, [83](#)
 - LgsxUtil_AllocMem, [83](#)
 - LgsxUtil_FreeMem, [83](#)
 - LgsxUtil_GetCurrentDateString, [84](#)
 - LgsxUtil_GetCurrentTimeString, [84](#)
 - LgsxUtil_Sleep, [84](#)
 - LgsxUtil_StrDupA, [84](#)
 - LgsxUtil_StrDupW, [85](#)
 - LgsxUtil_TimeEnd, [85](#)
 - LgsxUtil_TimeGetLapse, [85](#)
 - LgsxUtil_TimeGetLapseStr, [86](#)
 - LgsxUtil_TimeStart, [86](#)
 - LgsxUtil_W2A, [86](#)
- GeoTag Functions, [53](#)
 - Lgsx_GeoTagEnumAssets, [54](#)
 - Lgsx_GeoTagEnumCategoryEntries, [54](#)
 - Lgsx_GeoTagEnumFieldEntries, [55](#)
 - Lgsx_GeoTagGetCameraPosition, [55](#)
 - Lgsx_GeoTagGetCameraRelativePosition, [55](#)
 - Lgsx_GeoTagGetCategoryEntry, [56](#)
 - Lgsx_GeoTagGetCreationTimestamp, [56](#)
 - Lgsx_GeoTagGetDescription, [56](#)
 - Lgsx_GeoTagGetFieldEntry, [57](#)
 - Lgsx_GeoTagGetGuid, [57](#)
 - Lgsx_GeoTagGetMetadata, [57](#)
 - Lgsx_GeoTagGetModificationTimestamp, [58](#)
 - Lgsx_GeoTagGetName, [58](#)
 - Lgsx_GeoTagGetPosition, [58](#)
 - Lgsx_GeoTagGetRelativePosition, [59](#)
 - Lgsx_GeoTagGetSetupIndex, [59](#)
 - Lgsx_GeoTagGetThumbnail, [59](#)
 - Lgsx_GeoTagHasCameraPosition, [60](#)
 - Lgsx_GeoTagHasDescription, [60](#)
 - Lgsx_GeoTagHasThumbnail, [61](#)
 - Lgsx_GeoTagRelease, [61](#)
 - Lgsx_ReaderEnumGeoTags, [61](#)
 - Lgsx_ReaderEnumGeoTagsOfSetup, [62](#)
 - Lgsx_ReaderGetGeoTag, [62](#)
 - Lgsx_ReaderGetGeoTagAsset, [62](#)
 - Lgsx_ReaderGetGeoTagAssetThumbnail, [63](#)
 - Lgsx_ReaderGetGeoTagDescription, [63](#)
 - Lgsx_ReaderGetGeoTagName, [64](#)
 - Lgsx_ReaderHasGeoTagDescription, [64](#)
 - Lgsx_ReaderMoveNextGeoTag, [65](#)
- GeoTagHandle, [116](#)
- ImagePixelType
 - LgsxCliet.h, [130](#)
- ImagePixelType_BGR
 - LgsxCliet.h, [130](#)
- ImagePixelType_BGRA
 - LgsxCliet.h, [130](#)
- ImagePixelType_GRAY
 - LgsxCliet.h, [130](#)
- ImagePixelType_RGB
 - LgsxCliet.h, [130](#)
- ImagePixelType_RGBA
 - LgsxCliet.h, [130](#)
- ImagePixelType_RGBE
 - LgsxCliet.h, [130](#)
- Leica LGSx SDK Getting Started Guide, [1](#)
- Lgsx_AssetHasType
 - Asset Functions, [23](#)
- Lgsx_AssetReadBinary
 - Asset Functions, [23](#)
- Lgsx_AssetReadImage
 - Asset Functions, [23](#)
- Lgsx_CategoryEntryGetCategory
 - Categories Functions, [29](#)
- Lgsx_CategoryEntryGetValue
 - Categories Functions, [29](#)
- Lgsx_CategoryEntryHasValue
 - Categories Functions, [29](#)
- Lgsx_CategoryEnumValues
 - Categories Functions, [30](#)
- Lgsx_CategoryGetCreationTimestamp
 - Categories Functions, [30](#)
- Lgsx_CategoryGetGuid
 - Categories Functions, [30](#)
- Lgsx_CategoryGetModificationTimestamp
 - Categories Functions, [31](#)
- Lgsx_CategoryGetName
 - Categories Functions, [31](#)
- Lgsx_CategoryGetPriority
 - Categories Functions, [31](#)
- Lgsx_CategoryGetValue
 - Categories Functions, [32](#)

- Lgsx_CategoryRelease
 - Categories Functions, [32](#)
- Lgsx_CategoryValueGetColor
 - Categories Functions, [32](#)
- Lgsx_CategoryValueGetCreationTimestamp
 - Categories Functions, [33](#)
- Lgsx_CategoryValueGetGuid
 - Categories Functions, [33](#)
- Lgsx_CategoryValueGetModificationTimestamp
 - Categories Functions, [33](#)
- Lgsx_CategoryValueGetPriority
 - Categories Functions, [34](#)
- Lgsx_CategoryValueGetValue
 - Categories Functions, [34](#)
- Lgsx_CategoryValueHasColor
 - Categories Functions, [34](#)
- Lgsx_FieldEntryGetField
 - Fields Functions, [36](#)
- Lgsx_FieldEntryGetValue
 - Fields Functions, [37](#)
- Lgsx_FieldGetCreationTimestamp
 - Fields Functions, [37](#)
- Lgsx_FieldGetGuid
 - Fields Functions, [37](#)
- Lgsx_FieldGetModificationTimestamp
 - Fields Functions, [38](#)
- Lgsx_FieldGetName
 - Fields Functions, [38](#)
- Lgsx_FieldGetPriority
 - Fields Functions, [38](#)
- Lgsx_FieldRelease
 - Fields Functions, [39](#)
- Lgsx_FreeHandle
 - General Functions, [80](#)
- Lgsx_GeoTagEnumAssets
 - GeoTag Functions, [54](#)
- Lgsx_GeoTagEnumCategoryEntries
 - GeoTag Functions, [54](#)
- Lgsx_GeoTagEnumFieldEntries
 - GeoTag Functions, [55](#)
- Lgsx_GeoTagGetCameraPosition
 - GeoTag Functions, [55](#)
- Lgsx_GeoTagGetCameraRelativePosition
 - GeoTag Functions, [55](#)
- Lgsx_GeoTagGetCategoryEntry
 - GeoTag Functions, [56](#)
- Lgsx_GeoTagGetCreationTimestamp
 - GeoTag Functions, [56](#)
- Lgsx_GeoTagGetDescription
 - GeoTag Functions, [56](#)
- Lgsx_GeoTagGetFieldEntry
 - GeoTag Functions, [57](#)
- Lgsx_GeoTagGetGuid
 - GeoTag Functions, [57](#)
- Lgsx_GeoTagGetMetadata
 - GeoTag Functions, [57](#)
- Lgsx_GeoTagGetModificationTimestamp
 - GeoTag Functions, [58](#)
- Lgsx_GeoTagGetName
 - GeoTag Functions, [58](#)
- Lgsx_GeoTagGetPosition
 - GeoTag Functions, [58](#)
- Lgsx_GeoTagGetRelativePosition
 - GeoTag Functions, [59](#)
- Lgsx_GeoTagGetSetupIndex
 - GeoTag Functions, [59](#)
- Lgsx_GeoTagGetThumbnail
 - GeoTag Functions, [59](#)
- Lgsx_GeoTagHasCameraPosition
 - GeoTag Functions, [60](#)
- Lgsx_GeoTagHasDescription
 - GeoTag Functions, [60](#)
- Lgsx_GeoTagHasThumbnail
 - GeoTag Functions, [61](#)
- Lgsx_GeoTagRelease
 - GeoTag Functions, [61](#)
- Lgsx_GetAssetAsImage
 - Asset Functions, [24](#)
- Lgsx_GetAssetAsImage2
 - Asset Functions, [24](#)
- Lgsx_GetAssetThumbImage
 - Asset Functions, [25](#)
- Lgsx_GetLastError
 - General Functions, [80](#)
- Lgsx_GetLastErrorNo
 - General Functions, [81](#)
- Lgsx_GetMigratedFieldGuid
 - Migrated Fields, [41](#)
- Lgsx_GetProductVersion
 - General Functions, [81](#)
- Lgsx_Initialize
 - Application Functions, [87](#)
- Lgsx_InitLicense
 - Licensing Functions, [88](#)
- Lgsx_InitLicenseEx
 - Licensing Functions, [88](#)
- Lgsx_InitLicenseEx2
 - Licensing Functions, [89](#)
- Lgsx_InitProductBuildNumber
 - General Functions, [81](#)
- Lgsx_InitProductName
 - General Functions, [82](#)
- Lgsx_InitProductVersion
 - General Functions, [82](#)
- Lgsx_InitProductVersionEx
 - General Functions, [82](#)
- Lgsx_IsLicensed
 - Licensing Functions, [89](#)
- Lgsx_IsLicensedExA
 - Licensing Functions, [90](#)
- Lgsx_LicenseDaysLeft
 - Licensing Functions, [90](#)
- Lgsx_LicenseDaysLeftExA
 - Licensing Functions, [90](#)
- Lgsx_LoadLicense
 - Licensing Functions, [90](#)

- Lgsx_LoadLicenseExA
 - Licensing Functions, [91](#)
- Lgsx_MetaAddBool
 - Metadata Functions, [93](#)
- Lgsx_MetaAddDouble
 - Metadata Functions, [93](#)
- Lgsx_MetaAddDouble3
 - Metadata Functions, [94](#)
- Lgsx_MetaAddDouble4
 - Metadata Functions, [94](#)
- Lgsx_MetaAddInt
 - Metadata Functions, [95](#)
- Lgsx_MetaAddInt64
 - Metadata Functions, [95](#)
- Lgsx_MetaAddMetadata
 - Metadata Functions, [96](#)
- Lgsx_MetaAddString
 - Metadata Functions, [96](#)
- Lgsx_MetaClone
 - Metadata Functions, [97](#)
- Lgsx_MetaCreateInstance
 - Metadata Functions, [97](#)
- Lgsx_MetaGetBool
 - Metadata Functions, [97](#)
- Lgsx_MetaGetDouble
 - Metadata Functions, [98](#)
- Lgsx_MetaGetDouble3
 - Metadata Functions, [98](#)
- Lgsx_MetaGetDouble4
 - Metadata Functions, [99](#)
- Lgsx_MetaGetInt
 - Metadata Functions, [99](#)
- Lgsx_MetaGetInt64
 - Metadata Functions, [100](#)
- Lgsx_MetaGetMetadata
 - Metadata Functions, [100](#)
- Lgsx_MetaGetString
 - Metadata Functions, [101](#)
- Lgsx_MetaGetString2
 - Metadata Functions, [101](#)
- Lgsx_MetaHas
 - Metadata Functions, [102](#)
- Lgsx_MetaMoveNext
 - Metadata Functions, [102](#)
- Lgsx_MetaMoveNext2
 - Metadata Functions, [103](#)
- Lgsx_MetaRemove
 - Metadata Functions, [104](#)
- Lgsx_MetaReset
 - Metadata Functions, [104](#)
- Lgsx_ReaderAssetGetGuid
 - Asset Functions, [25](#)
- Lgsx_ReaderClose
 - Basic Reader Functions, [18](#)
- Lgsx_ReaderCreate
 - Basic Reader Functions, [18](#)
- Lgsx_ReaderEndRuns
 - Run Functions, [49](#)
- Lgsx_ReaderEndSetups
 - Setup Functions, [41](#)
- Lgsx_ReaderEnumAssets
 - Asset Functions, [26](#)
- Lgsx_ReaderEnumCategories
 - Categories Functions, [35](#)
- Lgsx_ReaderEnumCoordSystems
 - User Coordinate System Functions, [78](#)
- Lgsx_ReaderEnumFields
 - Fields Functions, [39](#)
- Lgsx_ReaderEnumGeoTags
 - GeoTag Functions, [61](#)
- Lgsx_ReaderEnumGeoTagsOfSetup
 - GeoTag Functions, [62](#)
- Lgsx_ReaderEnumPoints
 - Point Cloud Functions, [73](#)
- Lgsx_ReaderEnumPointsEx
 - Point Cloud Functions, [74](#)
- Lgsx_ReaderEnumRuns
 - Run Functions, [49](#)
- Lgsx_ReaderEnumSetupCamera
 - Run Functions, [49](#)
- Lgsx_ReaderEnumSetupCameraOfSetup
 - Run Functions, [50](#)
- Lgsx_ReaderEnumSetups
 - Setup Functions, [42](#)
- Lgsx_ReaderEnumSetupsEx
 - Setup Functions, [42](#)
- Lgsx_ReaderEnumSetupsForSitemap
 - Sitemap Functions, [66](#)
- Lgsx_ReaderEnumSiteMaps
 - Sitemap Functions, [66](#)
- Lgsx_ReaderEnumSitemaps
 - Sitemap Functions, [67](#)
- Lgsx_ReaderEnumTarget
 - Target Functions, [47](#)
- Lgsx_ReaderEnumTargetOfSetup
 - Target Functions, [47](#)
- Lgsx_ReaderExtractPointCloud
 - Point Cloud Functions, [74](#)
- Lgsx_ReaderGetAsset
 - Asset Functions, [26](#)
- Lgsx_ReaderGetCategory
 - Categories Functions, [35](#)
- Lgsx_ReaderGetClassModels
 - Point Cloud Functions, [75](#)
- Lgsx_ReaderGetClassVisibles
 - Point Cloud Functions, [75](#)
- Lgsx_ReaderGetCubelImage
 - Setup Functions, [42](#)
- Lgsx_ReaderGetField
 - Fields Functions, [39](#)
- Lgsx_ReaderGetGeoTag
 - GeoTag Functions, [62](#)
- Lgsx_ReaderGetGeoTagAsset
 - GeoTag Functions, [62](#)
- Lgsx_ReaderGetGeoTagAssetThumbnail
 - GeoTag Functions, [63](#)

- Lgsx_ReaderGetGeoTagDescription
 - GeoTag Functions, [63](#)
- Lgsx_ReaderGetGeoTagName
 - GeoTag Functions, [64](#)
- Lgsx_ReaderGetImageLayerNames
 - Setup Functions, [43](#)
- Lgsx_ReaderGetImageLayerNames2
 - Setup Functions, [44](#)
- Lgsx_ReaderGetLastError
 - Basic Reader Functions, [19](#)
- Lgsx_ReaderGetLastError2
 - Basic Reader Functions, [19](#)
- Lgsx_ReaderGetLUTImage
 - Run Functions, [50](#)
- Lgsx_ReaderGetLUTImageOfSetup
 - Run Functions, [50](#)
- Lgsx_ReaderGetMetadata
 - Project Functions, [20](#)
- Lgsx_ReaderGetModelInfo
 - Model and Model Node Functions, [70](#)
- Lgsx_ReaderGetModelNodeInfo
 - Model and Model Node Functions, [72](#)
- Lgsx_ReaderGetModelNodes
 - Model and Model Node Functions, [72](#)
- Lgsx_ReaderGetPanolImage
 - Setup Functions, [44](#)
- Lgsx_ReaderGetProjectDescription
 - Project Functions, [21](#)
- Lgsx_ReaderGetProjectDescription2
 - Project Functions, [21](#)
- Lgsx_ReaderGetProjectInfo
 - Project Functions, [21](#)
- Lgsx_ReaderGetPropertyTypeInfo
 - Point Cloud Functions, [75](#)
- Lgsx_ReaderGetSetupCameraImage
 - Run Functions, [51](#)
- Lgsx_ReaderGetSetupGuid
 - Setup Functions, [46](#)
- Lgsx_ReaderGetSitemap
 - Sitemap Functions, [67](#)
- Lgsx_ReaderGetSitemapImage
 - Sitemap Functions, [67](#)
- Lgsx_ReaderHasGeoTagDescription
 - GeoTag Functions, [64](#)
- Lgsx_ReaderHasPassword
 - Basic Reader Functions, [19](#)
- Lgsx_ReaderMoveNextAsset
 - Asset Functions, [27](#)
- Lgsx_ReaderMoveNextCoordSystems
 - User Coordinate System Functions, [78](#)
- Lgsx_ReaderMoveNextCoordSystems2
 - User Coordinate System Functions, [78](#)
- Lgsx_ReaderMoveNextFields
 - Point Cloud Functions, [76](#)
- Lgsx_ReaderMoveNextGeoTag
 - GeoTag Functions, [65](#)
- Lgsx_ReaderMoveNextPoints
 - Point Cloud Functions, [76](#)
- Lgsx_ReaderMoveNextRun
 - Run Functions, [51](#)
- Lgsx_ReaderMoveNextSetup
 - Setup Functions, [46](#)
- Lgsx_ReaderMoveNextSetupCamera
 - Run Functions, [52](#)
- Lgsx_ReaderMoveNextSetupForSitemap
 - Sitemap Functions, [68](#)
- Lgsx_ReaderMoveNextSitemap
 - Sitemap Functions, [68](#)
- Lgsx_ReaderMoveNextSiteMapImage
 - Sitemap Functions, [69](#)
- Lgsx_ReaderMoveNextSitemapImage
 - Sitemap Functions, [69](#)
- Lgsx_ReaderMoveNextTarget
 - Target Functions, [47](#)
- Lgsx_ReaderOpen
 - Basic Reader Functions, [20](#)
- Lgsx_ReaderQueryPropertyTypes
 - Point Cloud Functions, [77](#)
- Lgsx_Uninitialize
 - Application Functions, [87](#)
- Lgsx_UnloadLicense
 - Licensing Functions, [91](#)
- Lgsx_UnloadLicenseExA
 - Licensing Functions, [91](#)
- LgsxCliet.h
 - DUAL_FISHEYE_CAMERA, [130](#)
 - FLAT, [130](#)
 - ImagePixelFormat, [130](#)
 - ImagePixelFormat_BGR, [130](#)
 - ImagePixelFormat_BGRA, [130](#)
 - ImagePixelFormat_GRAY, [130](#)
 - ImagePixelFormat_RGB, [130](#)
 - ImagePixelFormat_RGBA, [130](#)
 - ImagePixelFormat_RGBE, [130](#)
 - LUT_CAMERA, [130](#)
 - M3DPOSETYPE, [130](#)
 - PINHOLE_CAMERA, [130](#)
 - SitemapImageType, [130](#)
 - SitemapImageType_Acceptance, [131](#)
 - SitemapImageType_Background, [131](#)
 - SitemapImageType_Overview, [131](#)
 - SitemapImageType_Unknown, [131](#)
- LgsxMigratedMetaKeyField
 - Migrated Fields, [40](#)
- LgsxMigratedMetaKeyField_Category
 - Migrated Fields, [40](#)
- LgsxMigratedMetaKeyField_Label
 - Migrated Fields, [40](#)
- LgsxMigratedMetaKeyField_TagIndex
 - Migrated Fields, [40](#)
- LgsxUtil_A2W
 - General Functions, [83](#)
- LgsxUtil_AllocMem
 - General Functions, [83](#)
- LgsxUtil_FreeMem
 - General Functions, [83](#)

- LgsxUtil_GetCurrentDateStr
 - General Functions, [84](#)
- LgsxUtil_GetCurrentTimeString
 - General Functions, [84](#)
- LgsxUtil_Sleep
 - General Functions, [84](#)
- LgsxUtil_StrDupA
 - General Functions, [84](#)
- LgsxUtil_StrDupW
 - General Functions, [85](#)
- LgsxUtil_TimeEnd
 - General Functions, [85](#)
- LgsxUtil_TimeGetLapse
 - General Functions, [85](#)
- LgsxUtil_TimeGetLapseStr
 - General Functions, [86](#)
- LgsxUtil_TimeStart
 - General Functions, [86](#)
- LgsxUtil_W2A
 - General Functions, [86](#)
- Licensing Functions, [88](#)
 - Lgsx_InitLicense, [88](#)
 - Lgsx_InitLicenseEx, [88](#)
 - Lgsx_InitLicenseEx2, [89](#)
 - Lgsx_IsLicensed, [89](#)
 - Lgsx_IsLicensedExA, [90](#)
 - Lgsx_LicenseDaysLeft, [90](#)
 - Lgsx_LicenseDaysLeftExA, [90](#)
 - Lgsx_LoadLicense, [90](#)
 - Lgsx_LoadLicenseExA, [91](#)
 - Lgsx_UnloadLicense, [91](#)
 - Lgsx_UnloadLicenseExA, [91](#)
- LUT_CAMERA
 - LgsxClient.h, [130](#)
- M3DDUALFISHEYE, [117](#)
- M3DFLAT, [117](#)
- M3DLUTCAMERA, [118](#)
- M3DPINHOLE, [118](#)
 - distortionElements, [119](#)
 - distortionK1, [119](#)
 - distortionK2, [119](#)
 - distortionK3, [119](#)
 - distortionK4, [119](#)
 - distortionK5, [119](#)
 - distortionK6, [119](#)
 - distortionP1, [119](#)
 - distortionP2, [119](#)
- M3DPOSETYPE
 - LgsxClient.h, [130](#)
- Metadata Functions, [92](#)
 - Lgsx_MetaAddBool, [93](#)
 - Lgsx_MetaAddDouble, [93](#)
 - Lgsx_MetaAddDouble3, [94](#)
 - Lgsx_MetaAddDouble4, [94](#)
 - Lgsx_MetaAddInt, [95](#)
 - Lgsx_MetaAddInt64, [95](#)
 - Lgsx_MetaAddMetadata, [96](#)
 - Lgsx_MetaAddString, [96](#)
 - Lgsx_MetaClone, [97](#)
 - Lgsx_MetaCreateInstance, [97](#)
 - Lgsx_MetaGetBool, [97](#)
 - Lgsx_MetaGetDouble, [98](#)
 - Lgsx_MetaGetDouble3, [98](#)
 - Lgsx_MetaGetDouble4, [99](#)
 - Lgsx_MetaGetInt, [99](#)
 - Lgsx_MetaGetInt64, [100](#)
 - Lgsx_MetaGetMetadata, [100](#)
 - Lgsx_MetaGetString, [101](#)
 - Lgsx_MetaGetString2, [101](#)
 - Lgsx_MetaHas, [102](#)
 - Lgsx_MetaMoveNext, [102](#)
 - Lgsx_MetaMoveNext2, [103](#)
 - Lgsx_MetaRemove, [104](#)
 - Lgsx_MetaReset, [104](#)
- Migrated Fields, [40](#)
 - Lgsx_GetMigratedFieldGuid, [41](#)
 - LgsxMigratedMetaKeyField, [40](#)
 - LgsxMigratedMetaKeyField_Category, [40](#)
 - LgsxMigratedMetaKeyField_Label, [40](#)
 - LgsxMigratedMetaKeyField_TagIndex, [40](#)
- Model and Model Node Functions, [70](#)
 - Lgsx_ReaderGetModelInfo, [70](#)
 - Lgsx_ReaderGetModelNodeInfo, [72](#)
 - Lgsx_ReaderGetModelNodes, [72](#)
- name
 - CYASSET, [106](#)
- PINHOLE_CAMERA
 - LgsxClient.h, [130](#)
- PNT2D, [119](#)
- PNT3D, [120](#)
- Point Cloud Functions, [73](#)
 - Lgsx_ReaderEnumPoints, [73](#)
 - Lgsx_ReaderEnumPointsEx, [74](#)
 - Lgsx_ReaderExtractPointCloud, [74](#)
 - Lgsx_ReaderGetClassModels, [75](#)
 - Lgsx_ReaderGetClassVisibles, [75](#)
 - Lgsx_ReaderGetPropertyTypeInfo, [75](#)
 - Lgsx_ReaderMoveNextFields, [76](#)
 - Lgsx_ReaderMoveNextPoints, [76](#)
 - Lgsx_ReaderQueryPropertyTypes, [77](#)
- Project Functions, [20](#)
 - Lgsx_ReaderGetMetadata, [20](#)
 - Lgsx_ReaderGetProjectDescription, [21](#)
 - Lgsx_ReaderGetProjectDescription2, [21](#)
 - Lgsx_ReaderGetProjectInfo, [21](#)
- QUA4D, [120](#)
- Reader Functions, [17](#)
- Release notes, [3](#)
- Release Notes 2024.0.0, [8](#)
- Release Notes 2024.0.1, [7](#)
- Release Notes 2025.0.0, [4](#)
- Release Notes 2025.0.1, [3](#)
- Run Functions, [48](#)

- Lgsx_ReaderEndRuns, [49](#)
- Lgsx_ReaderEnumRuns, [49](#)
- Lgsx_ReaderEnumSetupCamera, [49](#)
- Lgsx_ReaderEnumSetupCameraOfSetup, [50](#)
- Lgsx_ReaderGetLUTImage, [50](#)
- Lgsx_ReaderGetLUTImageOfSetup, [50](#)
- Lgsx_ReaderGetSetupCameraImage, [51](#)
- Lgsx_ReaderMoveNextRun, [51](#)
- Lgsx_ReaderMoveNextSetupCamera, [52](#)
- sceneRepld
 - CYMODELINFO, [109](#)
- SCyRgdBdyTxf, [121](#)
- Setup Functions, [41](#)
 - Lgsx_ReaderEndSetups, [41](#)
 - Lgsx_ReaderEnumSetups, [42](#)
 - Lgsx_ReaderEnumSetupsEx, [42](#)
 - Lgsx_ReaderGetCubelImage, [42](#)
 - Lgsx_ReaderGetImageLayerNames, [43](#)
 - Lgsx_ReaderGetImageLayerNames2, [44](#)
 - Lgsx_ReaderGetPanoImage, [44](#)
 - Lgsx_ReaderGetSetupGuid, [46](#)
 - Lgsx_ReaderMoveNextSetup, [46](#)
- setupId
 - CYGEOTAG, [108](#)
- setupIndex
 - CYTRAJECTORYVERTEX, [116](#)
- Sitemap Functions, [65](#)
 - Lgsx_ReaderEnumSetupsForSitemap, [66](#)
 - Lgsx_ReaderEnumSiteMaps, [66](#)
 - Lgsx_ReaderEnumSitemaps, [67](#)
 - Lgsx_ReaderGetSitemap, [67](#)
 - Lgsx_ReaderGetSitemapImage, [67](#)
 - Lgsx_ReaderMoveNextSetupForSitemap, [68](#)
 - Lgsx_ReaderMoveNextSitemap, [68](#)
 - Lgsx_ReaderMoveNextSiteMapImage, [69](#)
 - Lgsx_ReaderMoveNextSitemapImage, [69](#)
- SitemapImageType
 - LgsxClient.h, [130](#)
- SitemapImageType_Acceptance
 - LgsxClient.h, [131](#)
- SitemapImageType_Background
 - LgsxClient.h, [131](#)
- SitemapImageType_Overview
 - LgsxClient.h, [131](#)
- SitemapImageType_Unknown
 - LgsxClient.h, [131](#)
- startTime
 - CYTRAJECTORYINFO, [115](#)
- stopTime
 - CYTRAJECTORYINFO, [115](#)
- Target Functions, [47](#)
 - Lgsx_ReaderEnumTarget, [47](#)
 - Lgsx_ReaderEnumTargetOfSetup, [47](#)
 - Lgsx_ReaderMoveNextTarget, [47](#)
- time
 - CYSETUPINFO, [112](#)
 - CYTRAJECTORYVERTEX, [116](#)
- type
 - CYASSET, [106](#)
- url
 - CYASSET, [107](#)
- User Coordinate System Functions, [77](#)
 - Lgsx_ReaderEnumCoordSystems, [78](#)
 - Lgsx_ReaderMoveNextCoordSystems, [78](#)
 - Lgsx_ReaderMoveNextCoordSystems2, [78](#)
- vertexIndex
 - CYSETUPINFO, [112](#)