MapLink Pro

Installation and Upgrade

AUM1117-1 |24 June 2024| Status: Approved

North Heath Lane, Horsham, West Sussex, RH12 5UX, United Kingdom
Tel: +44 1403 273 173 Email: info@envitia.com
www.envitia.com

TABLE OF CONTENTS

1 INTRODUCTION

This document contains information about installing and upgrading from a previous MapLink Release.

This MapLink release is a major upgrade adding support for handling Unicode text natively. As such there are a number of breaking changes that need to be addresses when upgrading an application to this release. These changes have been kept as small as possible and are addressed in this document.

There are a number of supporting documents that you may also wish to consult;

- MapLink Pro Quick Start Guide
- MapLink Pro 10.0 for Windows Release Notes
- MapLink Pro Developers Guide
- MapLink Pro Studio User Guide
- MapLink Pro API Help
- MapLink Pro Studio Help
- MapLink Pro: Deployment of End User Applications

1.1 Supported Processors

The following processors are supported:

- Intel® Pentium® 4 processor family and higher.
- Non Intel® processors compatible with the above processor.

Early AMD x64 bit processors which lack support for `CMPXCHG16B` are not supported as this operation is required. These processors are the original "AMD Opteron Generation 1" (revision E and earlier).

1.2 Supported Windows Platforms

MapLink Pro 10.0 is supported on the following Windows platforms:

- Windows 10
- Windows Server 2012 R2, no prerequisites.
- Windows 8.1, the update KB2919355 plus dependencies must be installed prior to installing MapLink. This update is essentially Windows 8.1 Service Pack (see https://support.microsoft.com/en-gb/kb/2919355). If this update is not present the VCRedist installation will fail with various errors.
- Windows Server 2012, no prerequisites.
- Windows 8, no prerequisites.
- Windows 7, with Service Pack 1. KB2999226 will be installed if required.

The Visual Studio 2015 redist update 3 (version 14.0.24215) will be installed if needed.

MapLink Pro 11.1 only supports 64bit development.

1.3 Training, Consultancy and Sub-Contracting

Envitia provides a range of training options to help you get the best from MapLink Pro and MapLink Studio. These courses greatly help to accelerate your development, produce

optimised applications more quickly and to explore alternative ways of achieving your objectives.

Dedicated consultancy can also be provided either on-site or remotely, allowing our experienced developers to guide you towards the most appropriate approach to your application arena. Customers frequently find this useful when adding additional new functionality to their systems.

Envitia can also help accelerate your development by developing the MapLink component of your application for you or by undertaking a more extensive part of your project for you. Envitia has extensive experience of developing applications internally and for external customers.

If you wish to discuss these opportunities, please contact Sales by email sales@envitia.com or by phone: +44 1403 273173.

1.4 Glossary of Terms and Definitions

API	Application Programming Interface
DBIF	Database Interfaces
DMS	Digital Mapping System
DPI	Dots per Inch
DDO	Dynamic Data Object
DO	Display Object
EPSG	European Petroleum Survey Group. This organisation defines a standardised database of Coordinate Systems. These contain numeric codes associated with coordinate system definitions http://www.epsg.org/
GML	Geographic Markup Language http://www.opengeospatial.org/standards/gml
IDE	Integrated Development Environment
JPEG	JPEG raster format
MFC	Microsoft Foundation Class
MDI	Multiple Document Interface
STL	C++ Standard Template Library
SDI	Single Document Interface
SDK	Software Developers Kit
TMF	Envitia Map Format. Native geometry file format.
TIFF	TIFF raster format
TMC	The units that MapLink Pro uses to define a rectilinear coordinate space for drawing Map data and Overlay data with.

2 WINDOWS INSTALLATION

The following installation information relates exclusively to the release of MapLink Pro for Windows.

MapLink Pro libraries are built using Visual Studio 2015 update 3 (see release notes for precise version).

If you are using other development environments please refer to section 3.4.

2.1 Installation from CD/Internet

If you install MapLink Pro from a CD insert the CD in an appropriate drive and run one of the MSI installers. When you have installed MapLink please also install any patches.

There are two installers available:

- **MapLinkPro_10_0_B_x64.exe** 64-bit MapLink Pro development. The default for this installation will be the 64bit programs and 64bit Studio Automation.
  Includes MapLink Studio.
- **MapLinkStudio_10_0_B_x64.exe** 64-bit MapLink Pro Studio.

Where:

B is the Build Label Number.

Follow the on-screen instructions to install MapLink Pro.

Please run the Licence Administrator once the installation has completed. This ensures that the licence keys are setup for the new installation correctly.

2.2 Un-installing MapLink Pro

To un-install MapLink Pro, use the "Program And Features"/"Add/Remove Programs" facility accessible from your computer's Control Panel. There is no need to revoke your licenses using the Licence Administrator.

On Windows 10 the 'Apps & features' method of removal should be used.

2.3 Upgrading Windows or Hard-drive

If you are changing your version of Windows (E.G. upgrading from Windows XP to Vista/Windows 7/Windows 8/Windows 10), then you **must** request a De-Authorisation Key for MapLink Pro from Envitia before upgrading. On installing the newer version of windows, you will then need to request a fresh Licence Key. This procedure also applies when you are re-formatting or upgrading the hard-drive on which MapLink Pro is installed.

MapLink Pro will tolerate a certain amount of change to a PC before the license becomes invalid. However when upgrading Windows version or replacing the hard-drive MapLink Pro is installed upon, it is recommended that you uninstall and then re-install MapLink Pro.

If you do not have an active maintenance and support contract, a Licence Transfer Fee will apply whenever you need a new key generated. This will occur even if the licence is not being transferred to a different PC.

3 UPGRADING ON WINDOWS

3.1 Upgrading from a Previous MapLink Pro Version

If upgrading from MapLink Pro 4.0 SP3 or later, this installation may safely be installed alongside the old version if you wish. However you may wish to uninstall the old version before installing this one, to save system resources. There is no need to revoke your licences using the MapLink Licence Key Administrator; these licences will automatically be transferred to the new installation. If you upgrade your machine to Vista or a newer Windows Operating System then you should revoke your licenses and request new ones.

Versions of MapLink Pro prior to 4.0 SP3 placed DLLs in the Windows System32 directory. This release does not – consequently, it is necessary to remove versions of MapLink Pro prior to 4.0 SP3 from the machine before installing this release. Use the Control Panel Add/Remove Programs utility to remove previous installations. There is no need to revoke your licenses using the Licence Administrator; these licenses will automatically be transferred to the new installation.

3.2 Using Multiple MapLink Pro Versions

All installations of MapLink Pro since MapLink Pro 4.0 SP3 can happily co-exist on the same machine without interfering with one another. It is quite common for developers to retain installations of older versions of MapLink Pro to allow them to support their legacy applications built against them.

There is one important point that must be understood in order to use any older version. MapLink Pro libraries are located at runtime using the Windows '`PATH`' environment variable unless they are found in the current working directory. When MapLink Pro is installed on a machine, an environment variable named '`MAPL_PATH`' is added to the system and it contains the location of the bin directory of the installation. The '`MAPL_PATH`' environment variable is then referenced from the '`PATH`' environment variable.

Whichever version of MapLink Pro has been installed most recently will therefore take control of the '`MAPL_PATH`' environment variable. This means that when attempting to run an application that requires the MapLink Pro runtime libraries from an older version of MapLink Pro, it will load the latest version instead.

There are a couple of ways to avoid this problem:

- When wishing to use an older version of MapLink Pro, update the value of the '`MAPL_PATH`' environment variable to point to the appropriate `bin` directory.

  NOTE: Changing the environment variable will only affect applications that are started after the change has been made, not ones that were opened before. If the application is a system service, such as Microsoft's Internet Information Services (IIS), a reboot may be required.

- Ensure that the working directory of the application is set to the appropriate version of MapLink Pro's `bin` directory. This may be achieved by copying the executable to the bin directory, by setting the working directory in a short-cut, via an IDE or programmatically.

- It should be noted that this section refers specifically to MapLink Pro not a user deployed application. For details on how to deploy your application please see the "MapLink Pro Windows: Deployment of End User Applications" document.

3.3 For Developers using Visual Studio 2015

All MapLink binaries are built using Microsoft Visual Studio 2015. So long as applications that are built against MapLink Pro also use the same Visual Studio version, the use of debug configurations is supported. When using a different version of Visual Studio, please refer to section 3.44.

3.3.1 Visual Studio 2015 Project Wizards

The Wizards we normally ship for Visual Studio are not present in the initial release.

3.4 For Developers using other Visual Studio Versions

Some sample projects that ship with MapLink Pro are also provided in Visual Studio 2010 form. In order to build the projects in other releases of Visual Studio, users will have to load the Visual Studio 2010 project and allow it to be upgraded.

You will also need to link against the release mode libraries, even when you need to include debug information in your application. This is a concept called Pseudo Debug (see section 3.4.1.1).

The Wizards for Visual Studio 2010 will not generate correct solutions.

3.4.1 Issues with Mixing Visual Studio Versions

Microsoft always modifies the Standard Template Library (STL) and the debug memory manager between compiler releases. It is therefore important that care is taken when mixing binaries built using different versions of Visual Studio.

This section describes the best ways to avoid issues that might arise from this practice. Please note that this only applies when developing using C++, rather than the MapLink .NET interface.

3.4.1.1 Pseudo Debug

All MapLink samples and Visual Studio Wizards for older versions of Visual Studio use a debug configuration called 'Pseudo Debug'.

The reason we have the concept of ‘Pseudo Debug’ is that the Microsoft Visual Studio licence does not permit the redistribution of the debug libraries.

The 'Pseudo Debug' configuration gets around the licence restriction by using the release libraries which can be distributed.

To manually convert a solution the steps to follow are:

- Rename the debug build configuration in both the solution and project to "`Pseudo-Debug`". Obviously this is not strictly necessary, but it can be helpful.
- The runtime library setting should be changed to "`Multi-Threaded DLL`" from "`Multi-Threaded Debug DLL`"
- The MapLink libraries that it is linked against should be changed to the release versions (I.E. remove the "d" at the end)
- The "`_DEBUG`" pre-processor definition should be replaced with the "`NDEBUG`" one

Additionally the Qt samples build using the 'Pseudo Debug' concept to minimise the issue with either obtaining or building Qt with Visual Studio 2015 Update 3.

3.4.1.2 Memory De-allocation

The MapLink SDKs attempt to ensure that objects are allocated in the Visual Studio runtime that MapLink was built with. That is one of the reasons why most MapLink C++ classes override the `new` operator.

When a user derives from one of our classes the allocation occurs in the runtime library for the version of Visual Studio they are using. Therefore if that version differs from the one used by MapLink, that memory must be released by the caller rather than relying upon MapLink.

In general, when passing a derived class to MapLink, if there is an ownership flag then you cannot let MapLink take ownership when using a mixed development environment.

The following classes are known to present this issue if not used correctly.

- `TSLCustomDataLayer`

  If you are not using the version of Visual Studio that MapLink was built with then you need to manage the deletion of the `TSLClientCustomDataLayer` derived object yourself.

- `TSLUserGeometryEntity`

  If you are not using the version of Visual Studio that MapLink was built with then you need to manage the deletion of the `TSLClientUserGeometryEntity` derived object yourself.

- `TSLDynamicRendererCustom`

  If you are not using the version of Visual Studio that MapLink was built with then you need to manage the deletion of the `TSLDynamicRendererCustom` derived object yourself.

- `TSLDynamicRendererFactory`

  If you are not using the version of Visual Studio that MapLink was built with then you cannot use this class.

- `TSLInteractionModeManager`

  If you derive a class from `TSLInteractionMode` you need to call `removeMode` and delete the mode yourself.

- `TSL3DInteractionModeManager`

  If you derive a class from `TSL3DInteractionMode` you need to call `removeMode` and delete the mode yourself.

- `TSLDisplayObject`

  A new method `releaseResources` has been added. This should be overridden in a mixed build environment.

- `TSLDynamicDataObject`

  A new method `destroy` has been added. This should be overridden in a mixed build environment.

4 INSTALLATION ON LINUX, EMBEDDED LINUX, ANDROID PLATFORMS

Embedded Linux and Android have their own installation setup. Please refer to the specific documentation included with these platforms. The installation in all these cases involves copying the contents of the distribution to an accessible hard drive and configuring the compiler to find the include and shared libraries.

4.1 Installation from CD

The Core MapLink API libraries for UNIX/Linux/VxWorks are shipped uncompressed on a separate CD-ROM or tape, in the following directories:

`config`	Contains the configuration files.
`docs`	Documentation & Help.
`include`	Contains the header files for the MapLink C++ API's.
`lib64`	Contains the 64bit MapLink shared libraries.[1]
`Maps`	Sample Maps.
`redist64`	Contains 64bit deployable components.[2]
`samples`	Contains samples of how to use the MapLink C++ API's.
`OptionalComponents`	Contains the Optional Library Components encrypted.

A script named `mapl_init` (csh shell) and `mapl_init_bash` (bash shell) is included to initialise your environment, prior to building the samples. MapLink expects `LD_LIBRARY_PATH` and `MAPL_HOME` environment variables to be correctly setup when MapLink applications are run. The samples expect `MAPL_LIB_DIR` to be correctly set in order to build them.

To install the MapLink API libraries for UNIX/Linux/VxWorks, simply copy the contents of the CD into a suitable directory on your system (for example, /usr2/MapLink). Do **not** copy different platforms to the same installation directory as library names are not unique.

Although the UNIX/Linux/VxWorks runtime libraries are not locked, a valid licence is required to legally use them.

4.2 Installation of TerrainSDK and Optional APIs

The optional API's are now shipped encrypted on the Main CD in the directory `OptionalComponents`.

This directory contains a `README.txt` file which explains how to decrypt the files and install the contents on top of the Main Development SDK.

4.2.1 Additional Dependencies

The Database Interface SDK and Entity Store SDK require the Oracle Instant Client (version 10.2, 64bit).

[1] 64bit is not shipped for all platforms.

[2] 64bit is not shipped for all platforms.

MapLink Java requires Java 1.4 or newer.

The MapLink Web-Map Server SDK has been tested using Apache Tomcat 7.0.

5 UPGRADING ON LINUX, SOLARIS, EMBEDDED LINUX, VXWORKS PLATFORMS

5.1 Upgrading from a Previous MapLink Pro Version

This release may be safely installed alongside any previous version of MapLink Pro provided it is installed to a different directory.

The MapLink Pro libraries supplied do not provide multiple versioned interfaces. You will therefore need to recompile your application when you upgrade. This is also true if only archive files are supplied.

5.2 Using Multiple MapLink Pro Versions

All versions of MapLink Pro can happily co-exist on the same machine without interfering with one another. It is quite common for developers to retain installation of older versions of MapLink Pro to allow them to support their legacy applications built against them.

There is one important point that must be understood in order to use any older version. MapLink Pro libraries are located at runtime using the `LD_LIBRARY_PATH` environment variable. When using multiple versions of MapLink Pro on the same machine the path to the `lib` or `lib64` directory of the MapLink installation that you wish to use should be added to the `LD_LIBRARY_PATH`. Only one MapLink Pro installation should be referenced in this fashion, as otherwise the entry that appears first determines the MapLink Pro libraries that will be used.

It should be noted that this section refers specifically to MapLink Pro not a user deployed application. For details on how to deploy your application please see the "MapLink Pro: Deployment of End User Applications" document.

6 UPGRADING A C++/.NET APPLICATION

Upgrading from MapLink Pro 8.0 to MapLink Pro 10.0 should be straight forward. There are a few breaking changes which are detailed below.

6.1 Breaking Changes for a Future Release

Deprecated methods older than two releases will be removed from the API.

The method calls which take what are essentially index values will be converted to unsigned values.

The current licence keying approach is very likely to be modified. The probability is that the next version of MapLink will require a machine to be re-licenced.

6.2 Breaking Changes for MapLink Pro 10.0

Numerous changes have been made to the MapLink Pro APIs and tools. The following are highlights that may cause issues with an application when upgrading to MapLink 10.0.

6.2.1 Visual Studio Support

This release has been built with 'Visual Studio 2015 Update 3'.

This change means that how you build against MapLink Pro will change.

For versions of Visual Studio older or newer than the above version you will need to build against MapLink Pro using a concept called 'Pseudo Debug' rather than Debug (see section 3.4).

6.2.2 Supported Windows platforms

The following Windows platforms are no longer support for this and newer releases of MapLink Pro:

- Windows XP
- Windows Vista
- Windows 8.0

The equivalent Server versions of the above are also no longer supported.

6.2.3 Core SDK (MapLink Pro main API)

The following methods have changed:

- `bool TSLDrawingSurfaceBase::getBackgroundColour (TSLStyleID *value)`

The following method has been deprecated:

- `bool TSLClientCustomDataLayer::pick (const TSLEnvelope& extent, TSLPickResultSet* results)`

  Use the new method:

  ```
  bool TSLClientCustomDataLayer::pick (TSLRenderingInterface *renderingInterface, const TSLEnvelope& extent, TSLPickResultSet* results)
  ```

The deprecated method will be removed in the next release.

The following functional updates have been made that may require modifications to the application:

- The drawing surface ID passed to `TSLNDynamicDataObject::instantiateDO` was previously incorrect. This method will now be called with the correct ID, or user ID if set.
- The ID passed to `TSL(N)DynamicDataObject::instantiateDO` when `TSLInteractionModeMagnify` creates a display object will now be correct if a user ID has been set on the drawing surface.

6.2.3.1 GDAL/OGR

The MapLink Core SDK now links against GDAL/OGR. This was done as part of an internal simplification of the loading of GIS data at runtime and to allow us to expand support in the future for loading more data formats at runtime.

Please refer to the 'MapLink Deployment of End User Applications' for additional detail on what needs to be added to an application deployment.

6.2.4 S-52 SDK

The `TSLS52StateObject` has a new pure virtual method called `SHOW_TEXT_HALO()`. This method should just return true for the original behaviour.

6.2.5 S-63 SDK

Data stores created with older versions of MapLink should be re-created it at all possible if you are using the `TSLNTSurface` for drawing, the OpenGL Drawing surface is not affected.

The rendering of areas has been changed so that polygons with holes will not have the holes drawn correctly. The drawing of polygons with holes with GDI is very expensive.

The media ingest (`TSLS*MediaIngestManager`) workflow has been changed to convert any polygons with holes into what we call keyholed polygons when the S-57/S-63 data is loaded for the first time. These types of polygons can be drawn efficiently using the `TSLNTSurface` and the OpenGL drawing surface.

6.2.6 .NET Applications

The .Net assembly '`Envitia.MapLink.NativeHelpers.dll`' has been added in this release. The MapLink .Net CoreSDK depends on this assembly.

Any application which uses the MapLink .Net CoreSDK will need an additional project reference, or `#using` declaration for this dll.

Note: This assembly is built for '`anyCPU`'. A copy is provided in both `bin` and `bin64` directories for convenience.

6.2.7 .NET Framework Versions

The version 2 .NET framework assemblies have been removed, only framework 4 assemblies have been provided.

6.2.8 MapLink Studio

The JHS Transverse Mercator projection was not taking into account false northing correctly. This has been corrected.

Any map generated using the JHS Transverse Mercator should be re-created.

6.2.9 LandLink DIGM Converter

The export function that exports a `TSLEntitySet` into a DIGM blob has been renamed from export to `exportToDIGM`.

6.2.10 ASRP Exporter and CADRG Exporter SDKs

The method's named export have been changed to include the product being exported as the keyword 'export' is now reserved by the C++ language.

6.3 Breaking Changes for MapLink Pro 8.0

The class names mentioned below for C++ are very similar to the class names in .NET.

6.3.1 Compiler Setup

When compiling applications that use MapLink Pro using Visual Studio, the compiler must be set to treat `wchar_t` as a native built-in type. This setting can be found under the `Configuration Properties->C/C++->Language` page of the project properties in Visual Studio.

6.3.2 Unicode

MapLink Pro now supports Unicode using the UTF-8 character encoding. Please review section 6.3.13 below.

6.3.3 Old File Formats

Windows and other Operating Systems only support a single Code Page at a time. Older versions of MapLink on Windows displayed multi-byte text data using the system code page on Windows. This approach will not work when the need is to display multiple languages.

This has necessitated a change to the way text is processed so that we store text as UTF-8 internally and expect an application to pass UTF-8 text to MapLink and to accept UTF-8 from MapLink.

For additional information please see section 6.3.13 below.

6.3.3.1 TSLVariant

When writing or reading a `TSLVariant` a `TSLVariantTypeChar` may be promoted to a `TSLVariantTypeString`. Both these types of variant are interchangeable as the definition depends on how long the text is in bytes.

6.3.3.2 Maps

Maps created for older versions of MapLink Pro should load and display without needing to be modified. If you have problems please refer to section 6.3.13.

If you have problems with older maps we do need to know so that we can assess the issues people are encountering to see if we need to modify how we convert text on reading.

6.3.4 Transverse Mercator Projection

EPSG have changed the formula used for Transverse Mercator while retaining the EPSG IDs for the affected Coordinate Systems that use Transverse Mercator.

The original formula "USGS Snyder" and "JHS" formulas produce similar results in a +-4 degree band around the central longitude. Outside this band the results diverge. The JHS algorithm is more accurate out to +-40 degrees.

EPSG recommend that the two formulas are not mixed.

EPSG recommend the use of the JHS formula.

The Snyder formula was used in MapLink 7.5 and older versions.

Both formulas are available in MapLink 8.0 and newer.

6.3.4.1 TSLCoordinateSystem & tsltransforms.dat

EPSG have changed the formula used for Transverse Mercator while retaining the EPSG IDs for the affected Coordinate Systems that use Transverse Mercator.

The original formula "USGS Snyder" and "JHS" formulas produce similar results in a +-4 degree band around the central longitude. Outside this band the results diverge. The JHS algorithm is more accurate out to +-40 degrees.

EPSG recommend that the two formulas are not mixed.

EPSG recommend the use of the JHS formula.

The Snyder formula was used in MapLink 7.5 and older versions. Both formulas are available in MapLink 8.0 and newer.

To address the EPSG change to the Transverse Mercator a number of changes have been made to `tsltransforms.dat` that may affect an application. The changes are outlined below:

- MapLink coordinate system IDs in the range [-5000..-9000] use the Transverse Mercator JHS projection algorithm.
- IDs in the range [-1..-4900] use the original USGS Snyder Transverse Mercator projection algorithm.
- The EPSG ID in the case of both Coordinate Systems are the same.
- The NAME has been updated to contain '`(Snyder)`' or '`(JHS)`' to distinguish the algorithm used.

Where:

ID is the value used in `TSLCoordinateSystem::findByID`() and returned by `id`().

NAME is the value used by `TSLCoordinateSystem::findByName`() and returned by `name`().

The method `TSLCoordinateSystem::findByName()` functionality has changed slightly.

When looking up a Coordinate System that uses Transverse Mercator projection the method expects one of two forms to be used, for example:

- "`UTM (WGS84) Zone 1 North (Snyder)`"
- "`UTM (WGS84) Zone 1 North (JHS)`"

For backwards compatibility the `findByName`() method will return the original Snyder Coordinate System if "`(Snyder)`" or "`(JHS)`" is missing from the name being searched for. In this case a warning will be placed on the `TSLThreadedErrorStack`.

The method `TSLCoordianteSystem::findByEPSG()` may not work as expected, for example;

```
const TSLCoordinateSystem *coordSystem =
                        TSLCoordianteSystem::findByEPSG(27700);
```

Returns the OSGB coordinate system using the Snyder Transverse Mercator formula. For the new formula you need to do the following:

```
const TSLCoordinateSystem *coordSystems[2];

int numberFound = TSLCoordianteSystem::findByEPSG(27700, coordSystems, 2);
```

You would need to check the `numberFound` variable and then validate the name of each returned `TSLCoordinateSystem` to see if it was the Snyder or JHS version. You could use the MapLink IDs as these are unique.

6.3.4.2 TSLCoordinateConvertor

`TSLCoordinateConvertor` calls will need to be updated to specify which Transverse Mercator formula to use.

The deprecated methods use the 'USGS Snyder' formula.

6.3.4.3 TSLMGRSGridDataLayer

The `TSLMGRSGridDataLayer` uses the `TSLCoordianteConvertor` to calculate both MGRS and UTM information.

The default Transverse Mercator formula used is the 'USGS Snyder' formula if the Coordinate System provided by the user or via the Drawing Surface does not use a Transverse Mercator projection. You may change the default formula if required.

6.3.4.4 TSLWMSDataLayer and TSMWMTSDataLayer

The layers default to using the Transverse Mercator Snyder formula in preference to using the JHS formula. This is because the majority of WMS/WMTS servers appear to not be using the JHS formula.

You can change the behaviour by calling the `setTransverseMercatorJHSFormula` method.

6.3.4.5 GML and WFS Client SDK

The TSLGMLInstanceDataLoader has a get/set Transverse Mercator formula.

In addition the WFS Client SDK has similar methods to get/set the Transverse Mercator formula.

6.3.5 TSLWMSDataLayer

The `getCRSAt`, `getDimensionAt` and `getStyleAt` methods of `TSLWMSServiceLayer` now return all CRSs, dimensions and styles that apply to that layer instead of only the ones defined on that layer in the service metadata.

The `getDimensionAt` method of `TSLWMSServiceLayer` has changed to return a new class containing information about the dimension. This change is not compilation compatible with code that was previously using the method with the 2nd and 3rd arguments as their default values.

6.3.6 TSLWMTSDataLayer

The `onTileMatrixSetNotSelected` callback on `TSLWMTSServiceSettingsCallbacks` no longer exists. Applications should instead implement the new `onChoiceOfServiceCRSs` abstract method. This new callback means it is no longer required for applications to manually identify and set common tile matrix sets on visible layers through the `TSLWMTSServiceLayer::selectTileMatrixSet` method.

6.3.7 Font Symbols

The symbol font character type has been modified to be a 32bit unsigned integer. This now represents a single UTF-32 code point rather than a single 8bit character. The values minus one (-1) indicates that the value should be ignored. The value minus two (-2) represents the concept of multiple values. These two values cannot be used. Reading of a font character created in an older version of MapLink is supported.

Writing of a font character to an older MapLink version file format will not work in all situations. Round-tripping is supported; i.e. to load an old TMF file and write out in an old format. Writing out a to an old format where a value used is outside the code page originally used or where the value is greater than 255 is unlikely to work as the value cannot be represented as a single multi-byte character.

The following methods are affected:

- `TSLRenderingInterface::setupSymbolAttributes`
- `TSLNTSurface::symbolStyleValue`
- `TSLMotifSurface::symbolStyleValue`

The following rendering attributes classes/methods/enums are affected:

- `Class: TSLRenderingAttributes::m_symbolFontCharacter`
- `Enum: TSLRenderingAttributeInt::TSLRenderingAttributeSymbolFontCharacter`
- `bool TSLEntityBase::setRendering (TSLRenderingAttributeInt attribute, int value)`
- `bool TSLEntityBase::getRendering (TSLRenderingAttributeInt attribute, int* result) const`

6.3.8 Definition of True & False

`True` and `False` were defined in `tslatomic.h`. These were not used and have been removed.

6.3.9 Editor SDK APP6A

Previously, the Editor SDK APP6A classes had a `TSLAPP6AHelper` object setup internally that defaulted to use the APP6A Symbol configuration file. The choice of configuration file to use could not be modified by the User. This meant that a situation could occur that a `TSLAPP6AHelper` class specified by the user could be told to use a different configuration file, e.g. a 2525B symbols file, whilst the helper class used internally by the Editor SDK would always use the default APP6A symbols file.

To avoid this, when the user derives from the `TSLAPP6ARequest` class, the user must set the `m_helper` member of their class to a `TSLAPP6AHelper` class object setup with the desired symbols configuration file.

6.3.10 .NET Applications

The change to using UTF-8 as the character encoding in the MapLink C++ code are principally hidden by the .NET API.

It is still advisable to read about the Unicode changes as many of the inherent limitations that previous MapLink releases had will no longer apply.

This section describes the specific breaking changes for .NET that we are aware of.

6.3.10.1 Font Symbols

The Symbol font character type has been modified to be a 32bit unsigned integer. This now represents a single UTF-32 code point. The values zero and minus one are ignored.

The following methods are affected:

- `TSLNRenderingInterface::setupSymbolAttributes`
- `TSLNDrawingSurface::symbolStyleValue`

The following rendering attributes classes/methods/enums are affected:

- `TSLNRenderingAttributes::symbolFontCharacter property`
- `Enum: TSLNRederingAttribureInt::TSLNRenderingAttributeSymbolFontCharacter`
- `bool TSLNEntityBase::setRendering (TSLNRenderingAttributeInt attribute, System::UInt32 value)`
- `bool TSLNEntityBase::getRendering (TSLNRenderingAttributeInt attribute, OUT System::UInt32 % result) const`

6.3.11 MapLink Studio

The following filters had a simplistic UTF-8 to Latin-1 conversion option in MapLink Studio:

- Shapefile
- MIF
- OS Mastermap
- OS Vector Map Local

This option will be ignored when generating new maps. The option may not work in exactly the same way when generating older maps from MapLink Studio.

We would therefore recommend writing out to the latest MapLink version and upgrading any application to take advantage of the correct display of UTF-8 strings.

Old maps should continue to be displayed correctly.

6.3.12 MapLink Studio Automation

The Maplink Studio Automation interface specifies strings as LPCTSTR. Programs that use this interface should require no updates due to studio now being a Unicode application.

6.3.13 Unicode

The MapLink Pro backward compatibility has been broken with the introduction of Unicode support in the way 8-bit characters are handled.

6.3.13.1 Text/String Handling

All text data within MapLink is now UTF-8 encoded. MapLink expects all string passed via the API functions to be encoded in UTF-8 and will return all text to user applications in UTF-8.

Prior to support of Unicode users may have relied upon 8-bit character strings being passed through the MapLink Pro API without change. Now this is only true for 7-bit ASCII[3] and 8-bit UTF-8 encoded strings.

You will not be affected by these changes if:

- You only used 7-bit ASCII strings.
- You used the .NET SDKs.
- You use a non-Windows platform (these are usually UTF-8 by default).

In order to support Unicode with MapLink Pro the following behavioural changes were made at the API:

- Filename and path names have to be long filename/paths on Windows
- Text has to be UTF-8. Passing text in the System Code Page to MapLink is no longer supported.
- All text passed to MapLink is assumed to be UTF-8.

To minimise the impact of this change MapLink will read and convert all text from files generated by MapLink versions prior to 8.0 on load to UTF-8 where possible. The default conversion assumes the files contain text in the System Code Page on Windows, and CP-1252 on other platforms.

6.3.13.2 C++ SDKs

The MapLink API uses '`char *`' pointers to pass string information to and from an application.

Previous versions of MapLink assumed that the data passed were ASCII though there was no enforcement or checking performed. Internally the Microsoft CRT would have checked that data was valid ASCII when functions such as `isalnum`, `isalpha`, `islower` etc… were called generating an `assert` when running the debug version of MapLink.

This version of MapLink still uses '`char *`' pointers to pass string information to and from an application. The principal difference is that the data passed has to be UTF-8 which will accept 7-bit ASCII as a valid sub-set.

In the majority of cases an application may not need to adjust what is passed into MapLink unless the application is being upgraded to Unicode or the user did not use 7-bit ASCII.

MapLink provides two utility classes, `TSLUTF8Encoder` and `TSLUTF8Decoder`, to convert text strings between the System Code Page, wide characters, and UTF-8 to aid in updating applications that were previously passing text in the System Code Page to MapLink.

6.3.13.3 .NET SDKs

The .NET SDKs use the Windows concept of Unicode at the MapLink API.

The class `TSLNCoordinateConverter` takes System::Char for some of the conversion methods. The data passed in and out in these cases is assumed to be 7-bit ASCII.

6.3.13.4 Text

The following Geometry Text primitives are currently supported by MapLink:

- `TSLText / TSLNText`
- `TSLGeodeticText / TSLNGeodeticText`
- `TSL3DText / TSLN3DText`

`TSL3DText`/`TSLN3DText`[4] only supports a subset of 7-bit ASCII.

`TSLText` and `TSLGeodeticText` will display text in multiple languages. The text may contain more than one language and the languages displayed may be left to right and right to left.

6.3.13.5 Fonts

Not all fonts contain all the necessary glyph entries to display Unicode strings correctly. If some of the string displays correctly but the rest does not please try another font.

The font you use is key to the display of text. If the font does not support the language/script then you need to find an alternative font that does. You can add new fonts to tslfonts.dat file.

6.3.13.6 Vertical Text Alignment

We do not currently support vertically aligned text. We will draw the text but it will be drawn horizontally.

6.3.13.7 Right to Left Scripts

We support right to left scripts. The alignment of the string is not swapped as it can be in some text editors.

You can mix different scripts within a single text item.

6.3.13.8 Vector Font

The vector font support is limited to 7-bit ASCII on both the GDI and X11 Drawing Surfaces. Vector font drawing is not supported by the 2D OpenGL Drawing Surfaces.

All MapLink drawing surfaces support drawing rotated system text. This negates the need for the Vector font as this was primarily used for drawing rotated text on platforms that could not support drawing of rotated system fonts.

6.3.13.9 Windows

In previous versions of MapLink Text was drawn using the Code page of the application.

[4] Please contact support if this is an issue so that we can gauge the importance of supporting Unicode in the 3D Text primitive.

This approach limited the support language to a single language and did not correctly support complex scripted languages such as Arabic. This support was not documented and used as a work around by a few customers that required non-ASCII support.

MapLink now expects text to be UTF-8, though we have provided wchar_t helper methods, this allows multiple languages to be drawn and supported.

On reading old data we will attempt to convert the text data to UTF-8.

6.3.13.9.1 Non-Windows

On non-windows platforms MapLink mapped the character codes directly to the font being used. This was to all intents and purposes ASCII.

6.3.13.9.2 Filenames and Paths

- MapLink expects the filenames and paths on Windows to be long filenames/paths. Passing a short filename/path may not work correctly.
- Paths relative to a drive-specific working directory are unsupported, e.g d:file.txt (file.txt relative to the current working directory on drive D:)
- MapLink will attempt to convert a path without a drive letter using the current working directory, paths such as /secure/source/foo/bar.baz will work fine, a path such as foo/bar.baz will be used as a regular relative path
- The maximum supported filename and path length is 4096 characters.

6.3.13.10 Filters

Many of the filters in MapLink processed text data in the assumption that the data was ASCII. In the case of the following filters this was not always true:

- VPF
- S-57
- Shapefile[5]
- MIF[6]
- OS Mastermap[7]
- OS Vector Map Local[8]

The MapLink filters have been updated to identify the text encoding and to convert all text to UTF-8.

The ASCII filter has been enhanced to support UTF-8.

The GDAL/OGR filter is dependent upon the GDAL/OGR library correctly converting text to UTF-8.

[5] The panel in Studio had an option to convert 'UTF-8 to Latin-1' workaround.

[6] The panel in Studio had an option to convert 'UTF-8 to Latin-1' workaround.

[7] The panel in Studio had an option to convert 'UTF-8 to Latin-1' workaround.

[8] The panel in Studio had an option to convert 'UTF-8 to Latin-1' workaround.

6.4 Upgrading application from MapLink Pro 7.1 or Older

If you are using a version of MapLink Pro older than version 7.1 please refer to the "Release Notes" for the versions prior to MapLink 7.1 for details of changes and the MapLink 7.5 "Installation and Upgrade Notes".

If you do not have access to the above documents for the older versions please contact support@envitia.com for copies.

7 DLL AND LIBRARY NAMING AND THEIR LOCATIONS

7.1 Windows

7.1.1 DLL Naming Convention

A number of methods in the MapLink Pro API expect the user to pass a DLL name to them.

Because of the port in a previous version of MapLink to 64-bit the DLL names had to be changed to allow 32-bit and 64-bit MapLink based applications to coexist without complicating user code.

As such the following convention has been adopted (where '**D**' indicates debug, and '**64**' indicates 64-bit):

Build Configuration	DLL Name	Example	Filename Required
64-bit Release	DLLName**64**.DLL	RasterFilter**64**.DLL	RasterFilter
64-bit Debug	DLLName**64D**.DLL	RasterFilter**64D**.DLL	RasterFilter

This simplifies coding as the same DLL name (usually filter) is passed to MapLink Pro. We will then add the correct ending to the name based on the build configuration. As of MapLink 11.1 only 64-bit libraries are available to use however the naming of these has not been changed to enable compatibility with existing applications.

7.1.2 Library Naming Convention

The library names for MapLink Pro follow the same convention as for naming DLLs, so for example:

Build Configuration	DLL Name	Library Name
64-bit Release	MapLink **64**.DLL	MapLink **64**.lib
64-bit Debug	MapLink **64D**.DLL	MapLink **64D**.lib

8 .NET SDKs

Unlike the C++ API, the .NET API has remained largely unchanged in this release.

8.1 64-Bit Assemblies

Whilst the convention in assembly naming in .NET is the use the same name for both 32-bit and 64-bit assemblies, due to the dependency on the C++ MapLink libraries this is not possible. Instead the .NET assemblies have used the same suffixes used by the C++ binaries. Aside from this change there should be no other changes required to migrate to using 64-bit.

8.2 .NET Framework Versions

The version 2 .NET framework assemblies have been removed, only framework 4 assemblies have been provided.

9 64-BIT ISSUES

9.1 Supported Processors

The supported processors are listed in section 1.1.

Additionally early AMD x64 bit processors which lack support for `CMPXCHG16B` are not supported as this operation is required.

These processors are the original "AMD Opteron Generation 1" (revision E and earlier).

9.2 API Types

We have introduced new MapLink API types to help with porting and developing new Applications.

We advise that you use these new types. Please refer to the 'API Types' section of the MapLink Developer's Guide. On non-Windows platforms you are strongly advised to use these types.

9.3 Windows Stack Sizing

The stack model for Windows 64-bit is quite different from 32-bit. By default Windows only specifies a 2MB stack. We recommend that you use at least 4MB. This is because the stack is split into two one for data and one for return addresses.