ENVITIA

DISCOVER > ANALYSE > ACT

MapLink S63 and S52 SDK
Developer's Guide

AUM1111|14 June 2023| 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 is intended to give developers a guide to designing and implementing:

- An S-63 OEM Data Client application using the functionality that the MapLink S-63 SDK provides;
- Use of the S-52 SDK for visualising both S-63 and S-57 data.

It is assumed that the reader is both familiar with the International Hydrographic Office's (IHO) S-63 and S-52 Standards and the use of the ENVITIA MapLink Core SDK.

A Reference Workflow C++ sample application is supplied to demonstrate the use of the S-63 SDK and S-52 SDK.

The Reference Workflow sample demonstrates:

- S-63 Workflow.
- S-57 Workflow.
- S-52 rendering.
- S-63 OEM tests.

1.1. What is S-63?

S-63 is an IHO standard for encryption of S-57 data for use in ECDIS displays. It defines the lifecycle of the data and the encryption and decryption of the data. It is intended to be used by manufacturers of ECDIS displays and providers of S-63 data. The encryption is designed to: allow data users to check that the data is unmodified and has come from a bone-fide supplier; prevent use of the data on multiple machines or by unauthorised users; and mandates checks at data ingest and display time to prevent data being used outside of the permitted licence terms. In order to be able to decrypt publically provided data, the OEM must show compliance with the workflow using the test dataset, and is then issue with an OEM specific 'M_KEY' by the IHO.

Essentially, when an order is placed to purchase some S-63 encrypted data, the purchaser must provide a 'User Permit' which is used to encrypt the parts of the data. This permit includes both the 'M_KEY' of the OEM and the unique 'HW_ID' of the machine that will display the data. The supplied data also has portions that are encrypted using the Data Servers unique key, which allows the user of the data to verify that it has been uncorrupted and is genuinely from that provider. Other parts of the supplied data is a 'Cell permit' which is encrypted using information from the 'User Permit', Thus part of the S-63 data is generic and part is specific to the device that is displaying the data.

You cannot buy S-63 data without an IHO issued M_KEY and you can't get the M_KEY without showing compliance and you can't show compliance without following the workflow and having unique, secure HW_ID on every machine displaying the data.

1.1.1. How does the MapLink S-63 SDK help with compliance?

The MapLink S-63 SDK provides the vast majority of the functionality required for S-63 compliance and essentially implements all generic functionality leaving only the OEM specific portions to be implemented – such as secure storage of the unique HW_ID of the display and supply of accurate time information. These are usually via callbacks during the workflow processing.

The SDK is split into:

- Media Ingest which handles cell permit authentication, the data update workflow and conversion to encrypted TMF.
- Data Layer which uses the cell permits to validate permission to display and encrypted TMF to display the data – this can be used with the S-52 Dynamic Renderer. The display has various mandatory rules about warnings and errors which depend upon the display device time compared to the cell permit licence start and end dates.

1.2. What is S-52?

S-52 is an IHO standard for displaying, managing and implementing an ECDIS system.

The MapLink Pro S52 SDK implements the display rules for S-57 ENC data (excluding the Mariners Objects).

The MapLink Pro S63 SDK can be used for implementing the management of S-57 and S-63 data.

The OEM is responsible for implementing the Mariners Object display and actual ECDIS system and showing compliance.

1.3. Scope

When this document refers to classes and methods from the SDK it will use the C++ SDK class names. The.NET interfaces to this library, if provided, have class and method names that are very similar and provide equivalent functionality.

The usage of certain functions in the SDK is detailed in this document but it is intended that the reader refers to the MapLink S-63/S-52 API documentation for details of functionality not covered.

1.4. References & Related Documents

Note: References to the following documents are enclosed in square brackets in the text [X].

1. "IHO Data Protection Scheme", Edition 1.1 (March 2008) – Special Publication No. 63 (S-63)
2. "TEST DATA IMPLEMENTATION GUIDE, (Part of Appendix 1 of S-63 Edition)" Edition 1.1 (July 2004)
3. "S57 Edition 3.1: IHO Transfer Standard for Digital Hydrographic Data", Edition 3.1 (November 2000)
4. "S-57 Appendix B: ENC Product Specification", Edition 3.1 (November 2000)
5. "MapLink Pro for Windows Release Notes", URN1101-xx
6. IHO ECDIS Presentation Library, Edition 3.4, January 2008, Special Publication No. 52.
7. IHO Test Data Sets for ECDIS, Edition 1.1, December 2008, Special Publication No. 64.
8. "IHO Data Protection Scheme", Edition 1.2 – Special Publication No. 63 (S-63)
9. "TEST DATA IMPLEMENTATION GUIDE, (Part of Appendix 1 of S-63 Edition)" Edition 1.2

1.5. Glossary of Terms and Definitions

IHO	International Hydrographic Office
SA	S-63 Scheme Administrator
DS	S-63 Data Server
OEM	S-63 Original Equipment Manufacturer
Data Client	The S-63 application produced by an OEM
HW_ID	The unique, 5 character hexadecimal, value assigned by the OEM to each deployment of their Data Client
ENC	Electronic Chart Display as defined by the ENC Product Specification [4].
Cell Key	Symmetric Encryption key used when ENC data is encrypted and therefore decrypted
Cell Permit	A Cell Key that has been encrypted using a specific Data Client user's HW_ID and should therefore only be usable by them.
SSE XX	An S-63 defined error code that should be shown to the user when raised.
TMF	ENVITIA Propriety Geometry File Format
Filter	A filter is a component specifically designed to translate from a file format, such as Shapefile, ARCS, CADRG, S-57 etc..., to a format the MapLink Pro can read and display quickly.

2. MapLink S-63 SDK Structure

This first part of this document discusses the difference between the MapLink S-63 SDK and a 'Data Client' and provides an overview of the structure of the S-63 SDK. The term 'Data Client' used in this section refers to an OEM constructed client application.

This part of the document primarily discusses the loading, management and display of S-63 data. The loading, management and display of S-57 data is covered in passing as the concepts are similar though a lot less complex. The provided sample demonstrates both.

2.1. The Distinction between a Data Client and the MapLink S-63 SDK

MapLink does not provide a Data Client as the IHO's S-63 standard defines. Instead the MapLink S-63 SDK provides the building blocks that can be used to create a Data Client. By supplying the S-63 functionality in this form, developers can create their own branded application that could provide their users with additional abilities outside of being able to load S-63 data.

In keeping with the MapLink philosophy of not dictating the architecture of user applications the MapLink S-63 SDK does not directly show the error messages and warnings that the OEM is mandated to display. In the cases where the SDK can detect the circumstances that should raise these error messages the SDK will notify the calling application via a call-back. Not all of the S-63 mandated errors can be detected by MapLink, therefore there is a gap between what the S-63 SDK provides and fulfilling the OEM's IHO imposed requirements. Equally there are certain client system setup requirements, such as the secure storage of the IHO's X.509 certificate, which the SDK cannot guarantee and is thus left to the OEM to implement.

This document will attempt to describe each of these cases and therefore allow the developer to bridge the gap between the MapLink S-63 SDK and the Data Client S-63 defined requirements.

2.2. The MapLink S-63 SDK Building Blocks

The S-63 Standard defines two different processes that the OEM should be able to demonstrate can be completed using their internal private tools and their Data Client:

- The ability to generate 'User Permits'.
  These are the specially formatted requests that each Data Client user will require when making requests to their Data Supplier (DS) or suppliers for a set of ENC Cells that they wish to purchase. The ability to generate them should not be included in the Data Client but should instead be done by the OEM for each of their Data Client's users.
- The ability to ingest a set of 'Cell Permits' and 'Exchange Sets'.
  This is the actual process of consuming and displaying the data that the DS provides to the Data Client application user.

The MapLink S-63 SDK divides the second process into two separate blocks of functionality; the ability to consume the DS provided data is provided via the `TSLS63MediaIngestManager` while the ability to display the data is provided via the `TSLS63DataLayer`. This division was created so that Data Clients have the ability to perform the data ingest separately from the display, which is especially useful when ingesting large quantities of data.

The `TSLS63MediaIngestManager` will return to the Data Client, via a callback interface, the ingested S-63 data in the form of two blocks of data for each ENC cell both of which are also encrypted. The first of these data blocks will be used when ingesting updates to that ENC cell while the second is used for both displaying the data and when updating the ingested data. The Data Client application shall therefore need to store both blocks for future use.

The `TSLS63DataLayer` will request from the Data Client, via a callback interface, each of the ENC cell encrypted data blocks, provided by the `TSLS63MediaIngestManager` for each of the Cell

Permits it has been provided. It will then decrypt the blocks and display them through the drawing surface or surfaces into which the layer has been added.

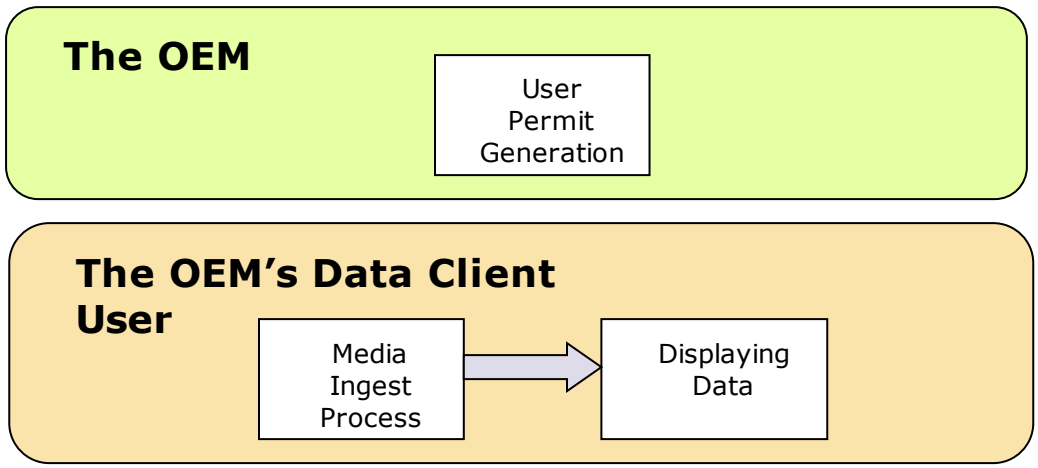


Figure 1 - This diagram shows who uses each of the three MapLink S-63 SDK Building Blocks.

Later sections in this document will discuss in more detail the steps that are required to fully utilise these building blocks.

2.3. Library Usage and Configuration

This section describes how to link in the MapLink S-63 SDK in the forms MapLink provides: a C++ SDK and a .NET SDK.

The S-63 SDK is run-time locked so that only developers who have purchased this SDK may use it. It should be noted that the unlock code required to access this SDK is not the licence key provided when purchasing other SDKs and input via the Licence Key Administrator.

There are two unlock codes;

- `TSLKeyedS57MediaIngest`: Provides access to ingesting and display of unencrypted S-57 data only.
- `TSLKeyedS63`: Provides access to the full S-63 SDK functionality for encrypted and unencrypted S-57

This section will describe how to perform this unlocking through each of the SDKs.

2.3.1. C++

The C++ version of the MapLink S-63 SDK comes in 2 different flavours (debug or release).

It should be noted that the library to be linked with should be determined by the Core SDK library that you are using within your application. For example, if you are using the DLL Release mode version of the Core SDK (`MapLink.lib/MapLink64.dll`) then you must also use the equivalent S-63 SDK library (`MapLinkS63.lib/MapLinkS63.lib`).

MapLinkS63.lib or MapLinkS6364.lib Release mode, DLL version. Uses Multithreaded DLL C++ run-time library. Your application must also link the MapLink CoreSDK library `MapLink.lib/MapLink64.lib`. Requires the `S57Filter.DLL/ S57Filterd.DLL` at runtime	MapLinkS63d.lib or MapLinkS6364d.lib Debug mode, DLL version. Uses Debug Multithreaded DLL C++ run-time library. Your application must also link the MapLink CoreSDK library `MapLinkd.lib`/MapLink64d.lib. Requires the `S57Filterd.DLL/ S57Filter64d.DLL` at runtime. Should not be redistributed to deployments.

To unlock the S-63 C++ SDK, the OEM should call `unlockSupport` on the `TSLUtilityFunctions` utility class, which is exposed by the C++ Core SDK. The first parameter to this static function should be the `TSLKeyedS63` value from the `TSLKeyedOption` enumeration while the second should be the unlock code. This will unlock loading of S-63 media and S-57 media.

The S-63 SDK can be unlocked just for S-57 media using the `TSLKeyedS57MediaIngest` unlock option.

2.3.2. .NET

The .NET interface to the MapLink S-63 SDK is exposed by the `Envitia.MapLink.S63.DLL` assembly under the namespace `Envitia.MapLink.S63`. This library is dependent on both the

Release mode C++ MapLink S-63 SDK and the Core .NET SDK assembly `Envitia.MapLink.DLL`. All dependencies for both these libraries will need to be redistributed with the Data Client[1].

The standard .NET assemblies are built using the .NET Framework 4 although additional versions are supplied built using Framework 4. Additionally, a 64-bit version of the assemblies are included. Refer to the MapLink Developer's Guide for further information.

To unlock the S-63 .NET SDK, the OEM should call `unlockSupport` on the `TSLNUtilityFunctions` utility class, which is exposed by the Core .NET SDK. The first parameter to this static function should be the `TSLNKeyedS63` value from the `TSLNKeyedOption` enumeration while the second should be the unlock code.

2.3.3. Limitations

The S63 SDK cannot be used with an Accelerated Surface or a 3D Surface. Please contact support or your account manager to discuss.

[1] Refer to section 5.2 of the MapLink Pro for Windows Release Notes [5] for details of the Core .NET SDK dependencies.

2.4. Data Client Prerequisites

There are some S-63 mandated prerequisites to the Media Ingest and Display processes that MapLink cannot implement on behalf of the Data Client. The OEM must implement these in order to fulfil the requirements of the IHO's OEM testing procedure. These are discussed in the following subsections.

2.4.1. Scheme Administrator's Certificate

The SA's X.509 certificate must be held securely on the Data Client user's computer ([2] *"Section 5.4.3: Test 3.3 – Data Client Authenticate SA Public Key on CD Against Installed SA Public Key"*). This is not clearly stated in the S-63 Standard section dealing with the installation of the SA's certificate ([1] *"Section 10.6.1: Authenticate/Verify SA Digital Certificate"*) but is stated in the S-63 Appendix 1. The Data Client must also provide the ability for the user to update the securely held SA's certificate in the case where the currently held one becomes expired or has been replaced by the SA.

2.4.2. Assignment and Storage of Data Client User's HW_ID

The OEM must also supply each Data Client user with a unique 5 digit hexadecimal number ([1] *"Section 4.2.2: HW_ID Format"*). It is suggested in the S-63 standard that perhaps this HW_ID should be provided to the Data Client user via a dongle. In either case the HW_ID must be stored within the system in a secure way, both to prevent the HW_ID being used on a different machine and to prevent the one installed being changed. MapLink will need to be provided with this HW_ID for almost every operation that it performs so the Data Client will need ready access to it.

2.4.3. Display of S-63 Errors and Warnings

S-63 defines 27 different error codes ([1] *"Section11: S-63 Error Codes and Explanations"*) of which 22 apply to the Data Client application. These errors must be displayed to the user in some way, perhaps through an error dialog, an error log or incorporated into the data display. MapLink can detect all but 1 of these errors; `SSE 14` which should be raised when the system date does not match a reliable alternative source, such as a GPS device.

Additionally S-63 defines certain warning messages that must also be presented to the user. These warnings should be presented in the same way as the errors listed previously, but as these warnings don't have S-63 defined error codes, the format of the message may need to be different. An example of these warnings is the case where a cell has been cancelled but is retained in the Data Client's data store ([1] *"Section 6.2.3.2: Managing Cancelled Cells"*).

It is therefore the OEMs responsibility to ensure that their application displays any errors and warnings that the MapLink S-63 SDK detects and implement the raising of the errors that MapLink cannot detect.

2.5. User Permit Generation

A User Permit must be provided to a DS whenever a user of the Data Client requests access to a set of ENC Cells; in return the DS will supply a set of Cell Permits. The User Permit provides a secure method of transmitting the user's unique HW_ID, which should have been assigned to them by the OEM, but also ensures the user is utilising a Scheme Administrator (SA) approved Data Client.

Each Data Client user will therefore request from, or be provided by, their OEM with a User Permit. This User Permit will, in normal circumstances, remain the same for each request that the user makes to their Data Supplier or Suppliers.

To generate a User Permit, the OEM will require the user's HW_ID, that it has previously assigned, and their own unique M_KEY and M_ID that should have been assigned to them by the SA when they were approved as an OEM.

The MapLink S-63 SDK class `TSLS63UserPermitGenerator` takes each of these three parameters as input to its `generateUserPermit` method. The M_KEY and M_ID parameters should be provided as ASCII strings, while the user's HW_ID should be wrapped in an instance of the `TSLS63HWID` class. Assuming that MapLink validates all of the passed parameters successfully, then the method will return a success status and provide the User Permit by populating the passed 'result' parameter.

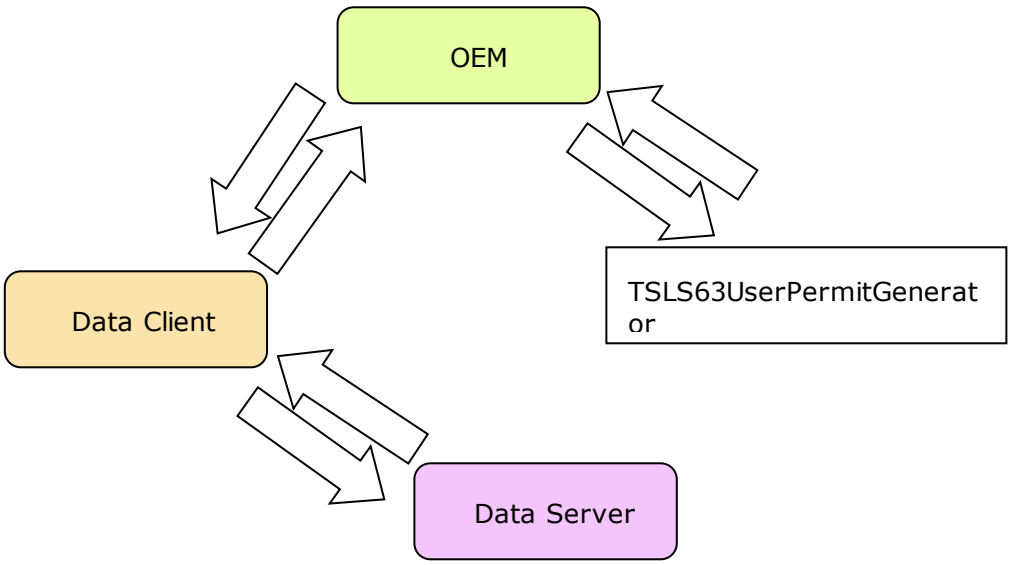


Figure 2

(1) The user of the Data Client makes a User Permit request to the OEM passing their HW_ID.

(2) The OEM passes the user's HW_ID along with their SA assigned M_KEY and M_ID to the `TSLS63UserPermitGenerator`'s `generateUserPermit` method.

(3) The result of the generation call will be the User Permit.

(4) The OEM provides the User Permit to the Data Client user.

(5) The Data Client user uses the User Permit to make a purchase request to a DS.

(6) The DS returns a set of Cell Permits in the form of a PERMIT.TXT file.

2.6. Media Ingest

The Media Ingest functionality is the first of the MapLink S-63 SDK building blocks that should be incorporated in the Data Client. It provides the ability to ingest S-63 Exchange Sets into a Data Client held data store of ENC cells. By performing this initial ingest process the SDK is converting the encrypted S-63 data into an encrypted form suitable for fast display in MapLink.

This section will cover the steps required to utilise this functionality.

2.6.1. Media Ingest Manager and Setup

2.6.1.1. Creation

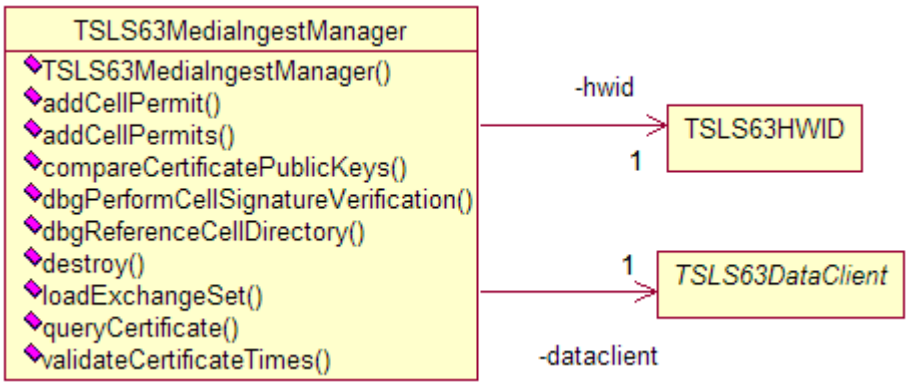


Figure 3 – UML diagram of the `TSLS63MediaIngestManager` and its relationships

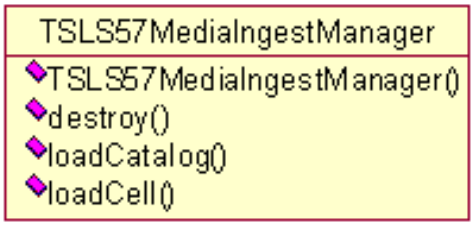


Figure 4 – UML diagram of the `TSLS57MediaIngestManager`

There are two media ingest manager classes. This section discusses only the `TSLS63MediaIngestManager`. The `TSLS57MediaIngestManager` is a simpler version specifically for loading S-57 data (unencrypted).

When constructing the `TSLS63MediaIngestManager` a number of parameters are required that will be used for exchange set ingests throughout its lifetime. These parameters cannot be changed later in the lifetime of the manager.

The list below documents these parameters and their meaning:

- The data client interface: This is the OEM implementation of the `TSLS63DataClient` callback class through which the manager will communicate with the Data Client during the ingest process. Section 2.7.1 describes the different callbacks and at which stages they will be called during the ingest.
- The certificate: This should be the path to the IHO's X.509 certificate held securely on the Data Client user's machine. This is discussed in section 2.4.1.
- The Data Client user's HW_ID: The unique 5-digit hexadecimal value that the OEM has assigned the user should be passed into the construction of a `TSLS63HWID` class that is in turn passed as this parameter.

- The MapLink Version: This is the version of MapLink at which the media ingest manager will produce output data at. This is configurable so that in future releases of MapLink, the data can be produced to be compatible with older versions. It should be noted that the manager cannot produce data in versions prior to MapLink 5.4 as this was when the S-63 SDK was added.

Should any of the parameters passed to the manager prove to be invalid then an error will be raised on the MapLink error stack and all subsequent calls to the manager will fail. Additionally, should the certificate prove to not exist at the path provided then "`SSE 05`" will be raised via the data client interface or should the certificate prove to be invalid or inappropriate then S-63 Error Code "`SSE 08`" will be sent to the `TSLS63DataClient::notifyError` callback.

2.6.1.2. Permit Ingest

Once the media ingest manager has been constructed, the first operation that the Data Client should perform is the ingest of the set of Cell Permits that the user has been provided with. The DS will provide to the user a PERMIT.TXT file containing all of the Cell Permits which they have purchased. As the user may subscribe to multiple Data Servers or a particular DS may issue multiple PERMIT.TXT files due to separate purchases, the user may need to ingest multiple different files and as such this process may need to be repeated a number of times.

The ability to load PERMIT.TXT files is provided via the `TSLS63CellPermitCollection` class, which is creatable via its static `create` method. This method takes the path to the PERMIT.TXT file and additionally the OEM implementation of the `TSLS63DataClient` callback interface class. Should the PERMIT.TXT file not be found at the location specified, then S-63 Error Code "`SSE 11`" will be raised via the callback interface ([1] "*Section 10.5.1: Check for a Cell Permit File*") while should the format of the file not be valid then `SSE 12` will be raised ([1] "*Section 10.5.2: Check Cell Permit Format*").

Assuming that the PERMIT.TXT file was loaded successfully, the class instance returned from the `create` method call can be queried for Cell Permits it contained. A single Cell Permit entry from the file is represented by an instance of the non-user-creatable `TSLS63CellPermit` class. This class in turn provides the ability to query the details of the entry.

The `TSLS63CellPermitCollection` class instance, or a subset of the `TSLS63CellPermit` instances it contains, should be added to the media ingest manager via the `addCellPermits` or `addCellPermit` methods.

A further '`dsID`' parameter is optional to these methods to handle PERMIT.TXT files conforming to V1.0 of the S-63 standard. The publicly available 2 character Data Server ID should be passed as this parameter so that MapLink can identify from which DS the permits were issued. The OEM can determine whether a loaded PERMIT.TXT file requires this parameter to be passed by querying whether the `dataServerID`[2] method on one of the contained `TSLS63CellPermit` instances returns NULL. Failure to provide a valid value for this parameter when it is required will result in an error being raised on the MapLink error stack and a failure return from the method.

As each permit is added to the manager, it performs the first stage of validation[3] on the permit which involves checking the contained checksum. This ensures that the permit has not been tampered with since it was issued by the DS. Should the Cell Permit Checksum prove to be

[2] The .NET versions of the `TSLS63CellPermit` class expose this functionality via a property rather than a function.

[3] MapLink performs later validation, such as checking that the HW_ID correctly decrypts the Cell Keys, when the Cell Permit is used to ingest an ENC Cell file or files.

incorrect then `SSE 13` will be raised via the callback interface ([1] "*Section 10.5.4: Check Cell Permit Check Sum*").

2.6.1.3. Exchange Set Ingest

Once a `TSLS63MediaIngestManager` has been properly constructed and populated with the Data Client user's set of cell permits, the ingest of an exchange set can begin. The Data Client should initiate an ingest by calling the manager's `loadExchangeSet` method, passing all required parameters – the meaning of which this section will describe.

An S-63 exchange set must contain two files that describe their contents; the PRODUCTS.TXT file and the SERIAL.ENC. These files usually reside in the INFO and ENC_ROOT root directories of the exchange set respectively although this is not always the case as the IHO supplied S-63 test data demonstrates. The path to the PRODUCTS.TXT should be passed as the first parameter to the `loadExchangeSet` method whereas the SERIAL.ENC file should be loaded first into an instance of the `TSLS63SerialENCFile` class and passed as the second parameter.

A SERIAL.ENC file can be loaded by passing the path to the `TSLS63SerialENCFile` class's static `create` method, which on successful loading of the file will return a class instance[4]. Should the loading of the file fail or the path passed be invalid then an error will be raised on the MapLink error stack. The instance of the `TSLS63SerialENCFile` class returned can then be queried for the file metadata and the `TSLS63SerialENCRecord` instances, that represent entries in the file, it contains.

The final parameter to the `loadExchangeSet` method is the path to the certificate used to sign the data. Under no circumstances should this path be set to the certificate contained in the exchange set. In most cases this path should be the path to the SA's certificate held securely on the Data Client and passed during the construction of the manager. If the data was self-signed by the DS rather than the SA, this should be set to the path to the securely held DS certificate installed on the client machine in a carefully controlled manner.

The return value of the method indicates whether the ingest completed without encountering any fatal errors. It is therefore possible that it will return true despite errors having been raised for a particular cell but the ingest process was able to proceed with further cells.

All errors that are encountered, regardless of whether they raise a `notifyError` callback or not, will result in an entry being placed on the MapLink error stack. It is therefore vital that the Data Client checks the MapLink error stack after this method has returned to determine if the ingest process completed sucessfully.

2.7. The Ingest Process

The result of the ingest process should be that the Data Client populates and updates their data store with the ENC cell data that the exchange set provides. This is assuming that the user has sufficient cell permits for the data and those permits are valid for that data.

MapLink will provide the actual cell data in two forms, both encrypted:

1. High performance TMF encoded data, suitable for display.
2. Intermediary data, suitable for applying data updates. This will only be required when applying updates to the ENC cell from the store through the ingest of an exchange set.

The encryption used by the manager for these cell files is the same as employed by S-63, using the same cell keys as provided by the Cell Permit. As such the cell permits will be required when displaying the data to allow decryption.

2.7.1. Ingest Callbacks and How to Process them

The `TSLS63DataClient` and interface class defines a number of callback methods that are used to keep the Data Client aware of the progress of the ingest and also to request additional information that the manager may require. The same callback interface is used when loading PERMIT.TXT files[5] and also by the `TSLS63DataLayer` when displaying the ingested data[6].

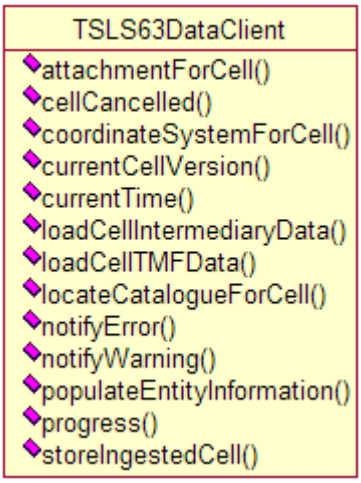


Figure 5 - The `TSLS63DataClient` interface class methods

To load S-57 data you need to use the `TSLS57DataClient`.

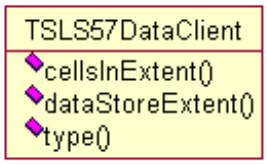


Figure 6 - The `TSLS57DataClient` interface class methods

This section outlines the callback methods that could be called during the ingest on an exchange set and how the Data Client should respond to them.

NOTE: The TSLS57DataClient inherits from the TSLS63DataClient.

2.7.1.1. notifyError

The `notifyError` method will be called by the manager whenever an S-63 defined error is encountered. It won't however be called if MapLink encounters an error that is not defined in the S-63 standard; instead an error will be placed on the MapLink error stack. The S-63 error code will be passed to the interface implementation in the form of an enumeration along with other appropriate information to the error. Should the error relate to a particular cell then the name of the cell and the id of DS that produced the cell will be passed otherwise these parameters will be NULL. A further '`additionalInformation`' parameter may be populated with a description of the error, but this is not intended to be displayed to the Data Client user.

The OEM must handle and display these errors to the Data Client user.

---

[5] See section 2.6.1.2.

[6] See section 2.8.1.1

2.7.1.2. notifyWarning

The `notifyWarning` method will be called by the manager whenever an S-63 defined error is encountered, but for which the S-63 standard does not define an error code. It won't however be called if MapLink encounters an error that is not defined in the S-63 standard; instead an error will be placed on the MapLink error stack. Should the error relate to a particular cell then the name of the cell and the id of DS that produced the cell will be passed otherwise these parameters will be NULL. A further '`additionalInformation`' parameter may be populated with a description of the error, but this is not intended to be displayed to the Data Client user.

The OEM must handle and display these errors to the Data Client user.

2.7.1.3. progress

The `progress` method will be called throughout the ingest process to allow the Data Client to provide some feedback to the user as to how much of the ingest process has completed and therefore how much is left. It also allows the Data Client to abort the ingest, although it is the Data Client's responsibility to rollback the state of data store to before the ingest began.

The Data Client does not need to respond to this callback, instead they could simply provide an implementation that returns that the ingest process should continue.

2.7.1.4. currentTime

To allow the Data Client to use an external source, such as a GPS device, to provide accurate time information this method is provided. The manager will not query the system clock for the current time, instead the `currentTime` method[7] will be called once per exchange set ingest so that the Data Client can provide it with the most accurate time information available.

Should the Data Client detect a mismatch between its time sources, it should raise `SSE 14` as stated in section 2.4.3.

2.7.1.5. locateCatalogueForCell

This callback will be invoked for each cell that will be ingested from the exchange set to determine the location of the CATALOG.031 file on the exchange set media. The reason that this is called for each set in the exchange set, despite in most cases the path always being the same, is to deal with the case where an exchange set spans multiple physical media.

This file should be found in the 'ENC_ROOT' directory of the exchange set.

Should the Data Client be unable to locate this file then returning false from this callback will cause the manager to skip that cell and move on to the next without placing an error on the MapLink error stack.

2.7.1.6. currentCellVersion

This callback will be invoked for each ENC cell in the exchange set to determine if the Data Client's data store already contains a version of that cell and if so at what version. Should the Data Client return that they do not have a version of that cell from that data server in their data store, denoted by returning a version of `-1`, then that cell will be treated as a new ingest. If on the other hand the Data Client returns a valid version then MapLink will determine if the exchange set contains any updates or re-issues with a higher version. Should the exchange set

[7] The .NET version of the `TSLS63DataClient` exposes this callback as an abstract property.

not contain a higher version then that ENC cell is ignored and the manager moves on to the next, otherwise it attempts to ingest the necessary updates.

2.7.1.7. coordinateSystemForCell

When the manager determines that a cell is a new ingest it queries, via this method, the coordinate system that it should project the decrypted data into. As MapLink currently only supports the display of ingested cells that use a common coordinate system, it is advisable to use a single coordinate system for all cells ingested into the Data Client's data store.

Should the Data Client not provide a valid coordinate system then an error will be placed on the MapLink error stack and the manager will move on to the next ENC cell.

2.7.1.8. attachmentForCell

The S57 standard [3], on which S-63 is based, provides the ability to add attachments to ENC cell data. These are files contained in the exchange set that usually describe the data in some way.

When the manager encounters one of these attachments in the Cell file it is currently ingesting it will call this callback providing the path to the target file. Additionally it provides two variables that the recipient of the callback should populate to allow the link to the source data to be maintained. The first of these variables should be populated with a two-character ID that can be used to lookup this attachment at display time from the geometry object to which the attached file is attached.

The second should be populated with a unique string value so that multiple attachments on the same geometry object can be differentiated at display time.

Should an update be applied to a cell that contains one or more attachments, this callback will be called for each attachment that the updated cell contains, including those that may not be contained in the current exchange set.

2.7.1.9. populateEntityInformation

As each ENC cell is ingested, and again when updates are applied to a cell in the Data Client's data store, each TMF entity will be passed in turn to the Data Client. This allows rendering attributes to be applied to the data and to allow the Data Client to make changes to the geometry.

Additionally, a further parameter is provided to permit the Data Client to associate additional entities with the entity passed. These associated entities will be added to the encrypted data that the Data Client will be told to store in their data store via a later `storeIngestedCell` callback.

2.7.1.10. cellCancelled

S-63 exchange sets permit the ability to mark an ENC Cell as being 'cancelled'. This is usually the case where a cell is replaced by a different cell or cells. Should a cell that the Data Client holds in its data store be deemed cancelled through the ingest of an exchange set, this method will be called to query the Data Client as to how to deal with the cell.

The Data Client may either opt to keep the cell in their data store or have it removed. If the Data Client keeps the cell in their store, the manager will load the cell's TMF data via a later callback, update it and finally return it to the store through another callback. This is required to ensure that, at display time, a warning is shown to the user notifying them of this cancellation.

2.7.1.11. loadCellTMF

When ingesting an update to a cell already held in the Data Client's data store from a previous ingest, this method will be called to load the cell's encrypted TMF from the store. An instance of the `TSLS63EncryptedData` class will be passed and it is expected that the Data Client populates it with data.

The Data Client can opt to cancel the ingest of this update or updates via the return value of the call. Equally it can opt to cancel the entire ingest process.

2.7.1.12. loadCellIntermediaryData

When ingesting an update to a cell already held in the Data Client's data store from a previous ingest, this method will be called to load the cell's encrypted Intermediary Data from the store. An instance of the `TSLS63EncryptedData` class will be passed and it is expected that the Data Client populates it with data.

The Data Client can opt to cancel the ingest of this update or updates via the return value of the call. Equally it can opt to cancel the entire ingest process.

2.7.1.13. storeIngestedCell

When the manager has finished ingesting all required files for a particular cell, this callback method will be made to prompt the Data Client to store the new or updated data. The four possible scenarios for calling this method are listed below:

- A cell is ingested for which no previous version is held in the data store.
- An update or updates for a cell are ingested when an older version of the cell is held in the data store.
- A cell contained in the data store has been cancelled by the exchange set but the Data Client opts to keep the cell in the data store.[8]
- An update or updates from the exchange set could not be applied to a data store held version of the cell previously ingest due to the expiry of the cell permit. The cell TMF data will need to be updated so that at display time a warning is shown indicating that the cell data is out-of-date.[8]

The parameters passed to this function are as follows:

- `cellTMF` : An instance of the `TSLS63EncryptedData` class containing the encrypted TMF.
- `cellIntermediaryData` : An instance of the `TSLS63EncryptedData` class containing the encrypted Intermediary Data.
- `cellName`/`dataServerID` : Parameters used to identify the cell being stored
- `cellExtent`: An instance of the `TSLGeodeticExtent` that represent the bounding Latitude and Longitude of the cell. These values will need to be stored for use at display time.
- `cellEdition`/`cellUpdateNumber`: The version of the cell to store. The `cellUpdateNumber` should be returned from the `cellVersion` callback when the manager requests the currently held version in future ingests.

[8] This is a special case where the `cellIntermediaryData` parameter to the function will be `NULL` as this data does not need updating.

2.7.1.14. TSLS57DataClient

This class inherits from TSLS63DataClient. The methods for the TSLS63DataClient need to be implemented as well as the methods cellsInExtent and dataStoreExtent.

2.7.2. Data Client's Data Store

As mentioned in previous sections, the ingest process will pass data to and query from the Data Client's data store through the use of callbacks on the `TSLS63DataClient/TSLS57DataClient` interface class. How this data store is implemented is left to the OEM to decide, however this section details a few tips that aid in achieving the best performance when ingesting exchange sets and displaying data. It will also repeat the details covered in previous sections as to what the Data Client should store in this data store.

For each Cell Name/Data Server ID combination, the data store should be able to store:

- The extent of the cell as provided by the initial ingest through the `storeIngestedCell` callback. This will be required when displaying the data via the `TSLS63DataLayer`.
- The current cell version as provided by the latest ingest through the `storeIngestedCell` callback. This will be required when both ingesting later exchange sets and when displaying the data via the `TSLS63DataLayer`.
- The encrypted cell data in both TMF and Intermediary forms. The encrypted TMF form will be required when ingesting and displaying the data, whereas the Intermediary form will only be required when ingesting future updates.
- Optionally, the OEM may wish to store the cell attachments in their data store. The paths to these attachments will be provided by the `attachmentForCell` callback method.

For the `TSLS63DataStore` OEM should be careful to index the data using both the cell name and DS id as the same cell names will be used by different Data Servers.

Ideally the storage of extent and cell version should be structured in a form that allows quick lookup of these values. Whereas the storage of the encrypted cell data and potentially their attachments need not require such constraints. The actual storage requirements will be dictated by the Data Clients performance requirements (display and ingest).

2.8. Display and Decryption

The decryption and display of the Data Client's data store data is performed by the `TSLS63DataLayer`. This layer acts in much the same way as any other MapLink data layer so is therefore displayable through a MapLink drawing surface. This section describes how to construct this layer, add the user's cell permits and how to respond to the callbacks the layer will invoke at display time.

The layer can be used to display both S-63 data and S-57 data.

2.8.1. The S-63 Data Layer and Setup

2.8.1.1. Creation

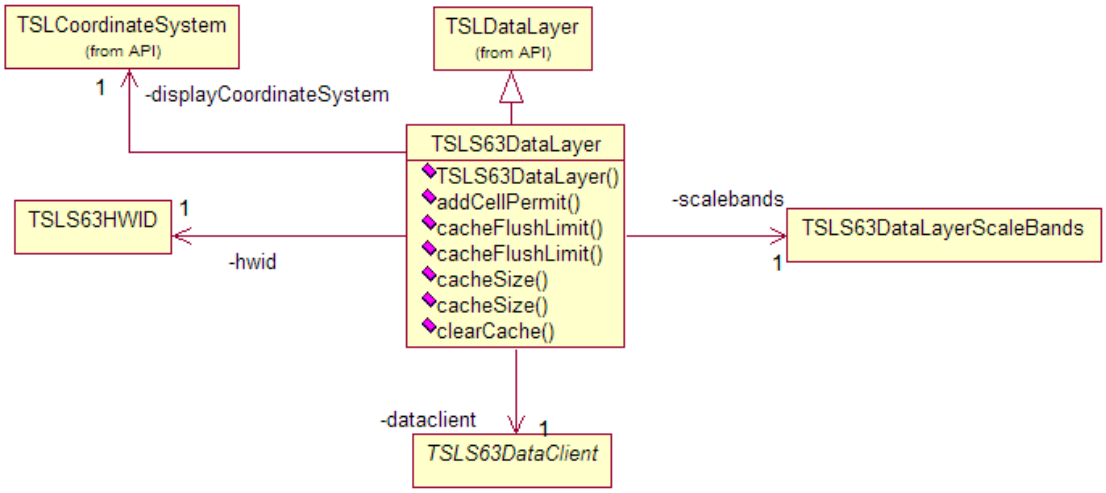


Figure 7 - UML diagram of the C++ `TSLS63DataLayer` and its relationships

When constructing the `TSLS63DataLayer` a number of parameters are required that will be used for display of the data throughout its lifetime. These parameters cannot be changed later in the lifetime of the layer.

The list below documents these parameters and their meaning:

- The data client interface: This is the OEM implementation of the `TSLS63DataClient` callback class through which the layer will communicate with the Data Client during the decryption and display process. Section 2.8.4 describes the different callbacks and at which stages they will be called during this process.
- The Data Client user's HW_ID: The unique 5-digit hexadecimal value that the OEM has assigned the user should be passed into the construction of a `TSLS63HWID` class that is in turn passed as this parameter.
- The `TSLCoordinateSystem` the layer is to use: This should be the same coordinate system that was used when ingesting the data through the `TSLS63MediaIngestManager`. Should the coordinate system differ from any of the cell data files loaded at display time, an error will be placed on the MapLink error stack and that cell disregarded.
- A populated `TSLS63DataLayerScaleBands` class instance: This class controls which S57 bands will be visible at a particular zoom level.

Should any of the parameters passed to the layer prove to be invalid then an error will be raised on the MapLink error stack and all subsequent calls to the layer will fail.

2.8.1.2. Scale Bands

S57 defines a set of scale at which cells are intended to be displayed. The name of the cell indicates which scale band that particular cell falls into, denoted by the third character in the cell name.

Band	Name	Scale Range	Range (approximate)
1	Overview	<1:1499999	1:150000 == 96NM
2	General	1:350000 – 1:1499999	1:350000 == 24NM
3	Coastal	1:90000 – 1:349999	1:90000 == 6NM
4	Approach	1:22000 – 1:89999	1:22000 == 1.5NM
5	Harbour	1:4000 – 1:21999	1:4000 == 0.25NM
6	Berthing	>1:4000	

Figure 8 – The S57 defined scale bands ([4] "*Section 2.1: Navigation Purpose*").

Although cells from different scale bands are not intended to be displayed together, MapLink allows a certain level of flexibility when determining which scale bands are visible at a certain zoom level. Multiple scale bands are permitted to be visible at a particular zoom level and additionally MapLink does not enforce the zoom level that a band becomes visible or invisible at matches the S57 intended value.

The way that the OEM defines which scale bands are visible at a certain zoom scale is handled through the `TSLS63DataLayerScaleBands` class that is passed to the data layer when it is constructed. For each scale band, the OEM can define an upper and lower threshold in Map Units (MUs) per Device Units (DUs). These thresholds can overlay between bands, but a particular band cannot have multiple sets of thresholds for which it is visible. Should multiple scale bands be defined as visible at a particular zoom level, the order in which the band was added to the class determines which bands appear above or below it.

Refer to the API documentation for the `TSLS63DataLayerScaleBands` class for more details.

2.8.1.3. Permit Ingest

Once the S-63 data layer has been constructed, the first operation that the Data Client should perform is the ingest of the set of Cell Permits that the user has been provided with. The DS will provide to the user a PERMIT.TXT file containing all of the Cell Permits which they have purchased. As the user may subscribe to multiple Data Servers or a particular DS may issue multiple PERMIT.TXT files due to separate purchases, the user may need to ingest multiple different files and as such this process may need to be repeated a number of times.

The ability to load PERMIT.TXT files is provided via the `TSLS63CellPermitCollection` class, which is creatable via its static `create` method. This method takes the path to the PERMIT.TXT file and the OEM implementation of the `TSLS63DataClient` callback interface class. Should the PERMIT.TXT file not be found at the location specified, then `SSE 11` will be raised via the

callback interface ([1] "*Section 10.5.1: Check for a Cell Permit File*") while should the format of the file not be valid then `SSE 12` will be raised ([1] "*Section 10.5.2: Check Cell Permit Format*").

Assuming that the PERMIT.TXT file was loaded successfully, the class instance returned from the `create` method call can be queried for Cell Permits it contained. A single Cell Permit entry from the file is represented by an instance of the non-user-creatable `TSLS63CellPermit` class. This class in turn provides the ability to query the details of the entry.

The `TSLS63CellPermit` instances that the `TSLS63CellPermitCollection` class contains, should be added to the data layer via the `addCellPermit` method.

The cell's geodetic extent should also be passed in the form of a `TSLGeodecticExtent` class instance. This extent is provided to the Data Client during the ingest process through the `storeIngestedCell` callback.

A further '`dsID`' parameter is optional to these methods to handle PERMIT.TXT files conforming to V1.0 of the S-63 standard. The publicly available 2 character Data Server ID should be passed as this parameter so that MapLink can identify from which DS the permits were issued. The OEM can determine whether a loaded PERMIT.TXT file requires this parameter to be passed by querying whether the `dataServerID`[9] method on one of the contained `TSLS63CellPermit` instances returns NULL. Failure to provide a valid value for this parameter when it is required will result in an error being raised on the MapLink error stack and a failure return from the method.

As each permit is added to the layer, it performs the first stage of validation[10] on the permit which involves checking the contained checksum. This ensures that the permit has not been tampered with since it was issued by the DS. Should the Cell Permit Checksum prove to be incorrect then `SSE 13` will be raised via the callback interface ([1] "*Section 10.5.4: Check Cell Permit Check Sum*")

2.8.2. The Decryption and Display Process

Only when S-63 Data Layer comes to draw the cells does it load the cell data from the Data Client's data store. Cells that are considered for drawing must have had a valid cell permit added to the layer and been configured via the `TSLS63DataLayerScaleBands` to be visible at the current zoom level.

Before the layer calls this load callback, it first performs additional validation on the cell permit including checking whether the permit has expired or is soon to expire. The former case will raise `SSE 25` and potentially `SSE 15` whereas the later will raise `SSE 20`.

Once the layer has retrieved the cell's TMF data from the Data Client via a callback, it attempts to decrypt and decompress it using the Cell Permit. Should it fail to decrypt the Cell Keys from the Cell Permit then `SSE 19` will be raised and if it fails to decrypt/decompress the data an error will be placed on the MapLink error stack. In this case that cell will be ignored in all future draws.

Should the decrypted/decompressed TMF data indicate that the cell data is out-of-date then the Data Client will be notified of error `SSE 27` while if the cell is cancelled then the Data Client will be notified by a `notifyWarning` callback.

---

[9] The .NET versions of the `TSLS63CellPermit` class expose this functionality via a property rather than a function.

[10] MapLink performs later validation when the Cell Permit is used to decrypt data files from the Data Client's data store. This includes checking that the HW_ID correctly decrypts the Cell Keys.

It should be noted that if the cell data is found to reside in the layer's cache then neither the out-of-date error nor the cancelled warning will be raised again. Additionally should the tile be redrawn, yet has been removed from the cache in the meantime, these errors/warnings will be raised again when the tile is reloaded from the Data Client's data store.

2.8.3. Data Caching

In much the same way that a number of other MapLink data layers do, the S-63 Data Layer attempts to reduce the number of times that a cell data file has to be loaded from the Data Client's data store through memory caching. This will store the decrypted TMF data in memory for the most recently used cells until the memory cache has reached a user defined limit, at which point the least recently used cell will be removed. The data layer exposes the ability to set and query this cache limit size as well as additional functions to clear the cache and set the cache flush limit.

It should be noted that the encrypted TMF the data store houses is also compressed, whereas the cache holds the data in uncompressed form. Therefore the size of the data in the data store does not correspond to how large it will be when populated to the layer's cache.

Obviously the persistent cache functionality that MapLink data layers such as the `TSLMapDataLayer` and `TSLRasterDataLayer` provide would defeat the purpose of the data protection offered by S-63 so are therefore not offered.

2.8.4. Display Callbacks and How to Process them

The `TSLS63DataClient` interface class defines a number of callback methods that are used to request additional information that the layer may require. The same callback interface is used when loading PERMIT.TXT files[11] and also by the `TSLS63MediaIngestManager` when ingesting data[12].

The `TSLS57DataClient` extents the callback interface by two methods.

This section will outline the callback methods that will be called during the decryption and display phase and how the Data Client should respond to them. **Figure 5** shows all the callback methods that this class defines although the data layer only utilises a subset of these.

2.8.4.1. notifyError

Refer to section 2.7.1.1.

2.8.4.2. notifyWarning

Refer to section 2.7.1.2.

2.8.4.3. currentTime

Refer to section 2.7.1.4.

[11] See section 2.8.1.3

[12] See section 2.6.1.1

2.8.4.4. loadCellTMF

When the data layer is about to draw a cell that is does not currently have in its cache, this method will be called to load the cell's encrypted TMF from the store. An instance of the `TSLS63EncryptedData` class will be passed and it is expected that the Data Client populates it with data.

The Data Client can however opt to cancel the display of this cell via the return value of the call. Equally it can opt to cancel the display of any further cells. By cancelling the current cell or the current and all further cells, when the layer comes to draw the next time it will re-request the data from the Data Client. The 'tile not found' return code on the other hand will cause that cell to be ignored in this and all future draws.

2.8.4.5. TSLS57DataClient display interface

- cellsInExtent

  This method should return a list of S-57 cell names which are in the requested extent. These are the cells which will be drawn.

- dataStoreExtent

  This method should return the total extent of the S-57 cells in the data store.

3. THE REFERENCE WORKFLOW AND OEM TEST REQUIREMENTS

The C++ 'S-63 Reference Workflow' application supplied with the MapLink S-63 SDK serves a dual purpose:

- It demonstrates how to utilise the functionality covered in this document.
- It demonstrates how the MapLink S-63 SDK can be used to fulfil the OEM test requirements specified by the "S63 Implementation Guide" [2], [9].
- It demonstrates the S-63 Workflow and S-57 Workflow.
- Demonstrates how to use the S-52 SDK.

The MapLink S-63 SDK is intended to abstract the intricacies of the S-63 standard from the OEMs whom use it to produce their Data Client through providing a simple API that is easy to incorporate into their application. Unfortunately this design does not lend itself perfectly to the OEM's testing requirements as certain tests expect a modular application in order to prove that certain errors are raised when necessary. For this reason the `TSLS63MediaIngestManager` exposes two methods that are purely intended to be used when completing the 'S-63 OEM 1.1' testing:

- `dbgPerformCellSignatureVerification`: This method is used to disable the cell signature verification check that is performed to ensure that the cell file has not been tampered with since publication and is issued by an SA approved DS.

  Test 5.5 ([2] "*Section 5.6.5: Test 5.5 – Decrypt Erroneous ENC*") supplies three erroneous ENC cell files yet unfortunately doesn't supply equivalent signature files. Therefore in order to achieve the required error message that the test mandates, the signature validation for the test must be disabled through calling this method.

  To prevent this method from being abused, should the cell signature validation be disabled yet the remaining checks performed in the cell pass, the manager will not complete the ingest of this cell.

- `dbgReferenceCellDirectory`: This method is used to compare the decrypted/decompressed data from an exchange set against files contained in a reference directory.

  Test 5.2 ([2] "*Section 5.6.2: Test 5.2 – Decrypt ENC Base Cells and Update File*") requires that the decrypted/decompressed cell files are checked for correctness against unencrypted/uncompressed reference files.

  Should a reference file not be found in the directory specified or the file found not match, an error will be placed on the MapLink error stack. In both cases the error raised with have the error category `TSLErrorCategoryDebug` which will assist in quickly differenciating errors raised through this check against errors raised for other reasons.

Neither of these methods should ever be used in a deployed Data Client.

The 'S-63 OEM 1.2' testing has also been mainly implemented in the Workflow sample. Where we are unable to implement the test we log the detail.

The code to the Workflow Sample is provided and you are recommened to read this in conjuction with the OEM tests supplied by IHO.

4. DATA PREPARATION FOR S-52 / AML

The S-57 and S-63 data has to be specifically prepared for displaying using the S-52 or AML Dynamic Renderer.

The following two sections are to highlight the necessary steps required.

4.1. ENC / AML Recommendation

ENC and AML are products that use S-57 data format.

ENC and AML are have specific use cases, as such it is recommended that they should not be included in the same map/datastore and that they should be rendered using the correct Dynamic Renderer.

4.2. Preparation using the S63 SDK

The S63 SDK provides a mechanism for passing configuration information to the S-57 filter.

The contents and location of the S-57 configuration file for S-52 / AML rendering are detailed in section 5.3.2.

The default configuration file can be found here:

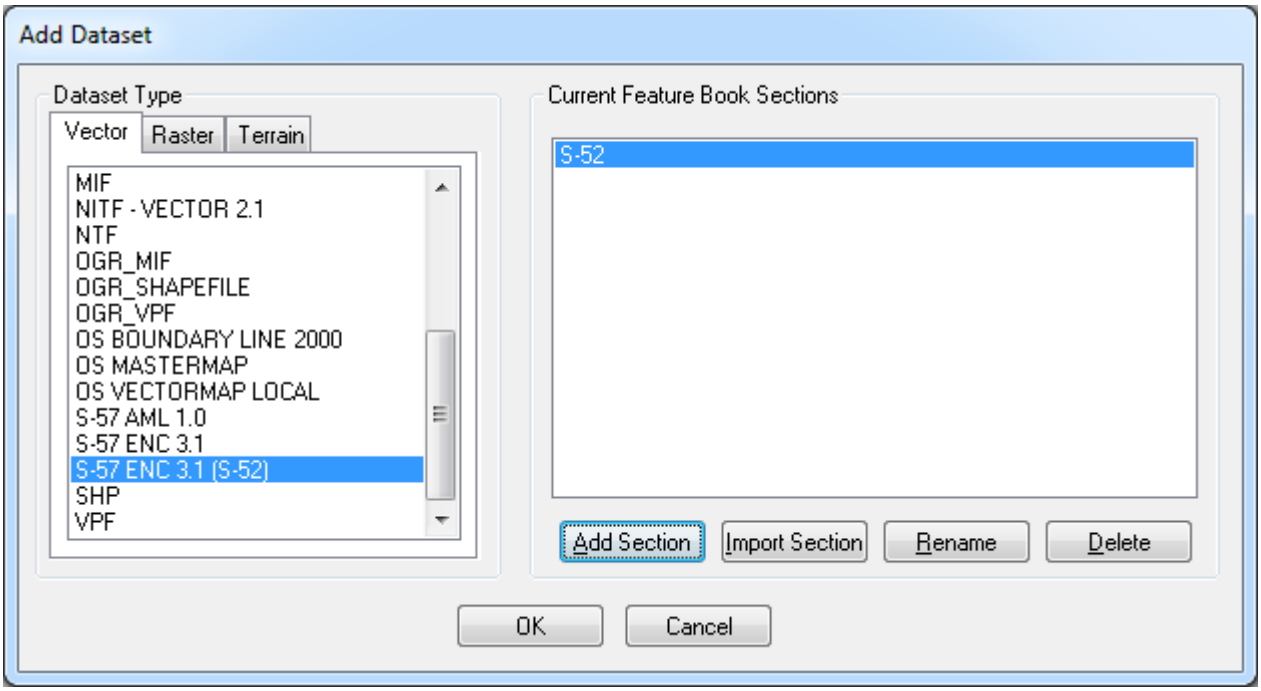
- `config/s52/S57filter.ini`

The Media managers for S-57 and S-63 can be passed the configuration files when data is loaded. The methods used are:

- `TSLS63MediaIngestManager::loadExchangeSet`
- `TSLS57MediaIngestManager::loadCatalog`
- `TSLS57MediaIngestManager::loadCell`

4.3. Preparation using MapLink Pro Studio

When you create a map in MapLink Studio you need to select the 'S-57 ENC 3.1 (S-52)' dataset configuration as shown below.



To help you we have created a basic template and a default feature book which can be found in:

- projects/S52 ENC
- projects/S52 AML
- config/featurebooks/s52.fbk

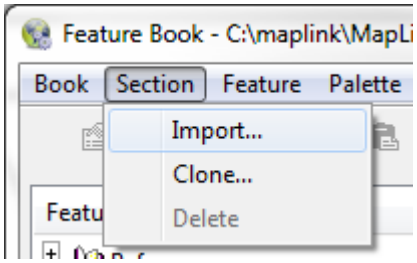
If you decide not to use either of these you need to ensure that once you have loaded all your S-57 data that you load the correct rendition file for S-52 or AML from:

- config/S52/S52.rnd
- config/S52/AML.rnd
- config/S52/AMLv2.rnd
- config/S52/AMLv3.rnd

4.3.1. Loading a Feature Book

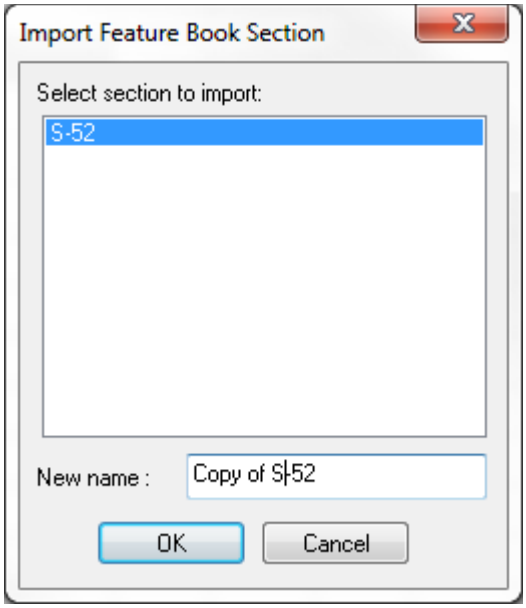
You should load the feature book before you load any data as you need to associate the data with a feature book section.

Open the feature book and select the Section menu. Next select Import.



Navigate to where the `.fbk` file is and select the file.

You will be prompted with which feature book section to load:

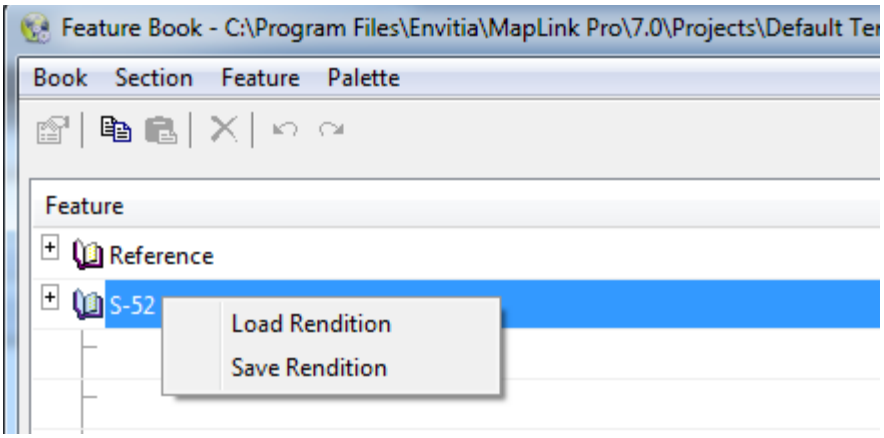


Change the name and click OK.

You should now load the Rendition into this feature book section.

4.3.2. Loading S-52 / AML Rendition

Loading the S-52/AML rendition file can be achieved by opening the Feature Book in MapLink Studio and loading the rendition as shown below (select the Feature Book section and press the right mouse button):



Load the appropriate rendition file. This will over-write any rendition that you may have setup with standard fallback rendering for S-52/AML.

Note: The rendering will still not appear as you might expect until you load the map into the Map Viewer. A very large proportion of the rendering is dynamic and defined at runtime.

5. S-52 SDK

The S-52 SDK consists of a number of components:

- MapLink Pro Studio S-52 Dataset configuration.
- Modifications to S-63 SDK for the S-57 Workflow.
- S-52 Dynamic Renderer – for drawing the data dynamically.
- S-52 Dynamic Renderer Plugin – for use with the Web Map Server (Super Map plugin).
- AML Dynamic Renderer – for drawing AML data dynamically.
- AML Dynamic Renderer Plugin – for use with the Web Map Server (Super Map plugin).

This part of the document principally deals with the S-52 and AML Dynamic Renderer and how to import the data for display with the Dynamic Renderer.

The previous part of the document deals with how to import and manage S-57/S-63 data. You will need to use the correct filter configuration to use the S-52 and/or AML Dynamic Renderer.

5.1. Library Usage and Configuration

This section describes how to link in the MapLink S-52. Currently only the C++ and .NET languages are supported.

The S-52 SDK either requires a Map created with the S-52 or AML configuration of the S-57 filter in MapLink Pro Studio (see 5.2.2) or via the S-63 Data-layer (see section 4.1).

5.1.1. C++

The C++ version of the MapLink S-52 SDK is supplied as two versions; a Debug version and a Release version.

It should be noted that the library to be linked with should be determined by the Core SDK library that you are using within your application. For example, if you are using the DLL Release mode version of the Core SDK (`MapLink.lib/MapLink64.dll`) then you must also use the equivalent S-52 SDK library (`MapLinkS52.lib/MapLinkS52.lib`).

`MapLinkS52.lib` or `MapLinkS5264.lib` Release mode, DLL version. Uses Multithreaded DLL C++ run-time library. Your application must also link the MapLink CoreSDK library `MapLink.lib/MapLink64.lib`. Requires the `S57Filter.DLL/ S57Filter64.DLL` at runtime	`MapLinkS52d.lib` or `MapLinkS5264d.lib` Debug mode, DLL version. Uses Debug Multithreaded DLL C++ run-time library. Your application must also link the MapLink CoreSDK library `MapLinkd.lib/MapLink64d.lib`. Requires the `S57Filterd.DLL/ S57Filter64d.DLL` at runtime. Should not be redistributed to deployments.

This SDK does not have an unlock code nor is it locked via the License Key Administrator. However it is dependent upon the S-57 filter and potentially the S-63 SDK. The S-63 SDK has two runtime unlock codes; one for using S-63 and S-57 media and the other just for using S-57 media.

5.1.2. .NET

The .NET interface to the MapLink S-52 SDK is exposed by the `Envitia.MapLink.S52.DLL` assembly under the namespace `Envitia.MapLink.S52`. This library is dependent on both the Release mode C++ MapLink S-52 SDK and the Core .NET SDK assembly `Envitia.MapLink.DLL`. All dependencies for both these libraries will need to be redistributed.

The standard .NET assemblies are built using the .NET Framework 4 although additional versions are supplied built using Framework 4. Additionally, a 64-bit version of the assemblies are included. Refer to the MapLink Developer's Guide for further information.

Please refer to the Release Notes for any limitations to the .NET SDK.

5.2. S-52 / AML Dynamic Renderer

The S-52 Dynamic Render (`TSLS52DynamicRender`) and the AML Dynamic Renderer (`TSLAMLDynamicRenderer`) makes use of the configuration files discussed in section 5.3.

The rest of this section takes you through the minimal setup of the Dynamic Renderer in a number of different scenarios.

5.2.1. S-52 / AML Basic Dynamic Renderer setup

The S-52 specific configuration file location is specified via the environment variable S52_HOME or by calling `TSLS52DynamicRenderer::s52HomeDirectory(path)` and/or `TSLAMLDynamicRenderer::s52HomeDirectory(path)`. It is very important to set this up correctly or the configuration files described in section 5.3 will not be found and the drawing will not be correct.

The S-52 home directory should be the same for both the S-52 and AML Dynamic Renderers.

An S-52 State object is required if you want to pass data to the renderer and control what is displayed and potentially how it is displayed.

```
TSLS52StateObject *m_s52StateObject;
```

In case a Mariner rendering procedure is called. The following may be required (see Limitations below).

```
TSLS52UserRenderingProcedure * userRenderingProcedure;
```

Create the objects...

```
// Set up S52 rendering objects incase the user requests S52 rendering.
m_s52StateObject = new S52StateObject();
m_userRenderingProcedure = new S52UserRenderingProcedure();
```

We can now create the S-52 Dynamic Renderer:

```
// Create the S52 renderer.
m_s52DynamicRenderer = new TSLS52DynamicRenderer(m_s52StateObject,
                                                 m_userRenderingProcedure);
```

Add the S-52 Dynamic Renderer to the Drawing Surface:

```
// A dynamic renderer is required for each map that has S52.
// A dynamic renderer has to be created for each Map Data-layer.
// It does not make sense to use this Dynamic Renderer with individual
// features.
surface->addDynamicRenderer(m_s52DynamicRenderer, -1, layer_name.c_str());
```

Load the S-52 default rendering to the surface or preferably the layer (see paragraph 5.3.1):

```
m_drawingSurface->loadRendering(pathToS52RenditionFile);
```

The variable `pathToS52RenditionFile`, should contain the full path to the `S52.rnd` in this case.

The AML Dynamic Renderer is setup in the same way as the S-52 Dynamic Renderer. The AML Dynamic Renderer will pick up the rendering lookup tables from the sub-directory `AML`.

The S-52 specification calls for a No-Data pattern to be drawn where we do not have S-57 data coverage. The following can achieve this requirement:

```
// Create a new no data layer
m_s52NoDataLayer = new TSLS52NoDataLayer(NULL);

// Add it to the surface
doc->addLayerToSurface(NO_DATA_LAYER_NAME, m_s52NoDataLayer,
m_drawingSurface);

// Make it so we cannot select the information.
m_drawingSurface->setDataLayerProps(NO_DATA_LAYER_NAME, TSLPropertySelect,
                                    (TSLPropertyValue)false);
```

5.2.1.1. Limitations

Drawing more than one scale band means that all delay rendered features are drawn. This may appear as double drawing. You could separate out the importing of data so that each scale band is imported into a different store. Each scale band would then be drawn in a different layer. This would create a hierarchy of display for the delayed drawn objects.

Displaying more than one scale band at once can affect performance. We would suggest that the scale-bands displayed are based upon the extents of the imported data and their extent overlapping the view extent.

If there is no data overlapping the view extent for that scale-band then we would suggest that the scale-band should possibly be ignored, otherwise you are likely to not see any data displayed.

Please also refer to section 5.2.3.

5.2.2. MapLink Studio Map

You can generate a Map from MapLink Studio specifically for drawing using the S-52 Dynamic Renderer or AML Dynamic Renderer.

You should use the specific S-57 Dataset configuration for S-52 or AML. This uses the configuration files discussed in section 5.3.

You are strongly advised to load one of the rendition files contained in the `config/s52` directory (`S52.rnd, AML.rnd, AMLv2.rnd, AMLv3.rnd`) file into the feature book section for the S-57 data loaded as this will provide the basic rendering setup for the features. If you do not load the rendition file you may find that not all the features will be drawn correctly. Alternatively, you could use one of the Product Datasets for S52/AML as these contain predefined Feature Book Sections.

The following is example code for displaying a map created for use with the S-52 Dynamic Renderer.

```
// See how many dynamic renderer hints we have
// This is for MapLink 7.1 and onwards only
// - This will detect S-52 maps generated by MapLink 7.0 and newer

bool thisMapContainsS52 =
   TSLS52DynamicRenderer::requiresS52Renderer(m_mapDataLayer);
```

A similar function is provided to detect the AML Dynamic Renderer:

```
bool thisMapContainsAML =
   TSLAMLDynamicRenderer::requiresAMLRenderer(m_mapDataLayer);
```

Once you know that you have a Map that needs an S-52 Dynamic Renderer than you need to create the Dynamic Renderer and an S-52 state object.

```
// Need to create default S52 rendering
if (thisMapContainsS52)
{
  if (!m_S52NoDataLayer)
  {
    // Add the "nodata" background behind everything.
    m_S52NoDataLayer = new TSLS52NoDataLayer(NULL);

    surface->addDataLayer( m_S52NoDataLayer, "s52nodatalayer" ) ;
    surface->setDataLayerProps( "s52nodatalayer", TSLPropertyBuffered, 1) ;
    surface->sendToBack("s52nodatalayer");
  }

  // An S52 state object is required. This allows the developer to change
  // parameters which affect the drawing of features.
  // You need to derive your own state object.
  TSLS52StateObject *stateObject = new S52StateObject;

  // A dynamic renderer is required for each map that has S52.
  // A dynamic renderer has to be created for each Map Data-layer.
  // It does not make sense to use this Dynamic Renderer with individual
  // features.
  TSLS52DynamicRenderer *s52DynamicRenderer =
    new TSLS52DynamicRenderer(stateObject, NULL);
  surface->addDynamicRenderer(s52DynamicRenderer, -1, layer_name.c_str());
}
```

You need to implement similar logic for the AML Dynamic Renderer.

5.2.2.1. Limitations

Tiling of the map will mean that symbols may be displayed more than once over polygons that appear contiguous.

Please also refer to section 5.2.3.

5.2.3. Limitations

The LIGHTS05 symbology procedure requires the ability to display lights when they are not on the screen. This is not possible to do in all cases. A work around is to expand the area that is drawn using the data layer property `TSLPropertyLoadedSymbolsAndTextViewExpansion` (`TSLDrawingSurfaceBase::setDataLayerProps`)

Mariners objects are not supported by the Dynamic Renderer. You can implement these in your application using a Standard Data-layer or Custom Data-layer. A mechanism has been supplied for user drawing if the Mariners rendering procedures are triggered as part of the S-52/AML Dynamic Rendering.

The DATCVR02 conditional symbology procedure cannot be completely implemented in the Dynamic Renderer. The larger scale data availability functionality has to be implemented at the application level (reference 6, section 4.3).

The DATCVR02 procedure calls for the display of a coverage chart. This has to be created at the application level as the cells are imported.

The DATCVR02 non-HO data display is not fully supported. The information could be added to the TMF data if required by the application. This is non-trivial to do, please contact support if you need this functionality.

5.2.4. S-63 Data-layer

The S-52 Dynamic Render works best with the S-63 SDK. You can use either S-57 data or S-63 data.

When you import data you must use the S-57 filter configuration discussed in section 5.3.

The filter configuration files must be correctly setup for loading of AML or ENC to ensure that the correct rendering rules are used when ingesting the S-57 data.

You create the S-52/AML Dynamic Renderer and S-52 state object in the same way as the previous example.

5.2.5. Dynamic Renderer Plugin

An S-52 and an AML Dynamic Renderer Plugin has been created for use with the Super Map Plugin for the MapLink Pro Web Map Server.

The plugin DLL names are:

- `TSLS52DynamicRendererPlugin[64/d/64d].dll`
- `TSLS52DynamicRendererPlugin[64/d/64d].dll`

5.3. Configuration Files

The configuration files for the S-52 SDK are mainly found in the directory:

- `$MAPL_HOME/config/s52`

Configuration files which are not in this directory can be found in:

- `$MAPL_HOME/config`

The additional configuration files are:

Filename	Description
`IHOS57Catalogue_v311.xml`	S-57 version 3.1.1. Object and Attribute catalogue.
`symbols/tslsymbolss52dai.dat`	S-52 symbols – generated from the DAI files. Loaded by default. Standard MapLink Pro format.
`palettes/s52daywebsafe.pal`	S-52 colours for daylight. Loaded by default. Standard MapLink Pro format.
`linestyles/tsllinestyless52.dat`	S-52 line-styles. Loaded by default. Standard MapLink Pro format.
`fillstyles/tslfillstyless52.dat`	S-52 fill-styles. Loaded by default. Standard MapLink Pro format.

See the MapLink Pro Developers Guide for information about the standard resource file formats.

5.3.1. Rendition File

The file `S52.rnd`, `AML.rnd`, `AMLv2.rnd`, `AMLv3.rnd` contains basic rendering setup for all standard S-57 objects for the specific product. The appropriate rendition file should be loaded into an application to provide the basic S-52/AML rendering.

The appropriate rendition file can also be loaded into a MapLink Pro Studio feature book section to provide basic rendering setup for S-57 data processed via Studio.

The file format is complex and we advise that the file is not hand modified. The rendering can be modified programmatically or within MapLink Studio and save a new version of the file.

5.3.2. S-57 Filter MapLink Studio Configuration

The S-57 Filter configuration can be altered by modifying the section entries in mapl.ini.

Settings	Description
`[S-57 *]`	Section Name. There are multiple configurations.
`CONFIGFILE=IHOS57Catalogue_XXXXX.xml`	S-57 Feature/Attribute Catalogue file to use. File used will be dependent upon configuration.
`S52_DISABLE_COSPATIAL_CALCULATIONS=1`	If present disables most of the co-spatial analysis. Only use for a simple viewer or for AML.

5.3.3. S-57 Filter Direct Import Configuration

The S-57 filter configuration file is different from the standard configuration file. The S-52 version can be found in the S-52 directory. This is the configuration file used by MapLink Studio.

The file can be found here:

- `config/s52/S57filter.ini`

Settings	Description
`[S-57]`	Section Name
`AttachmentDirectory=1`	Required
`ENABLES52RENDERING=1`	Enable S-52 Rendering processing. If this is not present then the S-57 imported data cannot be drawn using the S-52 Dynamic Renderer. In this case you would need to use rendition files for styling each scale band.
`ENABLES52AMLRENDERING=1`	If this is present, as shown, then the AML rendering rules will be used. If this is not present or set to `0` the AML rendering rules will **not** be used. This setting must be present along with the '**ENABLES52RENDERING=1**' setting for the rendering rules for AML to be used. Use in conjunction with the **S57CONFIGFILE** setting to ensure that rendering and features match.
`ExtractAttachments=1`	Required

DSExtractAttachments=1	Required
FILTER_METADATA_MAP=1	Required
FILTER_METADATA_ROOT=1	Required
S57INTENDEDUSAGE=-1	Required
S57INTENDEDUSAGE_DATASET=-1	Required
S57CONFIGFILE=IHOS57Catalogue_v311.xml	S-57 3.1.1 Feature/Attribute Catalogue file to use. For AML this should be set to use one of the AML Catalogue files.
S52_DISABLE_COSPATIAL_CALCULATIONS=1	If present disables most of the co-spatial analysis. Only use for a simple viewer.
S52_DISABLE_CENTRE_OF_GRAVITY_CALCULATIONS=1	If present disables the calculation of default weighted centre of gravity. It is advised that this is only used for debugging.
S52_SAVE_RENDITION=C:\temp\s52.rnd	This is used to create a new rendition for S-52. This should only be used if the S-52 Rendering tables have been changed. By default this entry should not be present. Once a new rendition file has been created this entry should be removed from the filter configuration file.

5.3.4. S-57 Attribute and Object Configuration

The file `IHOS57Catalogue_v311.xml` contains the S-57 object code and attribute code mapping to ASCII description and ASCII code.

The Attribute and Feature records contained in the file are described below.

You would not normally need to modify this file. If you modify this file you need to review the S-52 configuration files (see 5.3) and S-52 rendering tables (see 5.3.5) and update as necessary.

The file contains the attributes and features for S-57 3.1.1.

Attribute Record	
`<attribute ID="4">`	String version of ID value (internal ID)
`<label>boyshp</label>`	S-57 label (acronym – matches the attribute acronym in the S-52 rendering tables)
`<name>Buoy shape</name>`	Description
`<ID>4</ID>`	S-57 attribute ID value as numeric
`<acronym />`	
`<description />`	
`<attributeType>E</attributeType>`	Attribute type see below
`</attribute>`	This will appear at the end of the attribute and any associated attribute data.

Attribute Types:

- E : Enumeration
- F : Float
- L : List
- A : Coded string
- S : Type string

Enumeration: Each enumeration value (key) needs to be defined.

Example

```
<enum>

<key>503</key>

<shortDescription>practice and/or exercise purposes</shortDescription>

<fullDescription>practice and/or exercise purposes:  used for military
practice and/or exercise purposes only</fullDescription>

</enum>
```

List: Each value (key) needs to be defined.

Example
<enum> <key>1</key> <shortDescription>fixed bridge</shortDescription> <fullDescription>fixed bridge: A bridge having permanent horizontal and vertical alignment. (McGraw-Hill Dictionary of Scientific and Technical Terms, 3rd Edition, 1984)</fullDescription> </enum> <enum>

Feature Record	
<feature ID="111”>	Feature ID as a string (internal ID)
<label>RSCSTA</label>	S-57 feature name (acronym – matches the feature acronym in the S-52 rendering tables).
<featureClass>Rescue station</featureClass>	Description
<ID>111</ID>	S-57 Feature ID as a numeric.
<acronym />	
<description />	
<featureType />	
<attributesByRef>	
</attributesByRef>	
</feature>	

5.3.5. S-52 / AML Rendering Tables

The S52 SDK uses the standard S-52 rendering tables to define the rendering rules. The following files are provided:

- "table 1 paper chart points.txt"
- "table 2 simplified points.txt"
- "table 3 line symbolisation.txt"
- "table 4 area symbolisation.txt"
- "table 5 area plain boundaries.txt"

The AML version of these tables can be found in a sub-directory called AML. The AML Dynamic Renderer expects to find its versions of these files in this sub-directory.

Note: We have corrected some of the rules where they do not comply with the S-52 specification.

If you modify the rules in any of the above tables you must re-import all your S-57/S-63 data as MapLink Pro stores information about the rules on the geometry to improve the drawing performance.

The addition of new Rendering Procedures is not supported. This would require an update to the S-52 SDK. Please contact your sales representative if you require support for new Rendering Procedures.

The structure and contents of the files are defined in the S-52 document (reference 6, Presentation Library 3.4)

Experience has shown that the tables defined in the standard may need tweaking to actually meet the standard.

If you find an error in the tables please report this to support@envitia.com

The following files are using for testing:

- ASYMREFT.txt
- LSYSREFS.txt
- PSYMREFT.txt

5.3.6. Style Lookup files

The colour, fill style, line style and symbol names used in the S-52 Rendering tables require a mapping to standard MapLink Pro resource styles (tsllinetyles.dat, tslfillstyles.dat, tslsymbols.dat).

If you add new S-57 features you will need to update the files described in the following sections. You may also need to create new linestyles and fillstyles. We would ask you to share this with us if possible so that we can consider including the updates.

5.3.6.1. S52colour.dat

This file defines a mapping from the S-52 colour name to a MapLink colour index value. The RGB values of the colours are defined in the palette files (see paragraph 1.1.1).

File Contents Example	Description
S52CI 100 S52 Colour Index 110000 , 64 0 , NODTA , (no data) (= grey) 1 , CURSR , (cursor) (= orange)	File identifier and version number. Description MapLink Colour index start.This is the start of a reserved range for S-52 colours. Deliminator character. Number of records. Colour number, S-52 colour name, description

5.3.6.2. S52fillstyles.dat

This file defines a mapping from the S-52 Fillstyle name and colour to the MapLink Pro fillstyle index.

File Contents Example	Description
S52FSI 100 S52 Fill Style Index 0 , 25 444 , AIRARE02 , LANDF , Airport area 445 , DIAMOND1 , DEPCN , Depth less then safety contour 446 , DQUALA11 , CHGRD , 5m accuracy with full seafloor coverage	File identifier and version number. Description tslfillstyles.dat : Fillstyle base value Deliminator character. Number of records. Fillstyle number, S-52 fillstyle name, S-52 colour name, Description

5.3.6.3. S52linestyles.dat

This file defines a mapping from the S-52 Linestyle name and colour to the MapLink Pro fillstyle index.

File Contents Example	Description
S52LSI 101 S52 Line Style Index 0 , 78 18469 , 18800 , ACHARE51 , CHMGD , S52 Boundary of anchorage area 18470 , 18801 , ACHRES51 , CHMGD , S52 Boundary of prohibited/restricted anchorage area	File identifier and version number. Description tsllinestyles.dat : linestyle base value Deliminator character. Number of records. Outer linestyle number, inner linestyle number, S-52 linestyle name, S-52 colour name, Description

5.3.6.4. S52symbol.dat

This file defines a mapping from the S-52 Symbol name and colour to the MapLink Pro Symbol index.

Example	Description
S52SI 100 S52 Symbol Index 124000 , 559 0 , ACHARE02 , 503 , 402 , 206 , 263 1 , ACHARE51 , 1304 , 1229 , 629 , 779	File identifier and version number. Description tslsymbols.dat : Symbol style base value Deliminator character. Number of records. Symbol index, S-52 name, height, width, offsetX, offsetX

The height (SYVL), width (SYHL), offsetX (SYCL- SBXC) and offsetY (SYRW – SBXR) are extracted from the DAI files.

The information is extracted from the line:

```
SYMD   39CAIRNS01V007500075000468004770051000310
```

In the above example the name of the symbol is 'CAIRNS01'. Using the definitions below the relevant parts can be extracted.

```
SYNM A(8) name of the symbol
SYDF A(1) type of def: V=vector, R=raster
SYCL I(5) pivot point's column number; in [-9999,32767]
SYRW I(5) pivot point's row number; in [-9999,32767]
SYHL I(5) width of bounding box;    in [0,32767] for vector (doesn't include line width)
SYVL I(5) height of bounding box;   in [0,32767] for vector (doesn't include line width)
SBXC I(5) bounding box upper left column number; in [0,32767] for vector
SBXR I(5) bounding box upper left row number;    in [0,32767] for vector
```

Where:

- A is ASCII.
- I is integer.
- A(8) indicates 8 ASCII characters make up the field, I(5) indicates 5 numeric characters make up the field.

5.3.6.5. S52renderlevel.dat

MapLink Pro has the concept of render levels which is similar to the S-52 display level.

This file defines the earliest render level that a feature can be drawn at. It is based on the display level and if a rendering procedure is called on the lowest display level that the procedure can draw the feature at.

Example	Description
S52RL 100 S52 Procedure alternate render level ; 30 CLRLIN01 ; ; Clearing line (mariners' navigational object) DATCVR02 ; 3 ; Data coverage, scale boundaries, overscale (S-57) DEPARE02 ; ; Depth area colour fill and dredged area pattern fill (S-57) DEPCNT03 ; ; Depth contours, including safety contour (S-57)	Identifier, File Version Number Description Marker Number of entries S-57 feature Name ; Render Level ; Description

If the 'Render Level' field is left blank then the default rule render level is used. Render Levels are from 0 to 10.

Updating of this file correctly requires an understanding of the S-52 rendering procedures. In general if the S-57 feature is drawn using a simple rule (no rendering procedure called) then the render level can be left blank. If a rendering procedure is called then you need to set the render level to the lowest possible render level based on what the rendering procedure logic requires.

5.3.7. Palette Files

The following files are for Day, Dusk, Night time viewing of S-52 rendering.

- S52day.pal
- S52dusk.pal
- S52night.pal

The file format is the standard MapLink Pro palette file format (see the MapLink Pro Developers Guide).

6. FAQ

6.1. What is S-63?

S-63 is encrypted S-57 data

6.2. What does the S-63 SDK provide me with?

The S-63 SDK is a direct import data-layer. It is designed to load and manage updates to both S-63 and S-57 data using the workflows in the S-63 and S-64 documentation.

The S-63 SDK does not provide a complete solution as the IHO require the application to implement additional functionality that does not fit into an SDK, such as key management and data management.

6.3. Can the SDK load AML?

The S-63 SDK can load both ENC (3.1.1) and AML (1.0, 2.1, 3.0) data.

6.4. How do I display AML/ENC?

The complementary S-52 SDK supports the visualisation of ENC and AML for situational awareness (not for navigation).

There are certain aspects of S-52 that MapLink cannot implement at the SDK level that an application can implement.

The UKHO has stated that AML and ENC should not be mixed as such AML and ENC should be managed separately and use different S-52 Dynamic Renderers.

You can of course still use the MapLink rendering rather than the S-52 Dynamic Renderer.

6.5. Data Encryption

The IHO says that the data supplied should be maintained in an encrypted form and that the certificates have to be checked to make sure that the cell data has not expired.

The ingested data is passed to the data client encrypted (see storeIngestedCell 2.7.1.13). The application needs to provide a data store to store this information. There is no need to delete the data as it has been encrypted (see 2.7.1.13, 2.7.2, 2.8.2 and 2.8.4.4).