HEXAGON

White Paper

IMAGINE Developers' Toolkit

23 July 2019

Contents

Product Overview

IMAGINE Developers' Toolkit™ is a set of libraries and documentation for ERDAS IMAGINE® users who want to modify the commercial version of the software or develop entirely new applications to extend the capabilities of the software, to meet their specific project needs. The Toolkit includes a set of C/C++ language APIs that are intended for the experienced C/C++programmer. The IMAGINE Developers' Toolkit is available through the Hexagon Geospatial Developer Network (HGDN) Subscription program. The HGDN community provides an online, interactive tool used to present the most up-to-date documentation and examples for the Developers' Toolkit API.

ERDAS IMAGINE is a broad suite of software products designed to address the needs of a wide audience. Although the user interface is designed to make workflows easy for a variety of skill and experience levels, some organizations need to customize the software in order to streamline a particular production workflow. The basic customization capabilities of ERDAS IMAGINE are particularly well suited for modifying the easy-to-use graphical interface, while the IMAGINE Developers' Toolkit is designed primarily for users needing to tailor the software beyond this level.

What type of customization/modifications do you need to make? This is the most important question in determining which ERDAS IMAGINE tools you will need to use. ERDAS IMAGINE offers two broad categories of customization capabilities, each achievable with different sets of tools:

- Customizing
- Extending

Learn about specific types of modifications and new applications that you can develop using the IMAGINE Developers' Toolkit.

Customizing Capabilities

ERDAS IMAGINE provides a range of customization capabilities — from simple preferences to complete changes to the user interface. These customizations help organizations that want to change the language of the interface, for example, or need to simplify it for dedicated applications, such as photo-interpretation.

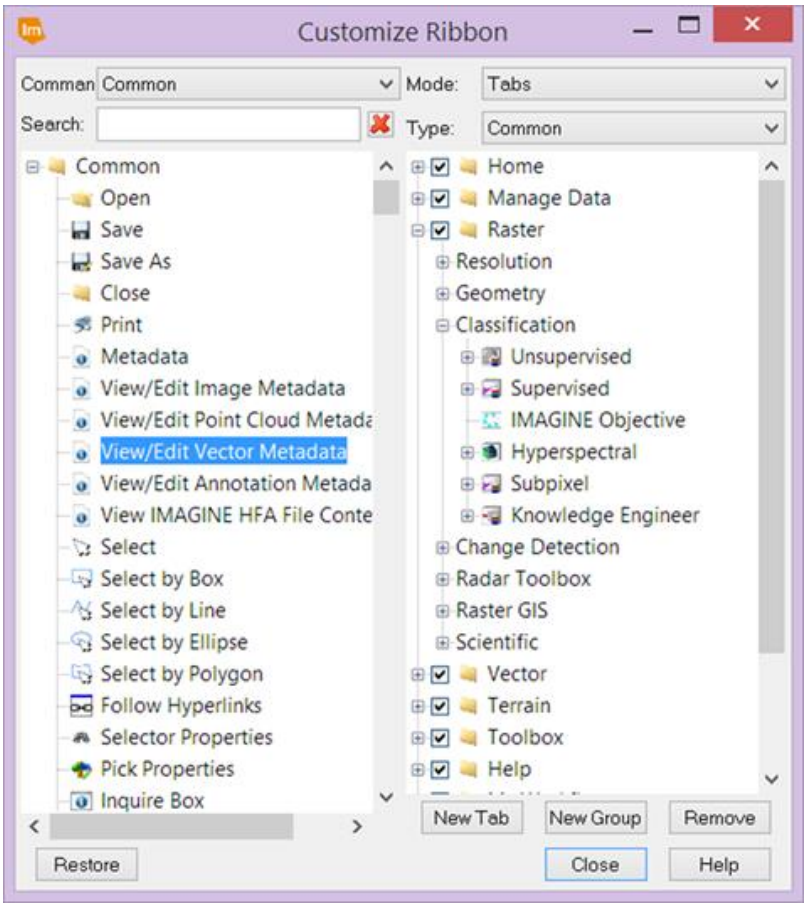
Preferences

The simplest and most straightforward means of customizing is provided by the Preference Editor. The Preference Editor manages a database of preference values that are used throughout the product to modify its behavior. A preference is a value that defines a user's choice for some optional aspect of the software. For example, ERDAS IMAGINE can open "bubble help" whenever the cursor rests on top of a user interface element for some period of time. By setting preferences, the user may define the length of time before bubble help opens or choose to never open it.

Preferences are saved through the user interface in a local sense. A local change affects only the current user by saving the settings in his/her home directory. A global change can be affected by moving a customized preference file to the ERDAS IMAGINE installation directory, thereby affecting all users (unless overridden by a local change). The Preference Editor is provided with every ERDAS IMAGINE license.

Ribbon Customization

Beginning with the release of ERDAS IMAGINE 2010, the main user interface is presented using the now-standard Microsoft ribbon style format. Ribbons themselves offer a high degree of customizability through standard capabilities such as Quick Access Toolbars and the Customize Ribbon dialog for re-organizing the user interface.



EML Script Changes

Many dialogs invoked from the ERDAS IMAGINE ribbon interface were developed with the ERDAS Macro Language (EML), which comes with every license of the software. This language is a scripting language that can be used to define the structure and content of the user interface, as well as provide some fundamental procedural scripting capabilities. Each script is interpreted at application start-up and converted to instructions for the native windowing system. These files are included in a virtual scripts directory when the associated application is built. Instead of being “compiled”, they are just copied to the IMAGINE_HOME directory for runtime. Each script is an ASCII file, which may be edited to change its contents. For example, the titles of all menus, buttons, and other user controls may be changed to another language. In addition, menu items may be added or deleted. In many cases (though not all), commands may be added to or deleted from existing dialogs.

Extending Capabilities

In addition to customizing the existing applications, some organizations may want to add new capabilities to the software, such as adding to the existing scripts, writing new Spatial Models, creating new Spatial Model Operators and embedding Geoprocessing into other applications using the Spatial Modeler SDK (SMSDK), or developing entirely new applications using the IMAGINE Developers’ Toolkit.

EML Script Additions

Because EML provides a procedural-scripting environment in addition to the user interface definition, it is possible to create new capabilities by combining existing ERDAS IMAGINE commands under new menu items or new buttons.

SML Additions

New applications can be built using Spatial Modeler Language (SML), which is a component of both IMAGINE Advantage® and IMAGINE Professional® licenses. Many tools on the Raster tab in ERDAS IMAGINE (for example, Decorrelation Stretch) were built primarily from SML scripts with an EML interface. Once you have developed a new algorithm using the graphical environment, you can generate an SML script that can be combined with a custom-developed user interface. The new application may be plugged into the existing ERDAS IMAGINE menu structure so that it functions like any other part of the system.

Spatial Modeler Dialogs

A more modern way to extend the user interface with new spatial models is to use the Spatial Model Editor to develop an algorithm using Post Input operators to define required inputs to the model. These types of models use the Port Input definitions to auto-generate dialogs, which can be easily added to the Ribbon interface without the need to use an interface language such as EML.

Spatial Modeler SDK

Spatial Modeler SDK (SMSDK) is a C++ toolkit for building, modifying, and running workflows on geospatial data. SMSDK can be used to build complex algorithms or simply to run routine tasks. Based on the Spatial Modeler tool initially developed for ERDAS IMAGINE 2013, SMSDK is extensible via a plugin mechanism where objects, such as operators, data types, and configuration dialogs, are discovered at runtime by demand-loading all DLLs found in a search path and identifying the Spatial Modeler objects implemented in those DLLs. Build add-ons to ERDAS IMAGINE or build and run workflows within your own product (the latter requiring additional signed agreement with Hexagon's Geospatial division) using the Spatial Modeler SDK.

Custom Applications

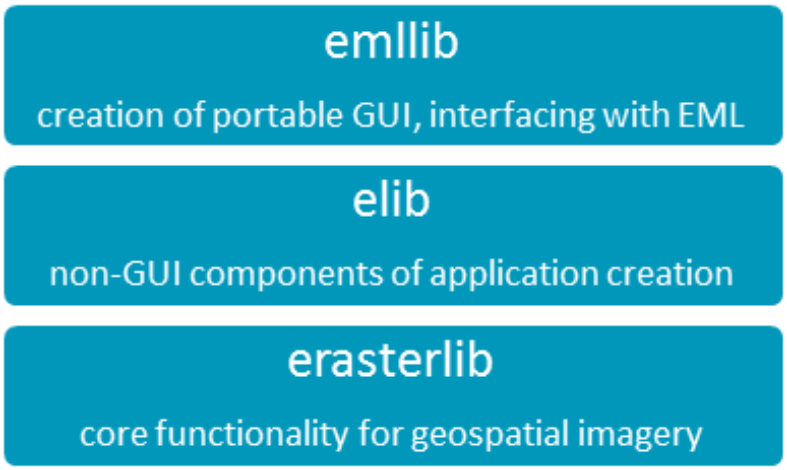
Extending ERDAS IMAGINE goes beyond changing interfaces and adding new image processing applications, and consequently generally requires the use of the IMAGINE Developers' Toolkit. One of the keys to making the system useful is the ability to use any data available, or to integrate entirely new types of image processing.

Custom DLLS

A major feature of the ERDAS IMAGINE architecture is the use of DLLs to provide for custom extensions. A DLL is a Dynamically Loadable Library, which is a piece of code to be located and used at run time by an application. ERDAS IMAGINE uses this to create plug-in sockets in a variety of areas, such as imagery access, coordinate conversion, coordinate projection, and font access. Without modification to existing applications, a Raster Format DLL may be written and added to the system, which allows all ERDAS IMAGINE applications to access data stored in previously unsupported file formats directly and without file conversion.

Product Definition and Requirements

IMAGINE Developers' Toolkit is the collection of libraries and associated headers and documentation used by Hexagon's Geospatial division development staff to create ERDAS IMAGINE products. The libraries are organized into several large functional groups.



- The core is the base library, called erasterlib (ERDAS Raster Library), which contains all the core functionality needed to create / read / write geospatial imagery.
- Layered on top of this is elib (ERDAS Library), which contains all the non-GUI components of application creation, such as session management, annotation I/O, and various utility packages.
- Layered on top of the elib library is the user interface library, called emllib (ERDAS Macro Language Library), which contains all the functions for creating a portable GUI and interfacing with EML.

Hexagon Geospatial Developers Network (HGDN) is an interactive online network of programmers and developers. All content in this environment is shared by HGDN Subscribers and Hexagon's Geospatial division employees, which makes this a true network of programmers. This method for presenting the IMAGINE Developers' Toolkit documentation to every programmer using it ensures that all content in the network is as up-to-date as possible. There are several parts to this web environment:

1. The most important feature is the Toolkit Library, which contains all supported Toolkit packages, data types, and functions. All entries are categorized for easy searching and every item is guaranteed to be complete.
2. Developers' FAQ section is a list of commonly asked questions with answers provided by Hexagon's Geospatial division employees.
3. Each item in the Toolkit Library Database is dynamically linked to every example. This means that when selecting a particular function or data type, a list of example program that contain that item will be returned. There is also a categorical search for Visual C/C++, EML, SML, and spatial models.
4. Each user will be able to download the complete project from the Examples Database. There is also a categorical search in place to help when downloading several examples at a time.
5. We have developed a peer-to-peer coding forum so that customers are able to ask questions on any developer-related topic.

Users of the IMAGINE Developers' Toolkit must be experienced C/C++ programmers. The IMAGINE Developers' Toolkit is available as part of an HGDN Subscription.

Problem Solving Examples

Example 1: Create a New Data Importer

With the number and variety of imagery formats always on the rise, the data importer and exporter is the most common type of application added to ERDAS IMAGINE. The IMAGINE Developers' Toolkit contains functions that provide I/O for the ERDAS IMAGINE file format, as well as functions that make it easier to create an importer / exporter that operates like those that are included with ERDAS IMAGINE.

An airborne scanner manufacturer might need to make their data easily accessible to potential customers who use ERDAS IMAGINE by providing an importer into an ERDAS IMAGINE compatible file format. As the data is read it could be written to an ERDAS IMAGINE .img file using the Image I/O functions in elib. The geographic information could also be used to create input to ERDAS IMAGINE rectification tools to provide a semi-automated process of geocorrection.

Example 2: Create a Plug-in for a Unique Sensor Model

While developing a system for photointerpretation of satellite data, a military contractor had a sensor that did not use a geometry similar to any of the typical frame cameras or push broom sensors, and they wanted to use the imagery without resampling in the ERDAS IMAGINE application. To solve this problem, they created a plug-in sensor model for use with ERDAS IMAGINE's geometric modeling. The sensor model describes the geometry of their sensor (for example, it tells how to convert a pixel coordinate row and column into a ground coordinate) and how to perform the inverse operation. Implemented as an ERDAS IMAGINE Geometric Model DLL, ground base coordinates could be read from the image using ERDAS IMAGINE with this model, as well as orthocorrect the image using existing ERDAS IMAGINE tools.

Example 3: Enhanced Atmospheric Correction

Dr. Rudolph Richter of DLR in Germany is a leading specialist in the study of atmospheric correction. His algorithms, known as ATCOR, have been widely used to remove the effects of the atmosphere from satellite images. The ERDAS IMAGINE distributor in Germany, GEOSYSTEMS GmbH, collaborated with Dr. Richter and used the IMAGINE Developers' Toolkit to create the ATCOR Workflow for IMAGINE module for use with ERDAS IMAGINE. Installed add-on modules in ERDAS IMAGINE are indicated by a new icon object in a tab in the ribbon. When launched, ATCOR Workflow for IMAGINE opens various tools for fine-tuning the atmospheric model before correction takes place, then displays the output in ERDAS IMAGINE.

More recently the SMSDK has been used to convert the ATCOR capabilities into operators for use in larger workflows built using the Spatial Model Editor.

Example 4: Develop a Photointerpreter Workstation

One of the earliest applications of the IMAGINE Developers' Toolkit was the development of the Target Materials Workstation, or TMWS. TMWS was developed to provide an easy-to-use workstation for military photointerpreters.

The interpreter's task is to look at an image, mark up any important features, and then print the results. They must be able to sharpen, contrast stretch, and perhaps rotate and scale the image. Then text, lines, and other well-defined graphics are placed on the image using annotation tools. This marked-up image is then inserted into one of several standard map templates and printed for dissemination. In addition, there is usually some central control of tasking.

TMWS was made by customizing ERDAS IMAGINE. The menus and tool palettes were simplified and tailored for the task at hand. In addition, special top-level icons were created to lead the operator through the process. Coordination of the workflow and data management was achieved by working with an already existing central SAP (formerly Sybase) database. This was done by using SAP (formerly Sybase) libraries in

conjunction with the IMAGINE Developers' Toolkit to create a module that could obtain information about the images to be analyzed, and then record the name and location of the final products.

Key Features

- ERDAS IMAGINE object manipulation routines
- Application environment routines
- Low-level file I/O and system access routines
- Abstract object manipulation routines
- Unicode storage classes
- EML GUI access routines extensive context-sensitive online Help
- Peer-to-peer forum available through the Hexagon Geospatial Developers Network
- Example programs
- IMAGINE Photogrammetry block file creation routines
- 2D visualization routines

Technical Specifications

Architecture

Routines

IMAGINE Developers' Toolkit is organized as various packages that create and support a type of data object. The package usually defines a few data types, which are then manipulated by the functions. These packages are described below in groups of routines.

DLL Classes

Some of the packages are built on a DLL mechanism that allows that package to be extended dynamically. For example, the eimg routines for reading and writing imagery from files are based on the Raster Formats DLL Class. Any application written using the eimg package will be able to read data from any of the supported file formats. The packages based upon an extensible DLL are indicated by the name of the DLL class in brackets, for example [Raster Formats DLL].

ERDAS IMAGINE Object Manipulation Routines

Package	Library	Description
Annotation	eant	Provides functions for creating, reading, writing, rendering, and editing ERDAS IMAGINE annotation elements and their styles. Annotation elements consist of polylines, polygons, ellipses, elliptical arcs, points, text, and groups.
Area of Interest	eaoi	Provides basic tools for area of interest analysis. It includes AOI structure operations (create, delete, copy, so forth), AOI input and output operations, and AOI rendering operations.
Color Library	eclb	Provides support for color libraries, RGB / IHS conversions, and level slicing.
Unit Conversion	ecvt	Provides routines for converting between units of distance, area, time, volume, angle, and so forth. The units supported are defined in a database (units.dat), which defines the relation between a standard unit and others. For example, the standard distance is in meters, all other distance units are defined as multiples of meters.

Descriptor Table Access	edsc	Provides basic tools for manipulating descriptor tables. A descriptor table is the part of an ERDAS IMAGINE .img file that stores ancillary information about a particular data value or set of data values in a raster layer. The information for each data value, or set of data values, might include a histogram value, color table look-up values, a class name, total area, and so forth. With the edsc functions, the programmer can easily create in-memory data structures for descriptor tables and easily pass descriptor table information between memory and disk without having to deal with the lower level ehfa, emif, and efio packages. Using the functions, the programmer can: • Create, open, and close a descriptor table • Create, copy, and delete a bin function – the part of the descriptor table that describes how data values in the raster layer are mapped to rows in the descriptor table • Create, open, read, write, and close a descriptor table column – the ancillary information
Digital Signal Processing	edsp	Provides functions that perform 1- and 2-dimensional forward and inverse Fast Fourier Transformations (FFTs) on complex or real data. Most functions in this routine have been specially designed for efficient transformation of real valued sequences and arrays. These sequences have certain symmetries that reduce both storage requirements and the number of computations. The length of each sequence must be a power of two.
Graphics Support	eevg	Provides functions for rendering vector-type data (annotation, vectors) to output devices. It handles the rendering of line styles, polygon fill styles, and smooth outline scalable text.
Floating-Point Graphics Arithmetic	efga	Provides routines for performing commonly used vector, matrix, point, and polynomial (including affine) transformation functions.
File Node Parse	efnp	Breaks up strings containing file names and ehfa node names and ranges of ehfa node names into their component parts. For example, use it to separate the path “D:/data/cherokee-hickory-flats.img(:Layer_1)” into the components: • D:/data/ • cherokee-hickory-flats.img • (:Layer_1)
General Data Arithmetic	egda	Provides basic tools to perform operations on various data objects based on the Egda_BaseData structure. Objects based on this data structure are prevalent in the IMAGINE Developers’ Toolkit libraries such as the eimg_PixelRect. The derivatives of this data structure include layer, stack, matrix, table, vector, window, and scalar objects. In addition to providing functions that create, copy, modify, and destroy these data types, the routines also handle operations such as data type conversion, standard point functions (for example, sin, cos, so forth), GIS point functions (for example, diversity or density), single argument operations (for example, not or sign), multiple argument operations (for example, multiply or divide), and neighborhood functions (for example, majority or convolve).
Block File Creation	epho	Supports the easy creation of IMAGINE Photogrammetry block structures. It is most likely to be used by applications that import other data formats into the block file format. The block structure stores all the information that is associated with a block of images. Some this data is created and maintained by the IMAGINE Photogrammetry libraries and some of the data is created and maintained by the BlockModelInterfaces DLLs.
Feature Space	efsp	Contains all the feature space functions, including: • Read and write of feature space nodes as children in .img files • Read and write lists of feature space nodes for communication with the transformation process • Convert the image data into feature space plots • Map auxiliary files (thematic) into the feature space
Hierarchical File Access	ehfa	Provides an implementation of the ERDAS IMAGINE EHFA file format. This format maintains an object-oriented representation of data in a disk file by using a tree structure. All non-ASCII files used by ERDAS IMAGINE are in this format (for example, .img, .ovr).

		Using the functions in this routine, the programmer can identify, create, copy, open and close EHFA files; and locate, read, and write objects within an EHFA file. The standard ERDAS IMAGINE data objects (layers, map information, descriptor tables, and so forth) can be accessed through higher level packages (eimg, edsc, and so forth), so these functions are normally used to manipulate application-specific data objects within EHFA files.
Histogram and Lookup Table	ehst	Provides various functions on histograms and lookup tables, primarily in support of the histogram tools and the contrast adjustment in the Viewer. Histogram, LUT, and breakpoint structures can be created, converted, and edited using these routines.
Image File Access	eimg	Provides tools to access and manipulate data in image files supported by the RasterFormats DLL Class, including ERDAS IMAGINE .img files. The package includes functions to read and write objects commonly found in geospatial raster datasets such as map information, projection parameters, statistics, histograms, covariance matrices, descriptor tables, class names, and color values. The package also provides high-level functions for processing the raster data within the dataset including functions to create, read, and write raster layers; associate windows, and areas of interest with a raster layer; and apply neighborhood functions to raster layers. An application may modify the actual data values in a raster layer using this package and its constructs.
Image Resampling [ResampleMethods DLL]	eirf	Provides tools to resample imagery that is being geometrically transformed (see Transform). Three resampling methods are supported: • Nearest neighbor • Bilinear interpolation • Cubic convolution Other types of resampling can be supported through the ResampleMethods DLL Class mechanism.
Kernel Library	eklb	Provides for reading and writing convolution filter kernels from a kernel library file.
Map Printing	emap	Provides routines for generating and reading map files (.map) and for creating, reading, and writing all types of files associated with hardcopy map output.
Pixel Management	epxm	Provides functions for creating and copying rectangular arrays of pixels of any of the thirteen data types supported by ERDAS IMAGINE. The pixel rectangles are compatible with the egda functions.
Raster GIS Analysis	erga	Provides tools to analyze raster data including tools for point analysis, neighborhood analysis, regional analysis, and feature extraction. This high-level package operates on top of the eimg package, unlike the egda, efga, and eirf packages that are not strictly dependent on any eimg constructs.
Seed / Region Growing	esed	Performs the region growing functions. Functions exist at different levels to do the raster region grow, convert the raster region to a polygon, and generate an annotation layer of the region polygon.
Signature Management	esig esig_io	Signature packages contain all the signature support functions: • esig provides functions to create signatures, set and get signature attributes, copy signatures, manage lists of signatures, merge signatures, and so forth. • esig_io contains all the file I/O functions for signatures and signature lists.
Statistics	esta	Provides tools for: • Manipulating (that is, reading and writing to/from files, creating and copying in-memory data structures, computing, and so forth) statistics (mean, standard deviation, and so forth) • Histograms • Covariance matrices • Color tables • Class names This middle level package is utilized by the eimg package but relies on the edsc, ehfa, and emif packages.

Vector	evec	Provides tools for: • Creating and deleting structures related to vector layers • Opening and closing vector layers • Reading features and their attributes • Getting descriptive information about the layer • Measuring distances and lengths • Modifying vertices of arcs (splining, generalizing, and densifying) • Determining the spatial relationship between points, lines, and polygons
Transform [GeometricModels DLL]	exfr	Provides a generalized set of tools for creating and managing coordinate transforms, regardless of the type. Each transform may be a combination of any number of the basic transforms. The package provides functions for mapping arrays of points through a transform, inverting transforms, and composing transforms. The package is also constructed so that the programmer can create new types of transforms. This package is extensible through the use of the Geometric Model DLLs.

Application Environment Routines

Package	Library	Description
Argument Parsing	earg	Provides a convenient means of creating applications that are controlled by command line options. Parsing the command line and looking for options is a tedious task and without a set of standard tools, the resulting command line syntax will vary from program to program.
Error Logging and Reporting	eerr	Provides functions to create, delete, and print error reports. Nearly every function in the libraries of the IMAGINE Developers' Toolkit requires an argument that represents a place for the function to pass back an error report to the calling function. Using the functions and macros defined in this package, the programmer can work within the error reporting system of the IMAGINE Developers' Toolkit.
Master Initialization	eint	Initialize the IMAGINE Developers' Toolkit. This step is required in an application before the other IMAGINE Developers' Toolkit functions can be used.
Progress Meter Support	emet	An ERDAS IMAGINE application can report its progress in an extensible way. A user can track the progress of an application by placing calls to the emet package at critical points in a process.
Magnetic Tape Access	emta	Provides a client-side API to the ERDAS IMAGINE tape server. Using the API to the tape server, an application can open, close, and manipulate (for example, rewind, skip files) tape devices, as well as read and write data to those devices. The type of device (8 mm, track, and so forth) and its location on the network (local, remote) are transparent to the API, and therefore, the application.
Preference Database	epmg	Provides a programmer with an interface to the Preference Database. Provides functions to access, set, and change the preferences for different categories at global and user levels. Also provides callback mechanisms to notify the clients when a preference value is changed.
Session Manager	esmg	Provides an interface to the ERDAS IMAGINE Session Manager. Primary functions are: • Connect to the Session Manager • Indicate job state • Indicate job progress • Log warning/status messages in the Session Log • Indicate the end of a job

Viewer	evue	Contains a set of functions for interacting with the ERDAS IMAGINE Viewer. Using the functions, the application can: • Query the contents of specified viewers • Load image, annotation, vector, or AOI layers into specified viewers • Add callbacks on images in the viewer • Create graphics representing lines or areas at specified locations • Interactively select areas or lines • Display map compositions • Set image extents • Create and destroy viewers • Warp images • Create screen links • Set convolution kernels

Low-level File I / O and System Access Routines

Package	Library	Description
Low Level File	efio	Provides functions for creating, reading, writing, locating, and finding information about files and directories. Many of these functions are simply wrappers for existing POSIX and ANSI C functions. The wrapper allows for expansion of environment variables in file names and directory paths, as well as the translation of the standard 'errno' variable into an error report consistent with the eerr package. Other functions allow temporary file names to be generated and for a directory path to be searched while attempting to locate a particular file. The package also allows efficient I/O that is portable across platforms.
Machine Independent Format	emif	Allows data to be read and written to disk in a format that is sharable across compilers, operating systems, and machine architectures. Using the package, a programmer can create and destroy dictionaries of object designs, create and destroy object designs in those data dictionaries, and read and write data to disk based on an object design. Standard ERDAS IMAGINE data objects (layers, map information, descriptor tables, and so forth) can be accessed through higher-level packages (eimg, edsc, and so forth), so these functions are normally used to manipulate application-specific data objects within EHFA files.
Timer	etim	Provides routines for timing code execution.

Abstract Object Manipulation Routines

Package	Library	Description
Binary Search Tree	ebst	Provides a complete set of tools for creating and maintaining sorted binary search trees. Modeled after the qsort/bsearch functions in the standard C libraries, the ebst package works with any type of programmer-supplied data for which the programmer provides a compare function.
Dynamic List Manager	edlm	Provides a means to create and retrieve data from lists and stacks of arbitrary size. The lists, or stacks, may contain any type of object. Lists and stacks are useful in situations in which you need to build a set of data by sequentially adding objects, but you do not know beforehand how many objects will be in the set.
Linked List	ellm	Provides a common set of functions used to create and manage linked lists. The lists may contain any type of data.
Miscellaneous	emsc	Defines macros and functions that deal with memory management or are otherwise not specific to any one routine in the IMAGINE Developers' Toolkit, for example, EMSC_MIN, EMSC_MAX, EMSC_CURRENT_VER.

Selection	esel	Provides a set of functions for implementing queries based on a simple query language. A query is then converted to an internal form, which is then used by the query evaluator. The result of the query is stored in a bitmapped table of Boolean results. There is a complete set of functions for managing the resulting Boolean tables.
String Manipulation	estr	Provides functions for 6 categories of character string manipulation: • General string manipulation (compare, change, duplicate, analyze character strings) • File name string parsing • String list manipulation • String expression evaluation • Numeric formatting to a character string • Miscellaneous string functions (for example, converting errno to a string description of the error)
Symbol Table	esym	Provides a set of functions used to create and manage symbol tables. Each holds one or more symbols that relate a string with some type of user data. The package provides functions for adding symbols to the table and for locating symbols within a given table.

EML GUI Access Routines

Package	Library	Description
EML	eeml	Provides a programmer with an interface to all the ERDAS Macro Language (EML) API functions. Application programs that present a GUI through an EML script should use the functions in this package. Functions include: • Create a GUI for the application program using the EML scripts as templates • Display and undisplay the dialog frames defined in the EML script • Add and remove callback functions on the GUI parts defined in the EML script • Manipulate GUI objects (get values, set values, and so forth)
Button	eeml_button	Provides functions specific to the EML button part. Functions include: • Get and set colors for color buttons • Arm and disarm and toggle buttons
Canvas	eeml_canvas	Provides a system independent interface to on-screen and off-screen graphics primitives. Many of the primitives are specifically designed to work with the structures from the eimg package, making it easier to read ERDAS IMAGINE files and display the data to the screen. These primitives can be used to build graphic applications such as the ERDAS IMAGINE Viewer. Using this package in conjunction with the rest of the EML package results in an application that is easily ported between the various operating systems and window systems that ERDAS IMAGINE supports.
Dialog	eeml_dialog	Contains functions that provide consistent interaction with the user for common user-interface operations such as asking the user for confirmation, displaying a warning message, displaying a status message.
Frame	eeml_frame	Provides functions specific to EML dialog frames. Functions include: • Set and get geometry to and from displayed and undisplayed frames • Display the frame on a different screen (dual-headed systems) • Set the title string, display the status message, show/hide the menu bar, show/hide the status bar, and so forth, access frame menus, and so forth.
Generic Part	eeml_generic	Provides functions specific to the EML generic part. The generic part is designed as a space holder in an EML dialog. It typically reserves space in the dialog for a user interface part that cannot be specified in EML, such as a CellArray or a canvas.

		Aspects of the user interface that cannot be handled in EML must be handled programmatically. Through the eeml_generic portion of the eeml package, functions are made available to the programmer that provide a consistent and portable way to display a busy cursor over a generic part while that part is being updated.
Menu	eeml_Menu	Provides the functions to create and interact with popup and pull-down menu systems.
Popup List	eeml_popuplist	Provides functions specific to the EML popup list part. Functions include: • Replace the old list with a new list • Replace the old picture list with a new picture list
Projection Editor	eeml_prjeditor	Provides a function to control an interactive Projection Editor that can be used by application programs to get projection parameters from the user.
Scrolled List	eeml_scrollist	Contains functions specific to the EML scroll list part. The functions allow application programs to manipulate the EML scroll list, for example, add items, replace items, and so forth.
Tablet Digitizer	eeml_tablet	Provides functions to access the Tablet Digitizer driver via an interactive GUI. Functions include: • Open and connect to the tablet digitizer • Add callbacks on the digitizer • Close and disconnect from the tablet digitizer
Text	eeml_text	Contains functions specific to the EML text part. Functions allow application programs to: • Insert text • Overwrite, replace, append text • Turn on, turn off the highlight attribute of the text • Get and set the cursor position on the text • Cut, copy, paste text • Set, get, clear the primary selection on the text
CellArray	eeui_CA	Provides a single user interface for dealing with the management of tabular data. The programmer provides the description of the columns (string, double, color, and so forth) and provides the functions to get and put the data. The CellArray package manages the navigation and display of any amount of data. Built-in functions allow search, sorting, and report generation.
Chart	eeml_Chart	Provides a graphical part that can be used to plot data in several modes. The chart is a tool that can plot tabular data as line graphs on a grid with an associated legend. This is the basis of the various profile tools in ERDAS IMAGINE.
Colorwheel	eeui_CW	Provides an HSI color part that can be used for color selection. A slider controls the intensity, with hue and saturation mapped to the angle around a circle and the distance from its center. To select colors, the user can drag a single point within the color wheel. The application is notified of color changes through a callback mechanism.
Histogram	eeui_Histogram	Provides an interactive tool for displaying and manipulating histogram data. User may zoom in to get more detail on any area of the histogram. In addition, the routine provides functions for controlling the contents of the plot as well as printing the histogram to a PostScript file.

2D Visualization Routines

Package	Library	Description
Viewer	edis	Provides a basic set of tools for embedding one or more viewers into an ERDAS IMAGINE application. A viewer can display various types of 2D data including vectors, annotation, and images. These routines also provide a means of extending the types of data that can be displayed within the viewer via the ViewLayer class.
ViewWindow	Edis_ViewWindow	Manages the display area of the viewer. When a viewer is created, a handle to the Edis_ViewWindow is returned and can be used with subsequent calls to the Edis_ViewWindow API. Use this API for toggling scroll bars, adding ViewLayers, scrolling imagery, and so forth.
ViewLayer	Edis_Viewlayer	Manipulating ViewLayer objects within a ViewWindow. Use for closing ViewLayers, changing band combinations, setting zoom levels, and so forth. This API can be used with existing ViewLayer types such as the TrueColor, PseudoColor, GrayScale, Annotation, Vector, and AOI layers.
ViewLayer class	Edis_ViewLayer	Defines the mechanism used for creating new types of ViewLayers that can be displayed within a ViewWindow. This becomes necessary when the ViewLayer types that are provided with the toolkit do not provide features required by the application.
Graphics Context **Graphics Target**		Both API provide ViewLayers, the mechanism needed to render them into the ViewWindow. When implementing new ViewLayer types using the Edis_ViewLayer class it is necessary to use these functions. Graphics Context is provided for setting attributes such as color, size, and so forth. Graphics Target is provided for doing the actual drawing.
Selector	Edis_SelNew	Provides an interface for creating graphical representations of objects that are used by the user for editing elements. The selectors do not provide a very robust set of display styles because they are typically used for displaying fast graphical elements including polylines, polygons, points, icons, and rectangles.
Selector class		Provides a mechanism for new selector types to be created when the existing selectors do not provide the features required by the application.
Picks		Provides tools used when the user is responsible for creating an element interactively using the mouse. Picks have been used for creating annotation elements, AOI elements, selecting viewers, and so forth. The API returns the coordinates selected by the user to the caller.
Raster Edit		Provides a few routines for modifying properties of raster images including contrast and convolutions kernels. These functions can also be used for modifying the pixel values in a TrueColor or GrayScale image layer.

About Hexagon

Hexagon is a global leader in sensor, software and autonomous solutions. We are putting data to work to boost efficiency, productivity, and quality across industrial, manufacturing, infrastructure, safety, and mobility applications.

Our technologies are shaping urban and production ecosystems to become increasingly connected and autonomous — ensuring a scalable, sustainable future.

Hexagon's Geospatial division creates solutions that deliver a 5D smart digital reality with insight into what was, what is, what could be, what should be, and ultimately, what will be.

Hexagon (Nasdaq Stockholm: HEXA B) has approximately 20,000 employees in 50 countries and net sales of approximately 4.3bn USD. Learn more at hexagon.com and follow us @HexagonAB.