REUSE MANUAL

GEOTRANS ENGINE

10xxxxxx.1

Implementation

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

1.1 PURPOSE OF THE REUSE MANUAL

This document describes the characteristics of the GEOTRANS ENGINE reusable software component and provides instructions on its installation and operation. The manual is a self-contained reference for the software engineer intending to incorporate the component in another software system. This manual was written with the assumption that the user has a basic working knowledge of C and is familiar with fundamental C concepts and terminology.

1.2 PURPOSE OF THE REUSABLE SOFTWARE COMPONENT

The purpose of GEOTRANS ENGINE is to provide a reusable component that supports the following coordinate conversions:

- Conversions between geodetic, geocentric, and local cartesian coordinate systems,
- Conversions between geodetic coordinates (latitude and longitude in radians) and various types of map projection coordinates (easting and northing in linear units),
- Conversions between geodetic coordinates and Military Grid Reference System (MGRS) or World Geographic Reference System (GEOREF) grid coordinates,
- Transformations of geodetic or geocentric coordinates between different global or local horizontal datums,
- Transformations between ellipsoid heights and geoid (or local MSL) heights, and
- Conversions that combine two or more of the above operations.

1.3 GENERAL INFORMATION

1.3.1 POINT OF CONTACT

U.S. Army Topographic Engineering Center (USATEC)
Digital Concepts and Analysis Center (DCAC)
ATTN: CETEC-PD-DS (Rick Joy)
7701 Telegraph Road
Alexandria, VA 22315-3864
Dan Specht (703) 428 - 6761 Project Manager

1.3.2 CERTIFICATION LEVEL

This RSC has been certified at level 4. A level 4 component satisfies the criteria for reliability, testing, and documentation for the Army Reuse Center (ARC). The component comes with test materials and a Reuse Manual that aids in integrating the component into a software system.

1.3.3 LEGAL RESTRICTIONS

This Reusable Software Component (RSC) contains data with Unlimited Government Rights.

SECTION 2. INSTALLATION

The following is a list of the files that make up the GEOTRANS ENGINE component:

Source Code Files:

```
engine.c
```

Header Files :

```
engine.h
```

Data Files :

```
none
```

The compilation instructions for the GEOTRANS ENGINE component are as follows:

DOS Makefile (Uses Microsoft C):

```
cl /nologo /W3 /FR /G2 /DNDEBUG /Gs /Ox /AM /D_DOS /c engine.c
```

UNIX Makefile (Uses gcc compiler):

```
cc -g -O -ansi -Wall -c engine.c
```

The compilation order of the GEOTRANS ENGINE component relative to other components is unconstrained.

2.1 PARTIAL REUSE

The GEOTRANS ENGINE component does not allow for partial reuse.

2.2 MODIFICATIONS

The GEOTRANS ENGINE component does not permit modifications.

SECTION 3. ENVIRONMENT

This section provides details on the environment under which GEOTRANS ENGINE was developed, tested, and executed.

3.1 HARDWARE

3.1.1 DEVELOPMENT

The following is a list of hardware configurations under which GEOTRANS ENGINE was developed and tested.

- SUN SparcStation 20
- IBM compatible Pentium PC

3.1.2 TARGET

The following is a list of hardware configurations under which GEOTRANS ENGINE was executed.

- SUN SparcStation 20
- IBM compatible Pentium PC

3.2 SOFTWARE

3.2.1 OPERATING SYSTEM

The following is a list of operating systems under which GEOTRANS ENGINE was executed and tested.

- Solaris 2.5
- Windows 95

3.2.2 COMPILERS

The following is a list of compilers on which GEOTRANS ENGINE was compiled successfully.

- GCC version 2.8.1
- Microsoft Visual C++ version 6

3.3 ASSUMPTIONS AND PERFORMANCE LIMITATIONS

There are no hardware or environment constraints. There are no limitations.

This RSC is written in ANSI C.

SECTION 4. GLOBAL RSC ENVIRONMENT

4.1 TYPES

The following is a list of significant visible types declared globally in GEOTRANS ENGINE with their descriptions.

4.1.1 ENUMERATED TYPES

The following enumerated types are used by parameters that are passed to public functions in GEOTRANS ENGINE:

```
/* Input or Output */
typedef enum Input_Output
{
  Input = 0,
  Output = 1
} Input_or_Output;

/* File or Interactive mode */
typedef enum File_Interactive
{
  File = 0,
  Interactive = 1
} File_or_Interactive;

/* Coordinate Type Enumeration */
typedef enum Coordinate_Types
{
  Geodetic,
  GEOREF,
  Geocentric,
  Local_Cartesian,
  MGRS,
  UTM,
  UPS,
  Albers_Equal_Area_Conic,
  Bonne,
  Cassini,
  Cylindrical_Equal_Area,
  Eckert4,
  Eckert6,
  Equidistant_Cylindrical,
  Lambert_Conformal_Conic,
  Mercator,
  Miller_Cylindrical,
  Mollweide,
  Orthographic,
  Polyconic,
  Polar_Stereo,
  Sinusoidal,
  Transverse_Mercator,
  Transverse_Cylindrical_Equal_Area,
  Van_der_Grinten
} Coordinate_Type;
```

```
/* Precision Enumeration */
typedef enum Precisions
{
  Degree,
  Ten_Minutes,
  Minute,
  Ten_Seconds,
  Second,
  Tenth_of_Second,
  Hundredth_of_Second
} Precision;

/* Heights */
typedef enum Height_Types
{
  No_Height,
  Ellipsoid_Height,
  Geoid_or_MSL_Height
} Height_Type;
```

4.1.2 STRUCTURE TYPES

The following structure types are used by parameters that are passed to public functions in GEOTRANS ENGINE:

```
/* Geocentric Coordinate Tuple Definition */
typedef struct Geocentric_Tuple_Structure
{
  double  x;   /* meters */
  double  y;   /* meters */
  double  z;   /* meters */
} Geocentric_Tuple;

/* Geodetic Coordinate System Definition */
typedef struct Geodetic_Structure
{
  Height_Type  height_type;
} Geodetic_Parameters;

/* Geodetic Coordinate Tuple Definition */
typedef struct Geodetic_Tuple_Structure
{
  double  longitude;   /* radians */
  double  latitude;    /* radians */
  double  height;      /* meters */
} Geodetic_Tuple;

/* GEOREF Coordinate Tuple Definition */
typedef struct GEOREF_Tuple_Structure
{
  char    string[21];
} GEOREF_Tuple;

/* Albers Equal Area Conic Coordinate System Definition */
typedef struct Albers_Equal_Area_Conic_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  std_parallel_1;    /* radians */
```

```
  double  std_parallel_2;    /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Albers_Equal_Area_Conic_Parameters;

/* Albers Equal Area Conic Coordinate Tuple Definition */
typedef struct Albers_Equal_Area_Conic_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Albers_Equal_Area_Conic_Tuple;

/* Bonne Coordinate System Definition */
typedef struct Bonne_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Bonne_Parameters;

/* Bonne Coordinate Tuple Definition */
typedef struct Bonne_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Bonne_Tuple;

/* Cassini Coordinate System Definition */
typedef struct Cassini_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Cassini_Parameters;

/* Cassini Coordinate Tuple Definition */
typedef struct Cassini_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Cassini_Tuple;

/* Cylindrical Equal Area Coordinate System Definition */
typedef struct Cylindrical_Equal_Area_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Cylindrical_Equal_Area_Parameters;

/* Cylindrical Equal Area Coordinate Tuple Definition */
typedef struct Cylindrical_Equal_Area_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Cylindrical_Equal_Area_Tuple;

/* Eckert IV Coordinate System Definition */
typedef struct Eckert4_Structure
```

```
{
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Eckert4_Parameters;

/* Eckert IV Coordinate Tuple Definition */
typedef struct Eckert4_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Eckert4_Tuple;

/* Eckert VI Coordinate System Definition */
typedef struct Eckert6_Structure
{
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Eckert6_Parameters;

/* Eckert VI Coordinate Tuple Definition */
typedef struct Eckert6_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Eckert6_Tuple;

/* Equidistant Cylindrical Coordinate System Definition */
typedef struct Equidistant_Cylindrical_Structure
{
  double  std_parallel;   /* radians */
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Equidistant_Cylindrical_Parameters;

/* Equidistant Cylindrical Coordinate Tuple Definition */
typedef struct Equidistant_Cylindrical_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Equidistant_Cylindrical_Tuple;

/* Lambert Conformal Conic System Definition */
typedef struct Lambert_Conformal_Conic_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  std_parallel_1;  /* radians */
  double  std_parallel_2;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Lambert_Conformal_Conic_Parameters;

/* Lambert Conformal Conic Coordinate Tuple Definition */
typedef struct Lambert_Conformal_Conic_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Lambert_Conformal_Conic_Tuple;
```

```
/* Local Cartesian Coordinate System Definition */
typedef struct Local_Cartesian_Structure
{
  double  origin_latitude;  /* radians */
  double  origin_longitude; /* radians */
  double  origin_height;    /* meters */
  double  orientation;      /* radians clockwise from north */
} Local_Cartesian_Parameters;

/* Local Cartesian Coordinate Tuple Definition */
typedef struct Local_Cartesian_Tuple_Structure
{
  double  x;  /* meters */
  double  y;  /* meters */
  double  z;  /* meters */
} Local_Cartesian_Tuple;

/* Mercator Coordinate System Definition */
typedef struct Mercator_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  scale_factor;      /* unitless */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Mercator_Parameters;

/* Mercator Coordinate Tuple Definition */
typedef struct Mercator_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Mercator_Tuple;

/* MGRS Coordinate Tuple Definition */
typedef struct MGRS_Tuple_Structure
{
  char    string[21];
} MGRS_Tuple;

/* Miller Cylindrical Coordinate System Definition */
typedef struct Miller_Cylindrical_Structure
{
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Miller_Cylindrical_Parameters;

/* Miller Cylindrical Coordinate Tuple Definition */
typedef struct Miller_Cylindrical_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Miller_Cylindrical_Tuple;

/* Mollweide Coordinate System Definition */
typedef struct Mollweide_Structure
{
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Mollweide_Parameters;
```

```
/* Mollweide Coordinate Tuple Definition */
typedef struct Mollweide_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Mollweide_Tuple;

/* Orthographic Coordinate System Definition */
typedef struct Orthographic_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Orthographic_Parameters;

/* Orthographic Coordinate Tuple Definition */
typedef struct Orthographic_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Orthographic_Tuple;

/* Polar Stereographic Coordinate System Definition */
typedef struct Polar_Stereo_Structure
{
  double  latitude_of_true_scale;   /* radians */
  double  longitude_down_from_pole; /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Polar_Stereo_Parameters;

/* Polar Stereographic Coordinate Tuple Definition */
typedef struct Polar_Stereo_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Polar_Stereo_Tuple;

/* Polyconic Coordinate System Definition */
typedef struct Polyconic_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Polyconic_Parameters;

/* Polyconic Coordinate Tuple Definition */
typedef struct Polyconic_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Polyconic_Tuple;

/* Sinusoidal Coordinate System Definition */
typedef struct Sinusoidal_Structure
{
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
```

```
} Sinusoidal_Parameters;

/* Sinusoidal Coordinate Tuple Definition */
typedef struct Sinusoidal_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Sinusoidal_Tuple;

/* Transverse Cylindrical Equal Area Coordinate System Definition */
typedef struct Transverse_Cylindrical_Equal_Area_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  scale_factor;      /* unitless */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Transverse_Cylindrical_Equal_Area_Parameters;

/* Transverse Cylindrical Equal Area Coordinate Tuple Definition */
typedef struct Transverse_Cylindrical_Equal_Area_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Transverse_Cylindrical_Equal_Area_Tuple;

/* Transverse Mercator Coordinate System Definition */
typedef struct Transverse_Mercator_Structure
{
  double  origin_latitude;   /* radians */
  double  central_meridian;  /* radians */
  double  scale_factor;      /* unitless */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Transverse_Mercator_Parameters;

/* Transverse Mercator Coordinate Tuple Definition */
typedef struct Transverse_Mercator_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Transverse_Mercator_Tuple;

/* UPS Coordinate Tuple Definition */
typedef struct UPS_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
  char    hemisphere;  /* "N" or "S" */
} UPS_Tuple;

/* UTM Coordinate System Definition */
typedef struct UTM_Structure
{
  long     zone;     /* 1..60 */
  long     override; /* 0 or 1 */
} UTM_Parameters;

/* UTM Coordinate Tuple Definition */
typedef struct UTM_Tuple_Structure
{
  double  easting;     /* meters */
```

```
  double  northing;    /* meters */
  long    zone;        /* 1..60 */
  char    hemisphere;  /* "N" or "S" */
}  UTM_Tuple;

/* Van Der Grinten Coordinate System Definition */
typedef struct Van_der_Grinten_Structure
{
  double  central_meridian;  /* radians */
  double  false_easting;     /* meters */
  double  false_northing;    /* meters */
} Van_der_Grinten_Parameters;

/* Van Der Grinten Coordinate Tuple Definition */
typedef struct Van_der_Grinten_Tuple_Structure
{
  double  easting;     /* meters */
  double  northing;    /* meters */
} Van_der_Grinten_Tuple;
```

4.2 CONSTANTS

The following is a list of significant visible constants declared globally in GEOTRANS ENGINE with their descriptions.

```
Engine return status codes:
ENGINE_INPUT_WARNING             : Warning returned by 1st conversion
ENGINE_INPUT_ERROR               : Error returned by 1st conversion
ENGINE_OUTPUT_WARNING            : Warning returned by 2nd conversion
ENGINE_OUTPUT_ERROR              : Error returned by 2nd conversion
ENGINE_NOT_INITIALIZED           : Initialize_Engine has not been called
ENGINE_ELLIPSOID_ERROR           : Error returned by Ellipsoid module
ENGINE_DATUM_ERROR               : Error returned by Datum module
ENGINE_GEOID_ERROR               : Error returned by Geoid module
ENGINE_INVALID_TYPE              : Invalid coordinate system type
ENGINE_INVALID_DIRECTION         : Invalid direction (input or output)
ENGINE_INVALID_STATE             : Invalid state (interactive or file)
ENGINE_INVALID_INDEX_ERROR       : Index is an invalid value
ENGINE_INVALID_CODE_ERROR        : Code was not found in table
ENGINE_ELLIPSOID_OVERFLOW        : Ellipsoid table overflow
ENGINE_DATUM_OVERFLOW            : 3-parameter datum table overflow
ENGINE_DATUM_SIGMA_ERROR         : Invalid standard error value
ENGINE_DATUM_DOMAIN_ERROR        : Invalid local datum domain of validity

Conversion status codes:
ENGINE_ORIGIN_LAT_ERROR          : Invalid Origin Latitude
ENGINE_CENT_MER_ERROR            : Invalid Central Meridian
ENGINE_EASTING_ERROR             : Invalid Easting
ENGINE_NORTHING_ERROR            : Invalid Northing
ENGINE_RADIUS_ERROR              : Point too far from center of projection
ENGINE_HEMISPHERE_ERROR          : Invalid Hemisphere
ENGINE_SCALE_FACTOR_ERROR        : Invalid Scale Factor
ENGINE_LON_WARNING               : Distortion will result if longitude is
                                     too far from Central Meridian
ENGINE_ORIGIN_LON_ERROR          : Invalid Origin Longitude
ENGINE_DATUM_WARNING             : Point outside datum validity rectangle
ENGINE_STDP_ERROR                : Invalid Standard Parallel
ENGINE_FIRST_STDP_ERROR          : Invalid 1st Standard Parallel
ENGINE_SECOND_STDP_ERROR         : Invalid 2nd Standard Parallel
```

```
ENGINE_FIRST_SECOND_ERROR       : 1st & 2nd Standard Parallel cannot be
                                   Opposite latitudes
ENGINE_ZONE_ERROR               : Invalid UTM Zone
ENGINE_ZONE_OVERRIDE_ERROR      : Invalid UTM zone Override
ENGINE_MGRS_STR_ERROR           : Invalid MGRS String
ENGINE_GEOREF_STR_ERROR         : Invalid GEOREF String
ENGINE_STR_LON_MIN_ERROR        : GEOREF string long. min. error
ENGINE_STR_LAT_MIN_ERROR        : GEOREF string lat. min. error

Common status codes
ENGINE_NO_ERROR                 : No error
ENGINE_LAT_ERROR                : Invalid Latitude
ENGINE_LON_ERROR                : Invalid Longitude
ENGINE_A_ERROR                  : Invalid Ellipsoid Semi-Major Axis
ENGINE_B_ERROR                  : Invalid Ellipsoid Semi-Minor Axis
ENGINE_A_LESS_B_ERROR           : Semi-Major Axis less than Semi_Minor Axis
```

4.3 VARIABLES

None.

4.4 INCLUDE FILES

```
ctype.h                        : Standard C character handling library
math.h                         : Standard C math library
stdio.h                        : Standard C input/output library
stdlib.h                       : Standard C general utility library
string.h                       : Standard C string handling library

albers.h                       : Albers Equal Area Conic projection
bonne.h                        : Bonne projection
cassini.h                      : Cassini projection
cyleqa.h                       : Cylindrical Equal Area
datum.h                        : Datum transformation
eckert4.h                      : Eckert IV projection
eckert6.h                      : Eckert VI projection
ellipse.h                      : Ellipsoid parameters
eqdcyl.h                       : Equidistant Cylindrical projection
geocent.h                      : Geocentric coordinates
georef.h                       : GEOREF coordinates
lambert.h                      : Lambert Conformal Conic projection
loccart.h                      : Local Cartesian coordinates
mercator.h                     : Mercator projection
mgrs.h                         : Military Grid Reference System coordinates
miller.h                       : Miller Cylindrical projection
mollweid.h                     : Mollweide projection
orthogr.h                      : Orthographic projection
polar.h                        : Polar Stereographic projection
polycon.h                      : Polyconic projection
sinusoid.h                     : Sinusoidal projection
trcyleqa.h                     : Transverse Cylindrical Equal Area projection
tranmerc.h                     : Transverse Mercator projection
ups.h                          : Universal Polar Stereographic coordinates
utm.h                          : Universal Transverse Mercator coordinates
grinten.h                      : Van der Grinten projection
```

4.5 DEPENDENCIES

This component depends on a collection of other components that perform individual coordinate conversions, map projections, and datum transformations, including:

- ALBERS – which performs forward and inverse Albers Equal Area Conic projections,
- BONNE – which performs forward and inverse Bonne projections,
- CASSINI – which performs forward and inverse Cassini projections,
- CYLINDRICAL EQUAL AREA – which performs forward and inverse Cylindrical Equal Area projections,
- DATUM – which performs datum transformations between WGS 84, WGS 72, and all standard local datums,
- ECKERT4 – which performs forward and inverse Eckert IV projections,
- ECKERT6 – which performs forward and inverse Eckert VI projections,
- ELLIPSOID – which provides parameter values for all standard ellipsoids,
- EQUIDISTANT CYLINDRICAL – which performs forward and inverse Equidistant Cylindrical projections,
- GEOID – which performs conversions between WGS 84 ellipsoid heights and geoid heights,
- GEOREF – which performs conversions between geodetic and World Geodetic Reference System coordinates,
- LAMBERT – which performs forward and inverse Lambert Conformal Conic projections,
- LOCAL CARTESIAN – which performs conversions between geodetic, geocentric, and Local Cartesian projections,
- MERCATOR – which performs forward and inverse Mercator projections,
- MGRS – which performs conversions between geodetic, UTM, UPS, and Military Grid Reference System coordinates,

- MILLER – which performs forward and inverse Miller Cylindrical projections,
- MOLLWEIDE – which performs forward and inverse Mollweide projections,
- ORTHOGRAPHIC – which performs forward and inverse Orthographic projections,
- POLAR STEREO – which performs forward and inverse Polar Stereographic projections,
- POLYCONIC – which performs forward and inverse Polyconic projections,
- SINUSOIDAL – which performs forward and inverse Sinusoidal projections,
- TRANSVERSE CYLINDRICAL EQUAL AREA – which performs forward and inverse Transverse Cylindrical Equal Area projections,
- TRANSVERSE MERCATOR – which performs forward and inverse Transverse Mercator projections,
- UPS – which performs conversions between geodetic and Universal Polar Stereographic coordinates,
- UTM – which performs conversions between geodetic and Universal Transverse Mercator coordinates,
- VAN DER GRINTEN – which performs forward and inverse Van der Grinten projections.

This component also depends on the standard C character handling, math, input/output, string handling, and general utility libraries.

SECTION 5. FUNCTIONS

This component contains the following public functions:

Initialize_Engine – This function sets the initial state of the engine in preparation for coordinate conversion and/or datum transformation operations.

Get_Coordinate_System_Count – This function returns the number of different coordinate systems (including projections and grid systems) supported by the engine.

Get_Coordinate_System_Index – This function returns the index of the coordinate system with the specified code.

Get_Coordinate_System_Type – This function returns the type of the coordinate system with the specified index.

Get_Coordinate_System_Name – This function returns the name of the coordinate system with the specified index.

Get_Coordinate_System_Code – This function returns the code of the coordinate system with the specified index.

Set_Coordinate_System – This function sets the current input or output coordinate system to the specified type.

Get_Coordinate_System – This function returns the type of the current input or output coordinate system.

Get_Datum_Count – This function returns the number of different datums supported by the engine.

Get_Datum_Index – This function returns the index of the datum with the specified code.

Get_Datum_Name – This function returns the name of the datum with the specified index.

Get_Datum_Code – This function returns the code of the datum with the specified index.

Get_Datum_Ellipsoid_Code – This function returns the code for the ellipsoid associated with the datum with the specified index.

Set_ Datum – This function sets the current input or output datum to the datum with the specified index.

Get_ Datum – This function returns the index of the current input or output datum.

Define_Datum – This function creates a new local 3-parameter datum with the specified code, name, shift values (relative to WGS84), standard errors values, and domain of validity.

Get_Ellipsoid_Count – This function returns the number of different ellipsoids supported by the engine.

Get_Ellipsoid_Index – This function returns the index of the ellipsoid with the specified code.

Get_Ellipsoid_Name – This function returns the name of the ellipsoid with the specified index.

Get_Ellipsoid_Code – This function returns the code of the ellipsoid with the specified index.

Define_Ellipsoid – This function creates a new ellipsoid with the specified code, name, and semi-major and semi-minor axis values.

Set_Precision – This function sets the current output precision to the specified level.

Get_Precision – This function returns the current output precision level.

Set_Geocentric_Coordinates – This function sets the current input or output Geocentric coordinates to the specified values.

Get_Geocentric_Coordinates – This function returns the current input or output Geocentric coordinates.

Set_Geodetic_Params – This function sets the current input or output Geodetic coordinate system parameters to the specified values.

Get_Geodetic_Params – This function returns the current input or output Geodetic coordinate system parameters.

Set_Geodetic_Coordinates – This function sets the current input or output Geodetic coordinates to the specified values.

Get_Geodetic_Coordinates – This function returns the current input or output Geodetic coordinates.

Set_GEOREF_Coordinates – This function sets the current input or output GEOREF coordinate string to the specified value.

Get_GEOREF_Coordinates – This function returns the current input or output GEOREF coordinate string.

Set_Albers_Equal_Area_Conic_Params – This function sets the current input or output Albers Equal Area Conic projection parameters to the specified values.

Get_Albers_Equal_Area_Conic_Params – This function returns the current input or output Albers Equal Area Conic projection parameters.

Set_Albers_Equal_Area_Conic_Coordinates – This function sets the current input or output Albers Equal Area Conic projection coordinates to the specified values.

Get_Albers_Equal_Area_Conic_Coordinates – This function returns the current input or output Albers Equal Area Conic projection coordinates.

Set_Bonne_Params – This function sets the current input or output Bonne projection parameters to the specified values.

Get_Bonne_Params – This function returns the current input or output Bonne projection parameters.

Set_Bonne_Coordinates – This function sets the current input or output Bonne projection coordinates to the specified values.

Get_Bonne_Coordinates – This function returns the current input or output Bonne projection coordinates.

Set_Cassini_Params – This function sets the current input or output Cassini projection parameters to the specified values.

Get_Cassini_Params – This function returns the current input or output Cassini projection parameters.

Set_Cassini_Coordinates – This function sets the current input or output Cassini projection coordinates to the specified values.

Get_Cassini_Coordinates – This function returns the current input or output Cassini projection coordinates.

Set_Cylindrical_Equal_Area_Params – This function sets the current input or output Cylindrical Equal Area projection parameters to the specified values.

Get_Cylindrical_Equal_Area_Params – This function returns the current input or output Cylindrical Equal Area projection parameters.

Set_Cylindrical_Equal_Area_Coordinates – This function sets the current input or output Cylindrical Equal Area projection coordinates to the specified values.

Get_Cylindrical_Equal_Area_Coordinates – This function returns the current input or output Cylindrical Equal Area projection coordinates.

Set_Eckert4_Params – This function sets the current input or output Eckert IV projection parameters to the specified values.

Get_Eckert4_Params – This function returns the current input or output Eckert IV projection parameters.

Set_Eckert4_Coordinates – This function sets the current input or output Eckert IV projection coordinates to the specified values.

Get_Eckert4_Coordinates – This function returns the current input or output Eckert IV projection coordinates.

Set_Eckert6_Params – This function sets the current input or output Eckert VI projection parameters to the specified values.

Get_Eckert6_Params – This function returns the current input or output Eckert VI projection parameters.

Set_Eckert6_Coordinates – This function sets the current input or output Eckert VI projection coordinates to the specified values.

Get_Eckert6_Coordinates – This function returns the current input or output Eckert VI projection coordinates.

Set_Equidistant_Cylindrical_Params – This function sets the current input or output Equidistant Cylindrical projection parameters to the specified values.

Get_Equidistant_Cylindrical_Params – This function returns the current input or output Equidistant Cylindrical projection parameters.

Set_Equidistant_Cylindrical_Coordinates – This function sets the current input or output Equidistant Cylindrical projection coordinates to the specified values.

Get_Equidistant_Cylindrical_Coordinates – This function returns the current input or output Equidistant Cylindrical projection coordinates.

Set_Lambert_Conformal_Conic_Params – This function sets the current input or output Lambert Conformal Conic projection parameters to the specified values.

Get_Lambert_Conformal_Conic_Params – This function returns the current input or output Lambert Conformal Conic projection parameters.

Set_Lambert_Conformal_Conic_Coordinates – This function sets the current input or output Lambert Conformal Conic projection coordinates to the specified values.

Get_Lambert_Conformal_Conic_Coordinates – This function returns the current input or output Lambert Conformal Conic projection coordinates.

Set_Local_Cartesian_Params – This function sets the current input or output Local Cartesian coordinate system parameters to the specified values.

Get_Local_Cartesian_Params – This function returns the current input or output Local Cartesian coordinate system parameters.

Set_Local_Cartesian_Conic_Coordinates – This function sets the current input or output Local Cartesian coordinates to the specified values.

Get_Local_Cartesian_Conic_Coordinates – This function returns the current input or output Local Cartesian coordinates.

Set_Mercator_Params – This function sets the current input or output Mercator projection parameters to the specified values.

Get_Mercator_Params – This function returns the current input or output Mercator projection parameters.

Set_Mercator_Coordinates – This function sets the current input or output Mercator projection coordinates to the specified values.

Get_Mercator_Coordinates – This function returns the current input or output Mercator projection coordinates.

Set_MGRS_Coordinates – This function sets the current input or output MGRS coordinate string.

Get_MGRS_Coordinates – This function returns the current input or output MGRS coordinate string.

Set_Miller_Cylindrical_Params – This function sets the current input or output Miller Cylindrical projection parameters to the specified values.

Get_Miller_Cylindrical_Params – This function returns the current input or output Miller Cylindrical projection parameters.

Set_Miller_Cylindrical_Coordinates – This function sets the current input or output Miller Cylindrical projection coordinates to the specified values.

Get_Miller_Cylindrical_Coordinates – This function returns the current input or output Miller Cylindrical projection coordinates.

Set_Mollweide_Params – This function sets the current input or output Mollweide projection parameters to the specified values.

Get_Mollweide_Params – This function returns the current input or output Mollweide projection parameters.

Set_Mollweide_Coordinates – This function sets the current input or output Mollweide projection coordinates to the specified values.

Get_Mollweide_Coordinates – This function returns the current input or output Mollweide projection coordinates.

Set_Orthographic_Params – This function sets the current input or output Orthographic projection parameters to the specified values.

Get_Orthographic_Params – This function returns the current input or output Orthographic projection parameters.

Set_Orthographic_Coordinates – This function sets the current input or output Orthographic projection coordinates to the specified values.

Get_Orthographic_Coordinates – This function returns the current input or output Orthographic projection coordinates.

Set_Polar_Stereo_Params – This function sets the current input or output Polar Stereographic projection parameters to the specified values.

Get_Polar_Stereo_Params – This function returns the current input or output Polar Stereographic projection parameters.

Set_Polar_Stereo_Coordinates – This function sets the current input or output Polar Stereographic projection coordinates to the specified values.

Get_Polar_Stereo_Coordinates – This function returns the current input or output Polar Stereographic projection coordinates.

Set_Polyconic_Params – This function sets the current input or output Polyconic projection parameters to the specified values.

Get_Polyconic_Params – This function returns the current input or output Polyconic projection parameters.

Set_Polyconic_Coordinates – This function sets the current input or output Polyconic projection coordinates to the specified values.

Get_Polyconic_Coordinates – This function returns the current input or output Polyconic projection coordinates.

Set_Sinusoidal_Params – This function sets the current input or output Sinusoidal projection parameters to the specified values.

Get_Sinusoidal_Params – This function returns the current input or output Sinusoidal projection parameters.

Set_Sinusoidal_Coordinates – This function sets the current input or output Sinusoidal projection coordinates to the specified values.

Get_Sinusoidal_Coordinates – This function returns the current input or output Sinusoidal projection coordinates.

Set_Transverse_Cylindrical_Equal_Area_Params – This function sets the current input or output Transverse Cylindrical Equal Area projection parameters to the specified values.

Get_Transverse_Cylindrical_Equal_Area_Params – This function returns the current input or output Transverse Cylindrical Equal Area projection parameters.

Set_Transverse_Cylindrical_Equal_Area_Coordinates – This function sets the current input or output Transverse Cylindrical Equal Area projection coordinates to the specified values.

Get_Transverse_Cylindrical_Equal_Area_Coordinates – This function returns the current input or output Transverse Cylindrical Equal Area projection coordinates.

Set_Transverse_Mercator_Params – This function sets the current input or output Transverse Mercator projection parameters to the specified values.

Get_Transverse_Mercator_Params – This function returns the current input or output Transverse Mercator projection parameters.

Set_Transverse_Mercator_Coordinates – This function sets the current input or output Transverse Mercator projection coordinates to the specified values.

Get_Transverse_Mercator_Coordinates – This function returns the current input or output Transverse Mercator projection coordinates.

Set_UPS_Coordinates – This function sets the current input or output Universal Polar Stereographic (UPS) projection coordinates to the specified values.

Get_UPS_Coordinates – This function returns the current input or output Universal Polar Stereographic (UPS) projection coordinates.

Set_UTM_Params – This function sets the current input or output Universal Transverse Mercator (UTM) projection parameters to the specified values.

Get_UTM_Params – This function returns the current input or output Universal Transverse Mercator (UTM) projection parameters.

Set_UTM_Coordinates – This function sets the current input or output Universal Transverse Mercator (UTM) projection coordinates to the specified values.

Get_UTM_Coordinates – This function returns the current input or output Universal Transverse Mercator (UTM) projection coordinates.

Set_Van_der_Grinten_Params – This function sets the current input or output Van der Grinten projection parameters to the specified values.

Get_Van_der_Grinten_Params – This function returns the current input or output Van der Grinten projection parameters.

Set_Van_der_Grinten_Coordinates – This function sets the current input or output Van der Grinten projection coordinates to the specified values.

Get_Van_der_Grinten_Coordinates – This function returns the current input or output Van der Grinten projection coordinates.

Convert – This function converts the current input coordinates, according to the current input datum, coordinate system, and parameters, and the current output datum, coordinate system, and parameters.

Get_Conversion_Errors – This function returns the 90% horizontal (circular), vertical (linear), and spherical error values for the most recent conversion.

Get_Conversion_Status – This function returns the current input or output conversion status.

Get_Conversion_Status_String – This function returns a character string that corresponds to the current input or output conversion status.

Get_Return_Code_String – This function returns a character string that corresponds to the current input or output conversion status.

The following example illustrates how the Engine can be used to convert Geodetic coordinates to Mercator projection coordinates, both relative to the WGS 84 datum:

FUNCTION CALL:

```
long status;

long datum_index;

double ce90, le90, se90;

Geodetic_Parameters  input_params;

Geodetic_Tuple input_coords;

Mercator_Parameters  output_params;

Mercator_Tuple output_coords;

status = Initialize_Engine();

status = Get_Datum_Index (“WGE”, &datum_index);

status = Set_Datum (Interactive, Input, datum_index);

status = Set_Coordinate_System (Interactive, Input, Geodetic);

status = Set_Geodetic_Params (Interactive, Input, input_params);

status = Set_Geodetic_Coordinates (Interactive, Input, input_coords);

status = Set_Datum (Interactive, Output, datum_index);

status = Set_Coordinate_System (Interactive, Output, Mercator);

status = Set_Mercator_Params (Interactive, Output, output_params);

status = Convert(Interactive);

status = Get_Mercator_Coordinates (Interactive, Output, &output_coords);

status = Get_Conversion_Errors (Interactive, &ce90, &le90, &se90);
```

INPUTS:

input_params.height_type:	Ellipsoid_Height
input_coords.latitude:	-35.0
input_coords.longitude:	75.0
input_coords.height:	0.0.
output_params.central_meridian	0.0
output_params.origin_latitude	0.0
output_params.scale_factor	1.0
output_params.false_easting	0.0
output_params.false_northing	0.0

OUTPUTS:

output_coords.easting:	8348961.81
output_coords.northing:	-4139372.76
output_coords.scale_factor:	0.0
ce90:	1.0
le90	1.0
se90	1.0

5.1 INITIALIZE_ENGINE

5.1.1 DESCRIPTION

This function sets the initial state of the engine in preparation for coordinate conversion and/or datum transformation operations.

5.1.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Initialize_Engine ();
```

5.1.3 DECLARATIONS

5.1.3.1 TYPES

Not applicable.

5.1.3.2 CONSTANTS

Not applicable.

5.1.3.3 VARIABLES

Not applicable.

5.1.4 DEPENDENCIES

This function depends on the function Initialize_Datums in the DATUM component, the function Initialize_Ellipsoids in the ELLIPSOID component, and the function Initialize_Geoid in the GEOID component. This function also depends on the local public functions Datum_Count and Set_Coordinate_System.

5.1.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_ELLIPSOID_ERROR          : Error returned by Ellipsoid module
ENGINE_DATUM_ERROR              : Error returned by Datum module
ENGINE_GEOID_ERROR              : Error returned by Geoid module
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
```

5.2 GET_COORDINATE_SYSTEM_COUNT

5.2.1 DESCRIPTION

This function returns the number of different coordinate systems (including projections and grid systems) supported by the engine.

5.2.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Coordinate_System_Count (long *count);
```

count Number of coordinate systems supported (output).

5.2.3 DECLARATIONS

5.2.3.1 TYPES

Not applicable.

5.2.3.2 CONSTANTS

Not applicable.

5.2.3.3 VARIABLES

Not applicable.

5.2.4 DEPENDENCIES

None.

5.2.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
```

5.3 GET_COORDINATE_SYSTEM_INDEX

5.3.1 DESCRIPTION

This function returns the index of the coordinate system with the specified code.

5.3.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Coordinate_System_Index ( const char *code,
            long *index);
```

code	The 2-letter code for the desired coordinate system (input),
index	The index of the coordinate system with the specified code (output).

Example:

```
status = Get_Coordinate_System_Index(code, index)
```

Inputs:

code	"GE" (GEOREF)

Outputs:

index	1

5.3.3 DECLARATIONS

5.3.3.1 TYPES

Not applicable.

5.3.3.2 CONSTANTS

Not applicable.

5.3.3.3 VARIABLES

Not applicable.

5.3.4 DEPENDENCIES

None.

5.3.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_CODE_ERROR       : Coordinate system code not found in list
```

5.4 GET_COORDINATE_SYSTEM_TYPE

5.4.1 DESCRIPTION

This function returns the type of the coordinate system with the specified index.

5.4.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Coordinate_System_Type (const long index,
            Coordinate_Type *system);
```

index The index of a given coordinate system (input),

system The type of the coordinate system referenced by index (output).

Example:

```
status = Get_Coordinate_System_Type(index, system)
```

Inputs:

Index 1

Outputs:

system GEOREF

5.4.3 DECLARATIONS

5.4.3.1 TYPES

The enumerated type Coordinate_Type is used, as defined in Section 4.1.1.

5.4.3.2 CONSTANTS

Not applicable.

5.4.3.3 VARIABLES

Not applicable.

5.4.4 DEPENDENCIES

None.

5.4.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_INDEX_ERROR      : Index out of valid range
```

5.5 GET_COORDINATE_SYSTEM_NAME

5.5.1 DESCRIPTION

This function returns the name of the coordinate system with the specified index.

5.5.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Coordinate_System_Name (const long index,
            char *name);
```

index The index of a given coordinate system (input),

name The name of the coordinate system referenced by index (output).

Example:

```
status = Get_Coordinate_System_Name(index, name)
```

Inputs:

Index 1

Outputs:

name "GEOREF"

5.5.3 DECLARATIONS

5.5.3.1 TYPES

Not applicable.

5.5.3.2 CONSTANTS

Not applicable.

5.5.3.3 VARIABLES

Not applicable.

5.5.4 DEPENDENCIES

None.

5.5.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
```

```
ENGINE_INVALID_INDEX_ERROR       : Index out of valid range
```

5.6 GET_COORDINATE_SYSTEM_CODE

5.6.1 DESCRIPTION

This function returns the code of the coordinate system with the specified index.

5.6.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Coordinate_System_Code (const long index,
            char *code);
```

index	The index of a given coordinate system (input),
code	The 2-letter code of the coordinate system referenced by index (output).

5.6.3 DECLARATIONS

5.6.3.1 TYPES

Not applicable.

5.6.3.2 CONSTANTS

Not applicable.

5.6.3.3 VARIABLES

Not applicable.

5.6.4 DEPENDENCIES

None.

5.6.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_INDEX_ERROR      : Index out of valid range
```

5.7 SET_COORDINATE_SYSTEM

5.7.1 DESCRIPTION

This function sets the current input or output coordinate system to the specified type.

5.7.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Coordinate_System (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Coordinate_Type     System);
```

State	Indicates whether the coordinate system is to be set for interactive or file processing (input),
Direction	Indicates whether the coordinate system is to be set for the input or output coordinate system or projection (input),
System	The type of the coordinate system (input).

Example:

```
status = Set_Coordinate_System (State, Direction, System)
```

Inputs:

State	Interactive
Direction	Input
System	GEOREF

Outputs:

None.

5.7.3 DECLARATIONS

5.7.3.1 TYPES

The enumerated type Coordinate_Type is used, as defined in Section 4.1.1.

5.7.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.7.3.3 VARIABLES

Not applicable.

5.7.4 DEPENDENCIES

None.

5.7.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

5.8 GET_COORDINATE_SYSTEM

5.8.1 DESCRIPTION

This function returns the type of the current input or output coordinate system.

5.8.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Coordinate_System (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Coordinate_Type           *System);
```

State — Indicates whether the coordinate system is to be returned for interactive or file processing (input),

Direction — Indicates whether the coordinate system is to be returned for the input or output coordinate system or projection (input),

System — The type of the current coordinate system (output).

Example:

```
status = Get_Coordinate_System (State, Direction, &System)
```

Inputs:

State — Interactive

Direction — Input

Outputs:

System — GEOREF

5.8.3 DECLARATIONS

5.8.3.1 TYPES

The enumerated type Coordinate_Type is used, as defined in Section 4.1.1.

5.8.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.8.3.3 VARIABLES

Not applicable.

5.8.4 DEPENDENCIES

None.

5.8.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

5.9 GET_DATUM_COUNT

5.9.1 DESCRIPTION

This function returns the number of different datums supported by the engine.

5.9.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum_Count (long *count);
```

count Number of datums in the table (output).

5.9.3 DECLARATIONS

5.9.3.1 TYPES

Not applicable.

5.9.3.2 CONSTANTS

Not applicable.

5.9.3.3 VARIABLES

Not applicable.

5.9.4 DEPENDENCIES

This function depends on the function Datum_Count in the DATUM component.

5.9.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_DATUM_ERROR              : Error returned by Datum module
```

5.10 GET_DATUM_INDEX

5.10.1 DESCRIPTION

This function returns the index of the datum with the specified code.

5.10.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum_Index ( const char *code,
            long *index);
```

code — The code for the desired datum (input),

index — The index of the datum in the table with the specified code (output).

Example:

```
status = Get_Datum_Index(code, index)
```

Inputs:

code — "TOY-C"

Outputs:

index — 201

5.10.3 DECLARATIONS

5.10.3.1 TYPES

Not applicable.

5.10.3.2 CONSTANTS

Not applicable.

5.10.3.3 VARIABLES

Not applicable.

5.10.4 DEPENDENCIES

This function depends on the function Datum_Index in the DATUM component.

5.10.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_CODE_ERROR       : Datum code not found
ENGINE_DATUM_ERROR              : Error returned by Datum module
```

5.11 GET_DATUM_NAME

5.11.1 DESCRIPTION

This function returns the name of the datum with the specified index.

5.11.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum_Name (const long index,
            char *name);
```

index The index of a given datum in the datum table (input),

name The name of the datum referenced by index (output).

Example:

```
status = Get_Datum_Name(index, name)
```

Inputs:

Index 201

Outputs:

name "TOKYO Okinawa"

5.11.3 DECLARATIONS

5.11.3.1 TYPES

Not applicable.

5.11.3.2 CONSTANTS

Not applicable.

5.11.3.3 VARIABLES

Not applicable.

5.11.4 DEPENDENCIES

This function depends on the function Datum_Name in the DATUM component.

5.11.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_INDEX_ERROR      : Index out of valid range
ENGINE_DATUM_ERROR              : Error returned by Datum module
```

5.12 GET_DATUM_CODE

5.12.1 DESCRIPTION

This function returns the code of the datum with the specified index.

5.12.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum_Code (const long index,
            char *code);
```

index The index of a given datum in the datum table (input),

code The datum code of the datum referenced by index (output).

5.12.3 DECLARATIONS

5.12.3.1 TYPES

Not applicable.

5.12.3.2 CONSTANTS

Not applicable.

5.12.3.3 VARIABLES

Not applicable.

5.12.4 DEPENDENCIES

This function depends on the function Datum_Code in the DATUM component.

5.12.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_INDEX_ERROR      : Index out of valid range
ENGINE_DATUM_ERROR              : Error returned by Datum module
```

5.13 GET_DATUM_ELLIPSOID_CODE

5.13.1 DESCRIPTION

This function returns the code for the ellipsoid associated with the datum with the specified index.

5.13.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum_Ellipsoid_Code (const long index,
            char *code);
```

index	The index of a given datum in the datum table (input),
code	The ellipsoid code for the ellipsoid associated with the datum referenced by index (output).

Function Call:

```
status = Get_Datum_Ellipsoid_Code(index, code)
```

Inputs:

index	201

Outputs:

code	"BR"

5.13.3 DECLARATIONS

5.13.3.1 TYPES

Not applicable.

5.13.3.2 CONSTANTS

Not applicable.

5.13.3.3 VARIABLES

Not applicable.

5.13.4 DEPENDENCIES

This function depends on the function Datum_Ellipsoid_Code in the DATUM component.

.

5.13.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_INDEX_ERROR      : Index out of valid range
ENGINE_DATUM_ERROR              : Error returned by Datum module
```

5.14 SET_ DATUM

5.14.1 DESCRIPTION

This function sets the current input or output datum to the datum with the specified index.

5.14.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Datum (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const long                index);
```

State	Indicates whether the datum is to be set for interactive or file processing (input),
Direction	Indicates whether the datum is to be set for the input or output coordinate system or projection (input),
index	The index of a given datum in the datum table (input).

Example:

```
status = Set_Datum (State, Direction, index)
```

Inputs:

State	Interactive
Direction	Input
index	201

Outputs:

None.

5.14.3 DECLARATIONS

5.14.3.1 TYPES

Not applicable.

5.14.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.14.3.3 VARIABLES

Not applicable.

5.14.4 DEPENDENCIES

None.

5.14.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

```
ENGINE_INVALID_INDEX_ERROR       : Index out of valid range
```

5.15 GET_ DATUM

5.15.1 DESCRIPTION

This function returns the index of the current input or output datum.

5.15.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Datum (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            long                      *index);
```

State	Indicates whether the datum is to be returned for interactive or file processing (input),
Direction	Indicates whether the datum is to be returned for the input or output coordinate system or projection (input),
index	Identifies the index of the current datum (output).

Example:

```
status = Get_Datum (State, Direction, &index)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

index	201

5.15.3 DECLARATIONS

5.15.3.1 TYPES

Not applicable.

5.15.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.15.3.3 VARIABLES

Not applicable.

5.15.4 DEPENDENCIES

None.

5.15.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

5.16 DEFINE_DATUM

5.16.1 DESCRIPTION

This function creates a new local 3-parameter datum with the specified code, name, shift values (relative to WGS84), standard errors values, and domain of validity.

5.16.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Define_Datum ( const char *code,
             const char *name,
             const char *ellipsoid_code,
             double delta_x,
             double delta_y,
             double delta_z,
             double sigma_x,
             double sigma_y,
             double sigma_z,
             double south_latitude,
             double north_latitude,
             double west_longitude,
             double east_longitude)
```

code	The code for the new datum (input),
name	The name for the new datum (input),
ellipsoid_code	The 2-letter ellipsoid code for the new datum (input),
delta_X	The X translation to WGS84 in meters for the new datum (input),
delta_Y	The Y translation to WGS84 in meters for the new datum (input),
delta_Z	The Z translation to WGS84 in meters for the new datum (input),
sigma_X	The standard error in X in meters for the new datum (input),
sigma_Y	The standard error Y in meters for the new datum (input),
sigma_Z	The standard error Z in meters for the new datum (input),
south_latitude	The southern edge of the validity rectangle for the new datum in radians (input),
north_latitude	The southern edge of the validity rectangle for the new datum in radians (input),
west_longitude	The western edge of the validity rectangle for the new datum in radians (input),
east_longitude	The eastern edge of the validity rectangle for the new datum in radians (input).

Example:

```
status = Define_Datum (code, name, ellipsoid_code,
            delta_x, delta_y, delta_z,
            sigma_x, sigma_Y, sigma_Z
            south_latitude, north_latitude,
            west_longitude, east_longitude)
```

Inputs:

code	"XXX-M"
name	"Experimental datum"
ellipsoid_code	"WGE"
delta_x	25.0

delta_y	25.0
delta_z	5.0
sigma_x	2.0
sigma_y	3.0
sigma_z	1.0
south_latitude	-1.57079632…
north_latitude	1.57079632…
west_longitude	-3.14159265…
east_longitude	3.14159265…

5.16.3 DECLARATIONS

5.16.3.1 TYPES

Not applicable.

5.16.3.2 CONSTANTS

Not applicable.

5.16.3.3 VARIABLES

Not applicable.

5.16.4 DEPENDENCIES

This function depends on the function Create_Datum in the DATUM component.

5.16.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
```

```
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_CODE_ERROR       : Code was not found in table
ENGINE_ELLIPSE_ERROR            : Error returned by Ellipsoid module
ENGINE_DATUM_ERROR              : Error returned by Datum module
ENGINE_DATUM_OVERFLOW           : 3-parameter datum table overflow
ENGINE_DATUM_SIGMA_ERROR        : Invalid standard error value
ENGINE_LAT_ERROR                : Invalid Latitude
ENGINE_LON_ERROR                : Invalid Longitude
ENGINE_DATUM_DOMAIN_ERROR       : Invalid local datum domain of validity
```

5.17 GET_ELLIPSOID_COUNT

5.17.1 DESCRIPTION

This function returns the number of different ellipsoids supported by the engine.

5.17.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Ellipsoid_Count (long *count);
```

count Number of ellipsoids in the ellipsoid table (output).

5.17.3 DECLARATIONS

5.17.3.1 TYPES

Not applicable.

5.17.3.2 CONSTANTS

Not applicable.

5.17.3.3 VARIABLES

Not applicable.

5.17.4 DEPENDENCIES

This function depends on the function Ellipsoid_Count in the ELLIPSOID component.

5.17.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occured in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_ELLIPSOID_ERROR          : Error returned by Ellipsoid module
```

5.18 GET_ELLIPSOID_INDEX

5.18.1 DESCRIPTION

This function returns the index of the ellipsoid with the specified code.

5.18.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Ellipsoid_Index (const char *code,
            long *index);
```

code — The 2-letter code for the desired ellipsoid (input),

index — The index of the ellipsoid in the table with the specified code (output).

Example:

```
status = Get_Ellipsoid_Index(code, index)
```

Inputs:

code — "ED"

Outputs:

index — 11

5.18.3 DECLARATIONS

5.18.3.1 TYPES

Not applicable.

5.18.3.2 CONSTANTS

Not applicable.

5.18.3.3 VARIABLES

Not applicable.

5.18.4 DEPENDENCIES

This function depends on the function Ellipsoid_Code in the ELLIPSOID component.

5.18.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occured in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_CODE_ERROR       : Code was not found in table
ENGINE_ELLIPSOID_ERROR          : Error returned by Ellipsoid module
```

5.19 GET_ELLIPSOID_NAME

5.19.1 DESCRIPTION

This function returns the name of the ellipsoid with the specified index.

5.19.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Ellipsoid_Name (const long index,
            char *name);
```

index The index of a given ellipsoid in the ellipsoid table (input),

name The name of the ellipsoid referenced by index (output).

Example:

```
status = Get_Ellipsoid_Name(index, name)
```

Inputs:

index: 11

Outputs:

name: "Everest 1969 (West Malasia)"

5.19.3 DECLARATIONS

5.19.3.1 TYPES

Not applicable.

5.19.3.2 CONSTANTS

Not applicable.

5.19.3.3 VARIABLES

Not applicable.

5.19.4 DEPENDENCIES

This function depends on the function Ellipsoid_Name in the ELLIPSOID component.

5.19.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occured in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_INDEX_ERROR      : Index outside of valid range
ENGINE_ELLIPSOID_ERROR          : Error returned by Ellipsoid module
```

5.20 GET_ELLIPSOID_CODE

5.20.1 DESCRIPTION

This function returns the code of the ellipsoid with the specified index.

5.20.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Ellipsoid_Code (const long index,
            char *code);
```

index The index of a given ellipsoid in the ellipsoid table (input),

code The 2-letter code of the ellipsoid referenced by index (output).

5.20.3 DECLARATIONS

5.20.3.1 TYPES

Not applicable.

5.20.3.2 CONSTANTS

Not applicable.

5.20.3.3 VARIABLES

Not applicable.

5.20.4 DEPENDENCIES

This function depends on the function Ellipsoid_Code in the ELLIPSOID component.

5.20.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occured in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_INDEX_ERROR      : Index outside of valid range
ENGINE_ELLIPSOID_ERROR          : Error returned by Ellipsoid module
```

5.21 DEFINE_ELLIPSOID

5.21.1 DESCRIPTION

This function creates a new ellipsoid with the specified code, name, and semi-major and semi-minor axis values.

5.21.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Define_Ellipsoid (const char *code,
            const char *name,
            double a,
            double b);
```

code	The 2-letter code for the new ellipsoid (input),
name	The name for the new ellipsoid (input),
a	The semi-major axis, in meters, of the new ellipsoid (input),
b	The semi-minor axis, in meters, of the new ellipsoid (input).

Example:

```
status = Define_Ellipsoid(code, name, a, b)
```

Inputs:

code	"XX"
name	"Experimental ellipsoid"
a	6377295.664
b	6356094.668

5.21.3 DECLARATIONS

5.21.3.1 TYPES

Not applicable.

5.21.3.2 CONSTANTS

Not applicable.

5.21.3.3 VARIABLES

Not applicable.

5.21.4 DEPENDENCIES

This function depends on the function Create_Ellipsoid in the ELLIPSOID component.

5.21.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occured in function
ENGINE_NOT_INITIALIZED          : Engine not initialized properly
ENGINE_INVALID_CODE_ERROR       : Code was not found in table
ENGINE_A_ERROR                  : Invalid Ellipsoid Semi-Major Axis
ENGINE_B_ERROR                  : Invalid Ellipsoid Semi-Minor Axis
ENGINE_A_LESS_B_ERROR           : Semi-Major Axis less than Semi_Minor Axis
ENGINE_ELLIPSOID_ERROR          : Error returned by Ellipsoid module
```

5.22 SET_PRECISION

5.22.1 DESCRIPTION

This function sets the current output precision to the specified level.

5.22.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
void Set_Precision (
            Precision Precis);
```

Precis The desired precision level (input).

Example:

```
Set_Precision (p)
```

Inputs:

p Hundredths_of_Second

Outputs:

None.

5.22.3 DECLARATIONS

5.22.3.1 TYPES

The enumerated type Precision is used, as defined in Section 4.1.1.

5.22.3.2 CONSTANTS

Not applicable.

5.22.3.3 VARIABLES

Not applicable.

5.23.4 DEPENDENCIES

None.

5.22.5 ERROR HANDLING

None.

5.23 GET_PRECISION

5.23.1 DESCRIPTION

This function returns the current output precision level.

5.23.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
void Get_Precision (Precision *Precis);
```

Precis The current precision level (output).

Example:

```
Get_Precision (p)
```

Inputs:

p Hundredths_of_Second

Outputs:

None.

5.23.3 DECLARATIONS

5.23.3.1 TYPES

The enumerated type Precision is used, as defined in Section 4.1.1.

5.23.3.2 CONSTANTS

Not applicable.

5.23.3.3 VARIABLES

Not applicable.

5.23.4 DEPENDENCIES

None.

5.23.5 ERROR HANDLING

None.

5.24 SET_GEOCENTRIC_COORDINATES

5.24.1 DESCRIPTION

This function sets the current input or output Geocentric coordinates to the specified values.

5.24.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Geocentric_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Geocentric_Tuple    coordinates);
```

State	Indicates whether the Geocentric coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Geocentric coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Geocentric coordinates (X, Y, Z) to be set (input).

Example:

```
status = Set_Geocentric_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(0.0, 6378160.0, 0.0)

Outputs:

None.

5.24.3 DECLARATIONS

5.24.3.1 TYPES

The structure type Geocentric_Tuple is used as defined in Section 4.1.2.

5.24.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.24.3.3 VARIABLES

Not applicable.

5.24.4 DEPENDENCIES

None.

5.24.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.25 GET_GEOCENTRIC_COORDINATES

5.25.1 DESCRIPTION

This function returns the current input or output Geocentric coordinates.

5.25.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Geocentric_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
        Geocentric_Tuple          *coordinates);
```

State	Indicates whether the Geocentric coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Geocentric coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Geocentric coordinate structure (X, Y, Z) to be returned (output).

Example:

```
status = Get_Geocentric_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(0.0, 6378160.0, 0.0)

Outputs:

None.

5.25.3 DECLARATIONS

5.25.3.1 TYPES

The structure type GEOREF_Tuple is used as defined in Section 4.1.2.

5.25.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.25.3.3 VARIABLES

Not applicable.

5.25.4 DEPENDENCIES

None.

5.25.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.26 SET_GEODETIC_PARAMS

5.26.1 DESCRIPTION

This function sets the current input or output Geodetic coordinate system parameters to the specified values.

5.26.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Geodetic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Geodetic_Parameters parameters);
```

State	Indicates whether the Geodetic coordinate system parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Geodetic coordinate system parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Geodetic coordinate system parameters (Height Type) to be set (input).

Example:

```
status = Set_Geodetic_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(Ellipsoid_Height)

Outputs:

None.

5.26.3 DECLARATIONS

5.26.3.1 TYPES

The structure type Geodetic_Parameters is used as defined in Section 4.1.2.

5.26.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.26.3.3 VARIABLES

Not applicable.

5.26.4 DEPENDENCIES

None.

5.26.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.26 GET_GEODETIC_PARAMS

5.26.1 DESCRIPTION

This function returns the current input or output Geodetic coordinate system parameters.

5.27.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Geodetic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Geodetic_Parameters       *parameters);
```

State	Indicates whether the Geodetic coordinate system parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Geodetic coordinate system parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Geodetic coordinate system parameter structure (Height Type) to be returned (output).

Example:

```
status = Get_Geodetic_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(Ellipsoid_Height)

5.27.3 DECLARATIONS

5.27.3.1 TYPES

The structure type Geodetic_Parameters is used as defined in Section 4.1.2.

5.27.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.27.3.3 VARIABLES

Not applicable.

5.27.4 DEPENDENCIES

None.

5.27.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.28 SET_GEODETIC_COORDINATES

5.28.1 DESCRIPTION

This function sets the current input or output Geodetic coordinates to the specified values.

5.28.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Geodetic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Geodetic_Tuple      coordinates);
```

State — Indicates whether the Geodetic coordinates are to be set for interactive or file processing (input),

Direction — Indicates whether the Geodetic coordinates are to be set for the input or output coordinate system or projection (input),

coordinates — Structure containing Geodetic coordinates (Latitude, Longitude, Height) to be set (input).

Example:

```
status = Set_Geodetic_Coordinates (State, Direction, coordinates)
```

Inputs:

State — Interactive

Direction — Input

coordinates — (45.0*PI/180.0, -75.0*PI/180.0, 0.0)

Outputs:

None.

5.28.3 DECLARATIONS

5.28.3.1 TYPES

The structure type Geodetic_Tuple is used as defined in Section 4.1.2.

5.28.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.28.3.3 VARIABLES

Not applicable.

5.28.4 DEPENDENCIES

None.

5.28.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.29 GET_GEODETIC_COORDINATES

5.29.1 DESCRIPTION

This function returns the current input or output Geodetic coordinates.

5.29.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Geodetic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Geodetic_Tuple            *coordinates);
```

State	Indicates whether the Geodetic coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Geodetic coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Geodetic coordinate structure (Latitude, Longitude, Height) to be returned (output).

Example:

```
status = Get_Geodetic_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(45.0*PI/180.0, -75.0*PI/180.0, 0.0)

Outputs:

None.

5.29.3 DECLARATIONS

5.29.3.1 TYPES

The structure type Geodetic_Tuple is used as defined in Section 4.1.2.

5.29.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.29.3.3 VARIABLES

Not applicable.

5.29.4 DEPENDENCIES

None.

5.29.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.30 SET_GEOREF_COORDINATES

5.30.1 DESCRIPTION

This function sets the current input or output GEOREF coordinate string to the specified value.

5.30.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_GEOREF_Coordinates (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const GEOREF_Tuple        coordinates);
```

State	Indicates whether the GEOREF coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the GEOREF coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing GEOREF coordinates (GEOREF string) to be set (input).

Example:

```
status = Set_GEOREF_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(AABB15001500)

Outputs:

None.

5.30.3 DECLARATIONS

5.30.3.1 TYPES

The structure type GEOREF_Tuple is used as defined in Section 4.1.2.

5.30.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.30.3.3 VARIABLES

Not applicable.

5.30.4 DEPENDENCIES

None.

5.30.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.31 GET_GEOREF_COORDINATES

5.31.1 DESCRIPTION

This function returns the current input or output GEOREF coordinate string.

5.31.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_GEOREF_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            GEOREF_Tuple              *coordinates);
```

State	Indicates whether the GEOREF coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the GEOREF coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	GEOREF coordinate structure (GEOREF string) to be returned (output).

Example:

```
status = Get_GEOREF_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive

Direction Input

coordinates (AABB15001500)

Outputs:

None.

5.31.3 DECLARATIONS

5.31.3.1 TYPES

The structure type GEOREF_Tuple is used as defined in Section 4.1.2.

5.31.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.31.3.3 VARIABLES

Not applicable.

5.31.4 DEPENDENCIES

None.

5.31.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.32 SET_ALBERS_EQUAL_AREA_CONIC_PARAMS

5.32.1 DESCRIPTION

This function sets the current input or output Albers Equal Area Conic projection parameters to the specified values.

5.32.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Albers_Equal_Area_Conic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Albers_Equal_Area_Conic_Parameters parameters);
```

State	Indicates whether the Albers Equal Area Conic projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Albers Equal Area Conic projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Albers Equal Area Conic projection parameters (Origin Latitude, Central Meridian, 1st Standard Parallel, 2nd Standard Parallel, False Easting, False Northing) to be set (output).

Example:

```
status = Set_Albers_Equal_Area_Conic_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 40.0*PI/180.0, 50.0*PI/180.0, 2000000.0, 5000000.0)

Outputs:

None.

5.32.3 DECLARATIONS

5.32.3.1 TYPES

The structure type Albers_Equal_Area_Conic_Parameters is used as defined in Section 4.1.2.

5.32.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.32.3.3 VARIABLES

Not applicable.

5.32.4 DEPENDENCIES

None.

5.32.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.33 GET_ALBERS_EQUAL_AREA_CONIC_PARAMS

5.33.1 DESCRIPTION

This function returns the current input or output Albers Equal Area Conic projection parameters.

5.33.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Albers_Equal_Area_Conic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Albers_Equal_Area_Conic_Parameters *parameters);
```

State	Indicates whether the Albers Equal Area Conic projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Albers Equal Area Conic projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Albers Equal Area Conic projection parameter structure (Origin Latitude, Central Meridian, 1st Standard Parallel, 2nd Standard Parallel, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Albers_Equal_Area_Conic_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 40.0*PI/180.0, 50.0*PI/180.0, 2000000.0, 5000000.0)

5.33.3 DECLARATIONS

5.33.3.1 TYPES

The structure type Albers_Equal_Area_Conic_Parameters is used as defined in Section 4.1.2.

5.33.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.33.3.3 VARIABLES

Not applicable.

5.33.4 DEPENDENCIES

None.

5.33.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.34 SET_ALBERS_EQUAL_AREA_CONIC_COORDINATES

5.34.1 DESCRIPTION

This function sets the current input or output Albers Equal Area Conic projection coordinates to the specified values.

5.34.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Albers_Equal_Area_Conic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Albers_Equal_Area_Conic_Tuple coordinates);
```

State	Indicates whether the Albers Equal Area Conic projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Albers Equal Area Conic projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Albers Equal Area Conic projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Albers_Equal_Area_Conic_Coordinates (State, Direction,
           coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(2000000.0, 5000000.0)

Outputs:

None.

5.34.3 DECLARATIONS

5.34.3.1 TYPES

The structure type Albers_Equal_Area_Conic_Tuple is used as defined in Section 4.1.2.

5.34.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.34.3.3 VARIABLES

Not applicable.

5.34.4 DEPENDENCIES

None.

5.34.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
```

```
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.35 GET_ALBERS_EQUAL_AREA_CONIC_COORDINATES

5.35.1 DESCRIPTION

This function returns the current input or output Albers Equal Area Conic projection coordinates.

5. 35.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Albers_Equal_Area_Conic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Albers_Equal_Area_Conic_Tuple *coordinates);
```

State	Indicates whether the Albers Equal Area Conic projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Albers Equal Area Conic projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Albers Equal Area Conic projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Albers_Equal_Area_Conic_Coordinates (State, Direction,
            &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(2000000.0, 5000000.0)

Outputs:

None.

5. 35.3 DECLARATIONS

5. 35.3.1 TYPES

The structure type Albers_Equal_Area_Conic_Tuple is used as defined in Section 4.1.2.

5. 35.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5. 35.3.3 VARIABLES

Not applicable.

5. 35.4 DEPENDENCIES

None.

5.36.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.36 SET_BONNE_PARAMS

5.36.1 DESCRIPTION

This function sets the current input or output Bonne projection parameters to the specified values.

5.36.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Bonne_Params (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Bonne_Parameters    parameters);
```

State	Indicates whether the Bonne projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Bonne projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Bonne projection parameters (Origin Latitude, Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Bonne_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.36.3 DECLARATIONS

5.36.3.1 TYPES

The structure type Bonne_Parameters is used as defined in Section 4.1.2.

5.36.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.36.3.3 VARIABLES

Not applicable.

5.36.4 DEPENDENCIES

None.

5.36.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.37 GET_BONNE_PARAMS

5.37.1 DESCRIPTION

This function returns the current input or output Bonne projection parameters.

5.37.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Bonne_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Bonne_Parameters          *parameters);
```

State	Indicates whether the Bonne projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Bonne projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Bonne projection parameter structure (Origin Latitude, Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Bonne_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive

Direction Input

Outputs:

parameters (45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

5.37.3 DECLARATIONS

5.37.3.1 TYPES

The structure type Bonne_Parameters is used as defined in Section 4.1.2.

5.37.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.37.3.3 VARIABLES

Not applicable.

5.37.4 DEPENDENCIES

None.

5.37.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.38 SET_BONNE_COORDINATES

5.38.1 DESCRIPTION

This function sets the current input or output Bonne projection coordinates to the specified values.

5.38.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Bonne_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Bonne_Tuple         coordinates);
```

State	Indicates whether the Bonne projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Bonne projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Bonne projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Bonne_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.38.3 DECLARATIONS

5.38.3.1 TYPES

The structure type Bonne_Tuple is used as defined in Section 4.1.2.

5.38.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.38.3.3 VARIABLES

Not applicable.

5.38.4 DEPENDENCIES

None.

5.38.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.39 GET_BONNE_COORDINATES

5.39.1 DESCRIPTION

This function returns the current input or output Bonne projection coordinates.

5.39.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Bonne_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Bonne_Tuple               *coordinates);
```

State	Indicates whether the Bonne projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Bonne projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Bonne projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Bonne_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.39.3 DECLARATIONS

5.39.3.1 TYPES

The structure type Bonne_Tuple is used as defined in Section 4.1.2.

5.39.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.39.3.3 VARIABLES

Not applicable.

5.39.4 DEPENDENCIES

None.

5.39.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

```
ENGINE_INVALID_TYPE              : Invalid coordinate system type
```

5.40 SET_CASSINI_PARAMS

5.40.1 DESCRIPTION

This function sets the current input or output Cassini projection parameters to the specified values.

5.40.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Cassini_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Cassini_Parameters  parameters);
```

State	Indicates whether the Cassini projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Cassini projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Cassini projection parameters (Origin Latitude, Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Cassini_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.40.3 DECLARATIONS

5.40.3.1 TYPES

The structure type Cassini_Parameters is used as defined in Section 4.1.2.

5.40.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.40.3.3 VARIABLES

Not applicable.

5.40.4 DEPENDENCIES

None.

5.40.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.41 GET_CASSINI_PARAMS

5.41.1 DESCRIPTION

This function returns the current input or output Cassini projection parameters.

5.41.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Cassini_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
        Cassini_Parameters        *parameters);
```

State	Indicates whether the Cassini projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Cassini projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Cassini projection parameter structure (Origin Latitude, Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Cassini_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

5.41.3 DECLARATIONS

5.41.3.1 TYPES

The structure type Cassini_Parameters is used as defined in Section 4.1.2.

5.41.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.41.3.3 VARIABLES

Not applicable.

5.41.4 DEPENDENCIES

None.

5.41.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.42 SET_CASSINI_COORDINATES

5.42.1 DESCRIPTION

This function sets the current input or output Cassini projection coordinates to the specified values.

5.42.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Cassini_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Cassini_Tuple         coordinates);
```

State	Indicates whether the Cassini projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Cassini projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Cassini projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Cassini_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive

Direction Input

coordinates (4000000.0, 4000000.0)

Outputs:

None.

5.42.3 DECLARATIONS

5.42.3.1 TYPES

The structure type Cassini_Tuple is used as defined in Section 4.1.2.

5.42.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.42.3.3 VARIABLES

Not applicable.

5.42.4 DEPENDENCIES

None.

5.42.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.43 GET_CASSINI_COORDINATES

5.43.1 DESCRIPTION

This function returns the current input or output Cassini projection coordinates.

5.43.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Cassini_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Cassini_Tuple             *coordinates);
```

State	Indicates whether the Cassini projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Cassini projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Cassini projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Cassini_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.43.3 DECLARATIONS

5.43.3.1 TYPES

The structure type Cassini_Tuple is used as defined in Section 4.1.2.

5.43.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.43.3.3 VARIABLES

Not applicable.

5.43.4 DEPENDENCIES

None.

5.43.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.44 SET_CYLINDRICAL_EQUAL_AREA_PARAMS

5.44.1 DESCRIPTION

This function sets the current input or output Cylindrical Equal Area projection parameters to the specified values.

5.44.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Cylindrical_Equal_Area_Params (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Cylindrical_Equal_Area_Parameters parameters);
```

State	Indicates whether the Cylindrical Equal Area projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Cylindrical Equal Area projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Cylindrical Equal Area projection parameters (Origin Latitude, Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Cylindrical_Equal_Area_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.44.3 DECLARATIONS

5.44.3.1 TYPES

The structure type Cylindrical_Equal_Area_Parameters is used as defined in Section 4.1.2.

5.44.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.44.3.3 VARIABLES

Not applicable.

5.44.4 DEPENDENCIES

None.

5.44.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.45 GET_CYLINDRICAL_EQUAL_AREA_PARAMS

5.45.1 DESCRIPTION

This function returns the current input or output Cylindrical Equal Area projection parameters.

5.45.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Cylindrical_Equal_Area_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Cylindrical_Equal_Area_Parameters *parameters);
```

State	Indicates whether the Cylindrical Equal Area projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Cylindrical Equal Area projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Cylindrical Equal Area projection parameter structure (Origin Latitude, Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Cylindrical_Equal_Area_Params (State, Direction, &parameters)
```

Inputs:

State Interactive

Direction Input

Outputs:

parameters (45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

5.45.3 DECLARATIONS

5.45.3.1 TYPES

The structure type Cylindrical_Equal_Area_Parameters is used as defined in Section 4.1.2.

5.45.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.45.3.3 VARIABLES

Not applicable.

5.45.4 DEPENDENCIES

None.

5.45.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.46 SET_CYLINDRICAL_EQUAL_AREA_COORDINATES

5.46.1 DESCRIPTION

This function sets the current input or output Cylindrical Equal Area projection coordinates to the specified values.

5.46.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Cylindrical_Equal_Area_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Cylindrical_Equal_Area_Tuple coordinates);
```

State	Indicates whether the Cylindrical Equal Area projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Cylindrical Equal Area projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Cylindrical Equal Area projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Cylindrical_Equal_Area_Coordinates (State, Direction,
            coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.46.3 DECLARATIONS

5.46.3.1 TYPES

The structure type Cylindrical_Equal_Area_Tuple is used as defined in Section 4.1.2.

5.46.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.46.3.3 VARIABLES

Not applicable.

5.46.4 DEPENDENCIES

None.

5.46.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.47 GET_CYLINDRICAL_EQUAL_AREA_COORDINATES

5.47.1 DESCRIPTION

This function returns the current input or output Cylindrical Equal Area projection coordinates.

5.47.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Cylindrical_Equal_Area_Coordinates (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
Cylindrical_Equal_Area_Tuple *coordinates);
```

State	Indicates whether the Cylindrical Equal Area projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Cylindrical Equal Area projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Cylindrical Equal Area projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Cylindrical_Equal_Area_Coordinaes (State, Direction,
           &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.47.3 DECLARATIONS

5.47.3.1 TYPES

The structure type Cylindrical_Equal_Area_Tuple is used as defined in Section 4.1.2.

5.47.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.47.3.3 VARIABLES

Not applicable.

5.47.4 DEPENDENCIES

None.

5.47.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.48 SET_ECKERT4_PARAMS

5.48.1 DESCRIPTION

This function sets the current input or output Eckert IV projection parameters to the specified values.

5.48.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Eckert4_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Eckert4_Parameters  parameters);
```

State	Indicates whether the Eckert IV projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Eckert IV projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Eckert IV projection parameters (Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Eckert4_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.48.3 DECLARATIONS

5.48.3.1 TYPES

The structure type Eckert4_Parameters is used as defined in Section 4.1.2.

5.48.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.48.3.3 VARIABLES

Not applicable.

5.48.4 DEPENDENCIES

None.

5.48.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

```
ENGINE_INVALID_TYPE              : Invalid coordinate system type
```

5.49 GET_ECKERT4_PARAMS

5.49.1 DESCRIPTION

This function returns the current input or output Eckert IV projection parameters.

5.49.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Eckert4_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Eckert4_Parameters        *parameters);
```

State	Indicates whether the Eckert IV projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Eckert IV projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Eckert IV projection parameter structure (Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Eckert4_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

5.49.3 DECLARATIONS

5.49.3.1 TYPES

The structure type Eckert4_Parameters is used as defined in Section 4.1.2.

5.49.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.49.3.3 VARIABLES

Not applicable.

5.49.4 DEPENDENCIES

None.

5.49.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.50 SET_ECKERT4_COORDINATES

5.50.1 DESCRIPTION

This function sets the current input or output Eckert IV projection coordinates to the specified values.

5.50.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Eckert4_Coordinates (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Eckert4_Tuple       coordinates);
```

State	Indicates whether the Eckert IV projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Eckert IV projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Eckert IV projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Eckert4_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.50.3 DECLARATIONS

5.50.3.1 TYPES

The structure type Eckert4_Tuple is used as defined in Section 4.1.2.

5.50.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.50.3.3 VARIABLES

Not applicable.

5.50.4 DEPENDENCIES

None.

5.50.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.51 GET_ECKERT4_COORDINATES

5.51.1 DESCRIPTION

This function returns the current input or output Eckert IV projection coordinates.

5.51.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Eckert4_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Eckert4_Tuple             *coordinates);
```

State	Indicates whether the Eckert IV projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Eckert IV projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Eckert IV projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Eckert4_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive

Direction Input

coordinates (4000000.0, 4000000.0)

Outputs:

None.

5.51.3 DECLARATIONS

5.51.3.1 TYPES

The structure type Eckert4_Tuple is used as defined in Section 4.1.2.

5.51.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.51.3.3 VARIABLES

Not applicable.

5.51.4 DEPENDENCIES

None.

5.51.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.52 SET_ECKERT6_PARAMS

5.52.1 DESCRIPTION

This function sets the current input or output Eckert VI projection parameters to the specified values.

5.52.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Eckert6_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Eckert6_Parameters  parameters);
```

State	Indicates whether the Eckert VI projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Eckert VI projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Eckert VI projection parameters (Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Eckert6_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.52.3 DECLARATIONS

5.52.3.1 TYPES

The structure type Eckert6_Parameters is used as defined in Section 4.1.2.

5.52.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.52.3.3 VARIABLES

Not applicable.

5.52.4 DEPENDENCIES

None.

5.52.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.53 GET_ECKERT6_PARAMS

5.53.1 DESCRIPTION

This function returns the current input or output Eckert VI projection parameters.

5.53.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Eckert6_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
        Eckert6_Parameters        *parameters);
```

State	Indicates whether the Eckert VI projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Eckert VI projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Eckert VI projection parameter structure (Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Eckert6_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

5.53.3 DECLARATIONS

5.53.3.1 TYPES

The structure type Eckert6_Parameters is used as defined in Section 4.1.2.

5.53.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.53.3.3 VARIABLES

Not applicable.

5.53.4 DEPENDENCIES

None.

5.53.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.54 SET_ECKERT6_COORDINATES

5.54.1 DESCRIPTION

This function sets the current input or output Eckert VI projection coordinates to the specified values.

5.54.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Eckert6_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Eckert6_Tuple       coordinates);
```

State	Indicates whether the Eckert VI projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Eckert VI projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Eckert VI projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Eckert6_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive

Direction Input

coordinates (4000000.0, 4000000.0)

Outputs:

None.

5.54.3 DECLARATIONS

5.54.3.1 TYPES

The structure type Eckert6_Tuple is used as defined in Section 4.1.2.

5.54.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.54.3.3 VARIABLES

Not applicable.

5.54.4 DEPENDENCIES

None.

5.54.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.55 GET_ECKERT6_COORDINATES

5.55.1 DESCRIPTION

This function returns the current input or output Eckert VI projection coordinates.

5.55.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Eckert6_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Eckert6_Tuple             *coordinates);
```

State	Indicates whether the Eckert VI projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Eckert VI projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Eckert VI projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Eckert6_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.55.3 DECLARATIONS

5.55.3.1 TYPES

The structure type Eckert6_Tuple is used as defined in Section 4.1.2.

5.55.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.55.3.3 VARIABLES

Not applicable.

5.55.4 DEPENDENCIES

None.

5.55.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.56 SET_EQUIDISTANT_CYLINDRICAL_PARAMS

5.56.1 DESCRIPTION

This function sets the current input or output Equidistant Cylindrical projection parameters to the specified values.

5.56.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Equidistant_Cylindrical_Params (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Equidistant_Cylindrical_Parameters parameters);
```

State	Indicates whether the Equidistant Cylindrical projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Equidistant Cylindrical projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Equidistant Cylindrical projection parameters (Standard Parallel, Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Equidistant_Cylindrical_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.56.3 DECLARATIONS

5.56.3.1 TYPES

The structure type Equidistant_Cylindrical_Parameters is used as defined in Section 4.1.2.

5.56.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.56.3.3 VARIABLES

Not applicable.

5.56.4 DEPENDENCIES

None.

5.56.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.57 GET_EQUIDISTANT_CYLINDRICAL_PARAMS

5.57.1 DESCRIPTION

This function returns the current input or output Equidistant Cylindrical projection parameters.

5.57.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Equidistant_Cylindrical_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Equidistant_Cylindrical_Parameters *parameters);
```

State	Indicates whether the Equidistant Cylindrical projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Equidistant Cylindrical projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Equidistant Cylindrical projection parameter structure (Standard Parallel, Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Equidistant_Cylindrical_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

5.57.3 DECLARATIONS

5.57.3.1 TYPES

The structure type Equidistant_Cylindrical_Parameters is used as defined in Section 4.1.2.

5.57.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.57.3.3 VARIABLES

Not applicable.

5.57.4 DEPENDENCIES

None.

5.57.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.58 SET_EQUIDISTANT_CYLINDRICAL_COORDINATES

5.58.1 DESCRIPTION

This function sets the current input or output Equidistant Cylindrical projection coordinates to the specified values.

5.58.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Equidistant_Cylindrical_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Equidistant_Cylindrical_Tuple coordinates);
```

State	Indicates whether the Equidistant Cylindrical projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Equidistant Cylindrical projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Equidistant Cylindrical projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Equidistant_Cylindrical_Coordinates (State, Direction,
            coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.58.3 DECLARATIONS

5.58.3.1 TYPES

The structure type Equidistant_Cylindrical_Tuple is used as defined in Section 4.1.2.

5.58.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.58.3.3 VARIABLES

Not applicable.

5.58.4 DEPENDENCIES

None.

5.58.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.59 GET_EQUIDISTANT_CYLINDRICAL_COORDINATES

5.59.1 DESCRIPTION

This function returns the current input or output Equidistant Cylindrical projection coordinates.

5.59.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Equidistant_Cylindrical_Coordinates (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
Equidistant_Cylindrical_Tuple *coordinates);
```

State	Indicates whether the Equidistant Cylindrical projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Equidistant Cylindrical projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Equidistant Cylindrical projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Equidistant_Cylindrical_Coordinates (State, Direction,
           &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.59.3 DECLARATIONS

5.59.3.1 TYPES

The structure type Equidistant_Cylindrical_Tuple is used as defined in Section 4.1.2.

5.59.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.59.3.3 VARIABLES

Not applicable.

5.59.4 DEPENDENCIES

None.

5.59.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.60 SET_LAMBERT_CONFORMAL_CONIC_PARAMS

5.60.1 DESCRIPTION

This function sets the current input or output Lambert Conformal Conic projection parameters to the specified values.

5.60.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Lambert_Conformal_Conic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Lambert_Conformal_Conic_Parameters parameters);
```

State — Indicates whether the Lambert Conformal Conic projection parameters are to be set for interactive or file processing (input),

Direction — Indicates whether the Lambert Conformal Conic projection parameters are to be set for the input or output coordinate system or projection (input),

parameters — Structure containing Lambert Conformal Conic projection parameters (Origin Latitude, Central Meridian, 1st Standard

Parallel, 2nd Standard Parallel, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Lambert_Conformal_Conic_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 40.0*PI/180.0, 50.0*PI/180.0, 2000000.0, 5000000.0)

Outputs:

None.

5.60.3 DECLARATIONS

5.60.3.1 TYPES

The structure type Lambert_Conformal_Conic_Parameters is used as defined in Section 4.1.2.

5.60.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.60.3.3 VARIABLES

Not applicable.

5.60.4 DEPENDENCIES

None.

5.60.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.61 GET_LAMBERT_CONFORMAL_CONIC_PARAMS

5.61.1 DESCRIPTION

This function returns the current input or output Lambert Conformal Conic projection parameters.

5.61.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Lambert_Conformal_Conic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Lambert_Conformal_Conic_Parameters *parameters);
```

State	Indicates whether the Lambert Conformal Conic projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Lambert Conformal Conic projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Lambert Conformal Conic projection parameter structure (Origin Latitude, Central Meridian, 1st Standard Parallel, 2nd Standard Parallel, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Lambert_Conformal_Conic_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive

Direction Input

Outputs:

parameters (45.0*PI/180.0, 0.0*PI/180.0, 40.0*PI/180.0, 50.0*PI/180.0, 2000000.0, 5000000.0)

5.61.3 DECLARATIONS

5.61.3.1 TYPES

The structure type Lambert_Conformal_Conic_Parameters is used as defined in Section 4.1.2.

5.61.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.61.3.3 VARIABLES

Not applicable.

5.61.4 DEPENDENCIES

None.

5.61.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.62 SET_LAMBERT_CONFORMAL_CONIC_COORDINATES

5.62.1 DESCRIPTION

This function sets the current input or output Lambert Conformal Conic projection coordinates to the specified values.

5.62.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Lambert_Conformal_Conic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Lambert_Conformal_Conic_Tuple coordinates);
```

State	Indicates whether the Lambert Conformal Conic projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Lambert Conformal Conic projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Lambert Conformal Conic projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Lambert_Conformal_Conic_Coordinates (State, Direction,
            coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(2000000.0, 5000000.0)

Outputs:

None.

5.62.3 DECLARATIONS

5.62.3.1 TYPES

The structure type Lambert_Conformal_Conic_Tuple is used as defined in Section 4.1.2.

5.62.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.62.3.3 VARIABLES

Not applicable.

5.62.4 DEPENDENCIES

None.

5.62.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.63 GET_LAMBERT_CONFORMAL_CONIC_COORDINATES

5.63.1 DESCRIPTION

This function returns the current input or output Lambert Conformal Conic projection coordinates.

5.63.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Lambert_Conformal_Conic_Coordinates (
```

```
        const File_or_Interactive State,
        const Input_or_Output     Direction,
        Lambert_Conformal_Conic_Tuple *coordinates);
```

State	Indicates whether the Lambert Conformal Conic projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Lambert Conformal Conic projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Lambert Conformal Conic projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Lambert_Conformal_Conic_Coordinates (State, Direction,
           &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(2000000.0, 5000000.0)

Outputs:

None.

5.63.3 DECLARATIONS

5.63.3.1 TYPES

The structure type Lambert_Conformal_Conic_Tuple is used as defined in Section 4.1.2.

5.63.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.63.3.3 VARIABLES

Not applicable.

5.63.4 DEPENDENCIES

None.

5.63.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.64 SET_LOCAL_CARTESIAN_PARAMS

5.64.1 DESCRIPTION

This function sets the current input or output Local Cartesian coordinate system parameters to the specified values.

5.64.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Local_Cartesian_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Local_Cartesian_Parameters parameters);
```

State	Indicates whether the Local Cartesian coordinate system parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Local Cartesian coordinate system parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Local Cartesian coordinate system parameters (Origin Latitude, Origin Longitude, Origin Height, Orientation) to be set (input).

Example:

```
status = Set_Local_Cartesian_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 0.0, 0.0*PI/180.0)

Outputs:

None.

5.64.3 DECLARATIONS

5.64.3.1 TYPES

The structure type Local_Cartesian_Parameters is used as defined in Section 4.1.2.

5.64.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.64.3.3 VARIABLES

Not applicable.

5.64.4 DEPENDENCIES

None.

5.64.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

```
ENGINE_INVALID_TYPE              : Invalid coordinate system type
```

5.65 GET_LOCAL_CARTESIAN_PARAMS

5.65.1 DESCRIPTION

This function returns the current input or output Local Cartesian coordinate system parameters.

5.65.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Local_Cartesian_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Local_Cartesian_Parameters *parameters);
```

State	Indicates whether the Local Cartesian coordinate system parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Local Cartesian coordinate system parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Local Cartesian coordinate system parameter structure (Origin Latitude, Origin Longitude, Origin Height, Orientation) to be returned (output).

Example:

```
status = Get_Local_Cartesian_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 0.0, 0.0*PI/180.0)

5.65.3 DECLARATIONS

5.65.3.1 TYPES

The structure type Local_Cartesian_Parameters is used as defined in Section 4.1.2.

5.65.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.65.3.3 VARIABLES

Not applicable.

5.65.4 DEPENDENCIES

None.

5.65.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.66 SET_LOCAL_CARTESIAN_COORDINATES

5.66.1 DESCRIPTION

This function sets the current input or output Local Cartesian coordinates to the specified values.

5.66.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Local_Cartesian_Coordinates (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Local_Cartesian_Tuple coordinates);
```

State	Indicates whether the Local Cartesian coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Local Cartesian coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Local Cartesian coordinates (x, y, z) to be set (input).

Example:

```
status = Set_Local_Cartesian_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(100000.0, -100000.0, -200.0)

Outputs:

None.

5.66.3 DECLARATIONS

5.66.3.1 TYPES

The structure type Local_Cartesian_Tuple is used as defined in Section 4.1.2.

5.66.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.66.3.3 VARIABLES

Not applicable.

5.66.4 DEPENDENCIES

None.

5.66.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.67 GET_LOCAL_CARTESIAN_COORDINATES

5.67.1 DESCRIPTION

This function returns the current input or output Local Cartesian coordinates.

5.67.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Local_Cartesian_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Local_Cartesian_Tuple     *coordinates);
```

State	Indicates whether the Local Cartesian coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Local Cartesian coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Local Cartesian coordinate structure (x, y, z) to be returned (output).

Example:

```
status = Get_Local_Cartesian_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive

Direction Input

coordinates (100000.0, -100000.0, -200.0)

Outputs:

None.

5.67.3 DECLARATIONS

5.67.3.1 TYPES

The structure type Local_Cartesian_Tuple is used as defined in Section 4.1.2.

5.67.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.67.3.3 VARIABLES

Not applicable.

5.67.4 DEPENDENCIES

None.

5.67.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.68 SET_MERCATOR_PARAMS

5.68.1 DESCRIPTION

This function sets the current input or output Mercator projection parameters to the specified values.

5.68.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Mercator_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Mercator_Parameters parameters);
```

State	Indicates whether the Mercator projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Mercator projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Mercator projection parameters (Origin Latitude, Central Meridian, Scale Factor, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Mercator_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 1.0, 4000000.0, 4000000.0)

Outputs:

None.

5.68.3 DECLARATIONS

5.68.3.1 TYPES

The structure type Mercator_Parameters is used as defined in Section 4.1.2.

5.68.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.68.3.3 VARIABLES

Not applicable.

5.68.4 DEPENDENCIES

None.

5.68.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.69 GET_MERCATOR_PARAMS

5.69.1 DESCRIPTION

This function returns the current input or output Mercator projection parameters.

5.69.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Mercator_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
        Mercator_Parameters       *parameters);
```

State	Indicates whether the Mercator projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Mercator projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Mercator projection parameter structure (Origin Latitude, Central Meridian, Scake Factor, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Mercator_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 1.0, 4000000.0, 4000000.0)

5.69.3 DECLARATIONS

5.69.3.1 TYPES

The structure type Mercator_Parameters is used as defined in Section 4.1.2.

5.69.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.69.3.3 VARIABLES

Not applicable.

5.69.4 DEPENDENCIES

None.

5.69.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.70 SET_MERCATOR_COORDINATES

5.70.1 DESCRIPTION

This function sets the current input or output Mercator projection coordinates to the specified values.

5.70.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Mercator_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Mercator_Tuple      coordinates);
```

State	Indicates whether the Mercator projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Mercator projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Mercator projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Mercator_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive

Direction Input

coordinates (4000000.0, 4000000.0)

Outputs:

None.

5.70.3 DECLARATIONS

5.70.3.1 TYPES

The structure type Mercator_Tuple is used as defined in Section 4.1.2.

5.70.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.70.3.3 VARIABLES

Not applicable.

5.70.4 DEPENDENCIES

None.

5.70.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.71 GET_MERCATOR_COORDINATES

5.71.1 DESCRIPTION

This function returns the current input or output Mercator projection coordinates.

5.71.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Mercator_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Mercator_Tuple            *coordinates);
```

State	Indicates whether the Mercator projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Mercator projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Mercator projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Mercator_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.71.3 DECLARATIONS

5.71.3.1 TYPES

The structure type Mercator_Tuple is used as defined in Section 4.1.2.

5.71.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.71.3.3 VARIABLES

Not applicable.

5.71.4 DEPENDENCIES

None.

5.71.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.72 SET_MGRS_COORDINATES

5.72.1 DESCRIPTION

This function sets the current input or output MGRS coordinate string.

5.72.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_MGRS_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
      const MGRS_Tuple          coordinates);
```

State — Indicates whether the MGRS coordinates are to be set for interactive or file processing (input),

Direction — Indicates whether the MGRS coordinates are to be set for the input or output coordinate system or projection (input),

coordinates — Structure containing MGRS coordinates (MGRS string) to be set (input).

Example:

```
status = Set_MGRS_Coordinates (State, Direction, coordinates)
```

Inputs:

State — Interactive

Direction — Input

coordinates — (18TWA8781582302)

Outputs:

None.

5.72.3 DECLARATIONS

5.72.3.1 TYPES

The structure type MGRS_Tuple is used as defined in Section 4.1.2.

5.72.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.72.3.3 VARIABLES

Not applicable.

5.72.4 DEPENDENCIES

None.

5.72.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.73 GET_MGRS_COORDINATES

5.73.1 DESCRIPTION

This function returns the current input or output MGRS coordinate string.

5.73.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_MGRS_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            MGRS_Tuple                *coordinates);
```

State	Indicates whether the MGRS coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the MGRS coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	MGRS coordinate structure (MGRS string) to be returned (output).

Example:

```
status = Get_MGRS_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input

coordinates (18TWA8781582302)

Outputs:

None.

5.73.3 DECLARATIONS

5.73.3.1 TYPES

The structure type MGRS_Tuple is used as defined in Section 4.1.2.

5.73.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.73.3.3 VARIABLES

Not applicable.

5.73.4 DEPENDENCIES

None.

5.73.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.74 SET_MILLER_CYLINDRICAL_PARAMS

5.74.1 DESCRIPTION

This function sets the current input or output Miller Cylindrical projection parameters to the specified values.

5.74.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Miller_Cylindrical_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Miller_Cylindrical_Parameters parameters);
```

State	Indicates whether the Miller Cylindrical projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Miller Cylindrical projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Miller Cylindrical projection parameters (Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Miller_Cylindrical_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.74.3 DECLARATIONS

5.74.3.1 TYPES

The structure type Miller_Cylindrical_Parameters is used as defined in Section 4.1.2.

5.74.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.74.3.3 VARIABLES

Not applicable.

5.74.4 DEPENDENCIES

None.

5.74.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.75 GET_MILLER_CYLINDRICAL_PARAMS

5.75.1 DESCRIPTION

This function returns the current input or output Miller Cylindrical projection parameters.

5.75.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Miller_Cylindrical_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Miller_Cylindrical_Parameters       *parameters);
```

State	Indicates whether the Miller Cylindrical projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Miller Cylindrical projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Miller Cylindrical projection parameter structure (Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Miller_Cylindrical_Params (State, Direction, &parameters)
```

Inputs:

State Interactive

Direction Input

Outputs:

parameters (0.0*PI/180.0, 4000000.0, 4000000.0)

5.75.3 DECLARATIONS

5.75.3.1 TYPES

The structure type Miller_Cylindrical_Parameters is used as defined in Section 4.1.2.

5.75.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.75.3.3 VARIABLES

Not applicable.

5.75.4 DEPENDENCIES

None.

5.75.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.76 SET_MILLER_CYLINDRICAL_COORDINATES

5.76.1 DESCRIPTION

This function sets the current input or output Miller Cylindrical projection coordinates to the specified values.

5.76.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Miller_Cylindrical_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Miller_Cylindrical_Tuple coordinates);
```

State	Indicates whether the Miller Cylindrical projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Miller Cylindrical projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Miller Cylindrical projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Miller_Cylindrical_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.76.3 DECLARATIONS

5.76.3.1 TYPES

The structure type Miller_Cylindrical_Tuple is used as defined in Section 4.1.2.

5.76.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.76.3.3 VARIABLES

Not applicable.

5.76.4 DEPENDENCIES

None.

5.76.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.77 GET_MILLER_CYLINDRICAL_COORDINATES

5.77.1 DESCRIPTION

This function returns the current input or output Miller Cylindrical projection coordinates.

5.77.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Miller_Cylindrical_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
Miller_Cylindrical_Tuple  *coordinates);
```

State	Indicates whether the Miller Cylindrical projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Miller Cylindrical projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Miller Cylindrical projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Miller_Cylindrical_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.77.3 DECLARATIONS

5.77.3.1 TYPES

The structure type Miller_Cylindrical_Tuple is used as defined in Section 4.1.2.

5.77.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.77.3.3 VARIABLES

Not applicable.

5.77.4 DEPENDENCIES

None.

5.77.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.78 SET_MOLLWEIDE_PARAMS

5.78.1 DESCRIPTION

This function sets the current input or output Mollweide projection parameters to the specified values.

5.78.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Mollweide_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Mollweide_Parameters parameters);
```

State	Indicates whether the Mollweide projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Mollweide projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Mollweide projection parameters (Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Mollweide_Params (State, Direction, parameters)
```

Inputs:

State	Interactive

Direction Input

parameters (0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.78.3 DECLARATIONS

5.78.3.1 TYPES

The structure type Mollweide_Parameters is used as defined in Section 4.1.2.

5.78.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.78.3.3 VARIABLES

Not applicable.

5.78.4 DEPENDENCIES

None.

5.78.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.79 GET_MOLLWEIDE_PARAMS

5.79.1 DESCRIPTION

This function returns the current input or output Mollweide projection parameters.

5.79.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Mollweide_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Mollweide_Parameters        *parameters);
```

State	Indicates whether the Mollweide projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Mollweide projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Mollweide projection parameter structure (Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Mollweide_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

5.79.3 DECLARATIONS

5.79.3.1 TYPES

The structure type Mollweide_Parameters is used as defined in Section 4.1.2.

5.79.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.79.3.3 VARIABLES

Not applicable.

5.79.4 DEPENDENCIES

None.

5.79.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.80 SET_MOLLWEIDE_COORDINATES

5.80.1 DESCRIPTION

This function sets the current input or output Mollweide projection coordinates to the specified values.

5.80.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Mollweide_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Mollweide_Tuple coordinates);
```

State — Indicates whether the Mollweide projection coordinates are to be set for interactive or file processing (input),

Direction — Indicates whether the Mollweide projection coordinates are to be set for the input or output coordinate system or projection (input),

coordinates Structure containing Mollweide projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Mollweide_Coordinates (State, Direction, coordinates)
```

Inputs:

State Interactive

Direction Input

coordinates (4000000.0, 4000000.0)

Outputs:

None.

5.80.3 DECLARATIONS

5.80.3.1 TYPES

The structure type Mollweide_Tuple is used as defined in Section 4.1.2.

5.80.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.80.3.3 VARIABLES

Not applicable.

5.80.4 DEPENDENCIES

None.

5.80.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.81 GET_MOLLWEIDE_COORDINATES

5.81.1 DESCRIPTION

This function returns the current input or output Mollweide projection coordinates.

5.81.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Mollweide_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Mollweide_Tuple  *coordinates);
```

State	Indicates whether the Mollweide projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Mollweide projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Mollweide projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Mollweide_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.81.3 DECLARATIONS

5.81.3.1 TYPES

The structure type Mollweide_Tuple is used as defined in Section 4.1.2.

5.81.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.81.3.3 VARIABLES

Not applicable.

5.81.4 DEPENDENCIES

None.

5.81.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.82 SET_ORTHOGRAPHIC_PARAMS

5.82.1 DESCRIPTION

This function sets the current input or output Orthographic projection parameters to the specified values.

5.82.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Orthographic_Params (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Orthographic_Parameters parameters);
```

State	Indicates whether the Orthographic projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Orthographic projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Orthographic projection parameters (Origin Latitude, Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Orthographic_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.82.3 DECLARATIONS

5.82.3.1 TYPES

The structure type Orthographic_Parameters is used as defined in Section 4.1.2.

5.823.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.82.3.3 VARIABLES

Not applicable.

5.82.4 DEPENDENCIES

None.

5.82.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.83 GET_ORTHOGRAPHIC_PARAMS

5.83.1 DESCRIPTION

This function returns the current input or output Orthographic projection parameters.

5.83.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Orthographic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Orthographic_Parameters   *parameters);
```

State	Indicates whether the Orthographic projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Orthographic projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Orthographic projection parameter structure (Origin Latitude, Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Orthographic_Params (State, Direction, &parameters)
```

Inputs:

State Interactive

Direction Input

Outputs:

parameters (45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

5.83.3 DECLARATIONS

5.83.3.1 TYPES

The structure type Orthographic_Parameters is used as defined in Section 4.1.2.

5.83.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.83.3.3 VARIABLES

Not applicable.

5.83.4 DEPENDENCIES

None.

5.83.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.84 SET_ORTHOGRAPHIC_COORDINATES

5.84.1 DESCRIPTION

This function sets the current input or output Orthographic projection coordinates to the specified values.

5.84.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Orthographic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Orthographic_Tuple  coordinates);
```

State	Indicates whether the Orthographic projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Orthographic projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Orthographic projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Orthographic_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.84.3 DECLARATIONS

5.84.3.1 TYPES

The structure type Orthographic_Tuple is used as defined in Section 4.1.2.

5.84.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.84.3.3 VARIABLES

Not applicable.

5.84.4 DEPENDENCIES

None.

5.84.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.85 GET_ORTHOGRAPHIC_COORDINATES

5.85.1 DESCRIPTION

This function returns the current input or output Orthographic projection coordinates.

5.85.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Orthographic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
      Orthographic_Tuple        *coordinates);
```

State	Indicates whether the Orthographic projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Orthographic projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Orthographic projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Orthographic_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.85.3 DECLARATIONS

5.85.3.1 TYPES

The structure type Orthographic_Tuple is used as defined in Section 4.1.2.

5.85.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.85.3.3 VARIABLES

Not applicable.

5.85.4 DEPENDENCIES

None.

5.85.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.86 SET_POLAR_STEREO_PARAMS

5.86.1 DESCRIPTION

This function sets the current input or output Polar Stereographic projection parameters to the specified values.

5.86.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Polar_Stereo_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Polar_Stereo_Parameters parameters);
```

State	Indicates whether the Polar Stereographic projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Polar Stereographic projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Polar Stereographic projection parameters (Latitude of True Scale, Longitude Down from Pole, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Polar_Stereo_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(90.0*PI/180.0, 0.0*PI/180.0, 400000.0, 400000.0)

Outputs:

None.

5.86.3 DECLARATIONS

5.86.3.1 TYPES

The structure type Polar_Stereo_Parameters is used as defined in Section 4.1.2.

5.86.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.86.3.3 VARIABLES

Not applicable.

5.86.4 DEPENDENCIES

None.

5.86.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.87 GET_POLAR_STEREO_PARAMS

5.87.1 DESCRIPTION

This function returns the current input or output Polar Stereographic projection parameters.

5.87.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Polar_Stereo_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Polar_Stereo_Parameters   *parameters);
```

State	Indicates whether the Polar Stereographic projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Polar Stereographic projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Polar Stereographic projection parameter structure (Latitude of True Scale, Longitude Down from Pole, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Polar_Stereo_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(90.0*PI/180.0, 0.0*PI/180.0, 400000.0, 400000.0)

5.87.3 DECLARATIONS

5.87.3.1 TYPES

The structure type Polar_Stereo_Parameters is used as defined in Section 4.1.2.

5.87.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.87.3.3 VARIABLES

Not applicable.

5.87.4 DEPENDENCIES

None.

5.87.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.88 SET_POLAR_STEREO_COORDINATES

5.88.1 DESCRIPTION

This function sets the current input or output Polar Stereographic projection coordinates to the specified values.

5.88.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Polar_Stereo_Coordinates (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Polar_Stereo_Tuple coordinates);
```

State	Indicates whether the Polar Stereographic projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Polar Stereographic projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Polar Stereographic projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Polar_Stereo_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.88.3 DECLARATIONS

5.88.3.1 TYPES

The structure type Polar_Stereo_Tuple is used as defined in Section 4.1.2.

5.88.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.88.3.3 VARIABLES

Not applicable.

5.88.4 DEPENDENCIES

None.

5.88.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.89 GET_POLAR_STEREO_COORDINATES

5.89.1 DESCRIPTION

This function returns the current input or output Polar Stereographic projection coordinates.

5.89.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Polar_Stereo_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Polar_Stereo_Tuple        *coordinates);
```

State	Indicates whether the Polar Stereographic projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Polar Stereographic projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Polar Stereographic projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Polar_Stereo_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.89.3 DECLARATIONS

5.89.3.1 TYPES

The structure type Polar_Stereo_Tuple is used as defined in Section 4.1.2.

5.89.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.89.3.3 VARIABLES

Not applicable.

5.89.4 DEPENDENCIES

None.

5.89.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.90 SET_POLYCONIC_PARAMS

5.90.1 DESCRIPTION

This function sets the current input or output Polyconic projection parameters to the specified values.

5.90.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Polyconic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Polyconic_Parameters parameters);
```

State	Indicates whether the Polyconic projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Polyconic projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Polyconic projection parameters (Origin Latitude, Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Polyconic_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.90.3 DECLARATIONS

5.90.3.1 TYPES

The structure type Polyconic_Parameters is used as defined in Section 4.1.2.

5.90.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.90.3.3 VARIABLES

Not applicable.

5.90.4 DEPENDENCIES

None.

5.90.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.91 GET_POLYCONIC_PARAMS

5.91.1 DESCRIPTION

This function returns the current input or output Polyconic projection parameters.

5.91.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Polyconic_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
        Polyconic_Parameters      *parameters);
```

State	Indicates whether the Polyconic projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Polyconic projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Polyconic projection parameter structure (Origin Latitude, Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Polyconic_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 4000000.0, 4000000.0)

5.91.3 DECLARATIONS

5.91.3.1 TYPES

The structure type Polyconic_Parameters is used as defined in Section 4.1.2.

5.91.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.91.3.3 VARIABLES

Not applicable.

5.91.4 DEPENDENCIES

None.

5.91.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.92 SET_POLYCONIC_COORDINATES

5.92.1 DESCRIPTION

This function sets the current input or output Polyconic projection coordinates to the specified values.

5.92.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Polyconic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Polyconic_Tuple     coordinates);
```

State	Indicates whether the Polyconic projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Polyconic projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Polyconic projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Polyconic_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive

Direction Input

coordinates (4000000.0, 4000000.0)

Outputs:

None.

5.92.3 DECLARATIONS

5.92.3.1 TYPES

The structure type Polyconic_Tuple is used as defined in Section 4.1.2.

5.92.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.92.3.3 VARIABLES

Not applicable.

5.92.4 DEPENDENCIES

None.

5.92.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.93 GET_POLYCONIC_COORDINATES

5.93.1 DESCRIPTION

This function returns the current input or output Polyconic projection coordinates.

5.93.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Polyconic_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Polyconic_Tuple           *coordinates);
```

State	Indicates whether the Polyconic projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Polyconic projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Polyconic projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Polyconic_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.93.3 DECLARATIONS

5.93.3.1 TYPES

The structure type Polyconic_Tuple is used as defined in Section 4.1.2.

5.93.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.93.3.3 VARIABLES

Not applicable.

5.93.4 DEPENDENCIES

None.

5.93.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.94 SET_SINUSOIDAL_PARAMS

5.94.1 DESCRIPTION

This function sets the current input or output Sinusoidal projection parameters to the specified values.

5.94.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Sinusoidal_Params (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const Sinusoidal_Parameters parameters);
```

State	Indicates whether the Sinusoidal projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Sinusoidal projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Sinusoidal projection parameters (Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Sinusoidal_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.94.3 DECLARATIONS

5.94.3.1 TYPES

The structure type Sinusoidal_Parameters is used as defined in Section 4.1.2.

5.94.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.94.3.3 VARIABLES

Not applicable.

5.94.4 DEPENDENCIES

None.

5.94.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.95 GET_SINUSOIDAL_PARAMS

5.95.1 DESCRIPTION

This function returns the current input or output Sinusoidal projection parameters.

5.95.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Sinusoidal_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Sinusoidal_Parameters     *parameters);
```

State	Indicates whether the Sinusoidal projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Sinusoidal projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Sinusoidal projection parameter structure (Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Sinusoidal_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive

Direction Input

Outputs:

parameters (0.0*PI/180.0, 4000000.0, 4000000.0)

5.95.3 DECLARATIONS

5.95.3.1 TYPES

The structure type Sinusoidal_Parameters is used as defined in Section 4.1.2.

5.95.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.95.3.3 VARIABLES

Not applicable.

5.95.4 DEPENDENCIES

None.

5.95.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.96 SET_SINUSOIDAL_COORDINATES

5.96.1 DESCRIPTION

This function sets the current input or output Sinusoidal projection coordinates to the specified values.

5.96.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Sinusoidal_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Sinusoidal_Tuple    coordinates);
```

State	Indicates whether the Sinusoidal projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Sinusoidal projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Sinusoidal projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Sinusoidal_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.96.3 DECLARATIONS

5.96.3.1 TYPES

The structure type Sinusoidal_Tuple is used as defined in Section 4.1.2.

5.96.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.96.3.3 VARIABLES

Not applicable.

5.96.4 DEPENDENCIES

None.

5.96.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.97 GET_SINUSOIDAL_COORDINATES

5.97.1 DESCRIPTION

This function returns the current input or output Sinusoidal projection coordinates.

5.97.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Sinusoidal_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Sinusoidal_Tuple          *coordinates);
```

State	Indicates whether the Sinusoidal projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Sinusoidal projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Sinusoidal projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Sinusoidal_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.97.3 DECLARATIONS

5.97.3.1 TYPES

The structure type Sinusoidal_Tuple is used as defined in Section 4.1.2.

5.97.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.97.3.3 VARIABLES

Not applicable.

5.97.4 DEPENDENCIES

None.

5.97.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

```
ENGINE_INVALID_TYPE              : Invalid coordinate system type
```

5.98 SET_TRANSVERSE_CYLINDRICAL_EQUAL_AREA_PARAMS

5.98.1 DESCRIPTION

This function sets the current input or output Transverse Cylindrical Equal Area projection parameters to the specified values.

5.98.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Transverse_Cylindrical_Equal_Area_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Transverse_Cylindrical_Equal_Area_Parameters parameters);
```

State	Indicates whether the Transverse Cylindrical Equal Area projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Transverse Cylindrical Equal Area projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Transverse Cylindrical Equal Area projection parameters (Origin Latitude, Central Meridian, Scale Factor, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Transverse_Cylindrical_Equal_Area_Params (State, Direction,
            parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 1.0, 4000000.0, 4000000.0)

Outputs:

None.

5.98.3 DECLARATIONS

5.98.3.1 TYPES

The structure type Transverse_Cylindrical_Equal_Area_Parameters is used as defined in Section 4.1.2.

5.98.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.98.3.3 VARIABLES

Not applicable.

5.98.4 DEPENDENCIES

None.

5.98.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.99 GET_TRANSVERSE_CYLINDRICAL_EQUAL_AREA_PARAMS

5.99.1 DESCRIPTION

This function returns the current input or output Transverse Cylindrical Equal Area projection parameters.

5.99.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Transverse_Cylindrical_Equal_Area_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Transverse_Cylindrical_Equal_Area_Parameters *parameters);
```

State	Indicates whether the Transverse Cylindrical Equal Area projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Transverse Cylindrical Equal Area projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Transverse Cylindrical Equal Area projection parameter structure (Origin Latitude, Central Meridian, Scake Factor, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Transverse_Cylindrical_Equal_Area_Params (State, Direction,
            &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 1.0, 4000000.0, 4000000.0)

5.99.3 DECLARATIONS

5.99.3.1 TYPES

The structure type Transverse_Cylindrical_Equal_Area_Parameters is used as defined in Section 4.1.2.

5.99.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.99.3.3 VARIABLES

Not applicable.

5.99.4 DEPENDENCIES

None.

5.99.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.100 SET_TRANSVERSE_CYLINDRICAL_EQUAL_AREA_COORDINATES

5.100.1 DESCRIPTION

This function sets the current input or output Transverse Cylindrical Equal Area projection coordinates to the specified values.

5.100.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Transverse_Cylindrical_Equal_Area_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Transverse_Cylindrical_Equal_Area_Tuple coordinates);
```

State	Indicates whether the Transverse Cylindrical Equal Area projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Transverse Cylindrical Equal Area projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Transverse Cylindrical Equal Area projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Transverse_Cylindrical_Equal_Area_Coordinates (State, Direction,
            coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.1000.3 DECLARATIONS

5.100.3.1 TYPES

The structure type Transverse_Cylindrical_Equal_Area_Tuple is used as defined in Section 4.1.2.

5.100.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.100.3.3 VARIABLES

Not applicable.

5.100.4 DEPENDENCIES

None.

5.100.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
```

```
ENGINE_INVALID_DIRECTION         : Invalid direction parameter value
ENGINE_INVALID_STATE             : Invalid state parameter value
ENGINE_INVALID_TYPE              : Invalid coordinate system type
```

5.101 GET_TRANSVERSE_CYLINDRICAL_EQUAL_AREA_COORDINATES

5.101.1 DESCRIPTION

This function returns the current input or output Transverse Cylindrical Equal Area projection coordinates.

5.101.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Transverse_Cylindrical_Equal_Area_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Transverse_Cylindrical_Equal_Area_Tuple *coordinates);
```

State	Indicates whether the Transverse Cylindrical Equal Area projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Transverse Cylindrical Equal Area projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Transverse Cylindrical Equal Area projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Transverse_Cylindrical_Equal_Area_Coordinates (State, Direction,
            &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.101.3 DECLARATIONS

5.101.3.1 TYPES

The structure type Transverse_Cylindrical_Equal_Area_Tuple is used as defined in Section 4.1.2.

5.101.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.101.3.3 VARIABLES

Not applicable.

5.101.4 DEPENDENCIES

None.

5.101.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.102 SET_TRANSVERSE_MERCATOR_PARAMS

5.102.1 DESCRIPTION

This function sets the current input or output Transverse Mercator projection parameters to the specified values.

5.102.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Transverse_Mercator_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Transverse_Mercator_Parameters parameters);
```

State	Indicates whether the Transverse Mercator projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Transverse Mercator projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Transverse Mercator projection parameters (Origin Latitude, Central Meridian, Scale Factor, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Transverse_Mercator_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(45.0*PI/180.0, 0.0*PI/180.0, 1.0, 4000000.0, 4000000.0)

Outputs:

None.

5.102.3 DECLARATIONS

5.102.3.1 TYPES

The structure type Transverse_Mercator_Parameters is used as defined in Section 4.1.2.

5.102.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.102.3.3 VARIABLES

Not applicable.

5.102.4 DEPENDENCIES

None.

5.102.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.103 GET_TRANSVERSE_MERCATOR_PARAMS

5.103.1 DESCRIPTION

This function returns the current input or output Transverse Mercator projection parameters.

5.103.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Transverse_Mercator_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Transverse_Mercator_Parameters *parameters);
```

State — Indicates whether the Transverse Mercator projection parameters are to be returned for interactive or file processing (input),

Direction	Indicates whether the Transverse Mercator projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Transverse Mercator projection parameter structure (Origin Latitude, Central Meridian, Scake Factor, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Transverse_Mercator_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(45.0*PI/180.0, 0.0*PI/180.0, 1.0, 4000000.0, 4000000.0)

5.103.3 DECLARATIONS

5.103.3.1 TYPES

The structure type Transverse_Mercator_Parameters is used as defined in Section 4.1.2.

5.103.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.103.3.3 VARIABLES

Not applicable.

5.103.4 DEPENDENCIES

None.

5.103.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.104 SET_TRANSVERSE_MERCATOR_COORDINATES

5.104.1 DESCRIPTION

This function sets the current input or output Transverse Mercator projection coordinates to the specified values.

5.104.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Transverse_Mercator_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Transverse_Mercator_Tuple coordinates);
```

State	Indicates whether the Transverse Mercator projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Transverse Mercator projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Transverse Mercator projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Transverse_Mercator_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.104.3 DECLARATIONS

5.104.3.1 TYPES

The structure type Transverse_Mercator_Tuple is used as defined in Section 4.1.2.

5.104.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.104.3.3 VARIABLES

Not applicable.

5.104.4 DEPENDENCIES

None.

5.104.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.105 GET_TRANSVERSE_MERCATOR_COORDINATES

5.105.1 DESCRIPTION

This function returns the current input or output Transverse Mercator projection coordinates.

5.105.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Transverse_Mercator_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Transverse_Mercator_Tuple *coordinates);
```

State	Indicates whether the Transverse Mercator projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Transverse Mercator projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Transverse Mercator projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Transverse_Mercator_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.105.3 DECLARATIONS

5.105.3.1 TYPES

The structure type Transverse_Mercator_Tuple is used as defined in Section 4.1.2.

5.105.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.105.3.3 VARIABLES

Not applicable.

5.105.4 DEPENDENCIES

None.

5.105.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.106 SET_UPS_COORDINATES

5.106.1 DESCRIPTION

This function sets the current input or output Universal Polar Stereographic (UPS) coordinates to the specified values.

5.106.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_UPS_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const UPS_Tuple           coordinates);
```

State	Indicates whether the UPS coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the UPS coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing UPS coordinates (Easting, Northing, Hemisphere) to be set (input).

Example:

```
status = Set_UPS_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(2000000.0, 2000000.0, ‘N’)

Outputs:

None.

5.106.3 DECLARATIONS

5.106.3.1 TYPES

The structure type UPS_Tuple is used as defined in Section 4.1.2.

5.106.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.106.3.3 VARIABLES

Not applicable.

5.106.4 DEPENDENCIES

None.

5.106.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

```
ENGINE_INVALID_TYPE              : Invalid coordinate system type
```

5.107 GET_UPS_COORDINATES

5.107.1 DESCRIPTION

This function returns the current input or output Universal Polar Stereographic (UPS) coordinates.

5.107.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_UPS_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            UPS_Tuple                 *coordinates);
```

State	Indicates whether the UPS coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the UPS coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	UPS coordinate structure (Easting, Northing, Hemisphere) to be returned (output).

Example:

```
status = Get_UPS_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(2000000.0, 2000000.0, ‘N’)

Outputs:

None.

5.107.3 DECLARATIONS

5.107.3.1 TYPES

The structure type UPS_Tuple is used as defined in Section 4.1.2.

5.107.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.107.3.3 VARIABLES

Not applicable.

5.107.4 DEPENDENCIES

None.

5.107.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.108 SET_UTM_PARAMS

5.108.1 DESCRIPTION

This function sets the current input or output Universal Transverse Mercator (UTM) parameters to the specified values.

5.108.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_UTM_Params (
```

```
const File_or_Interactive State,
const Input_or_Output     Direction,
const UTM_Parameters      parameters);
```

State	Indicates whether the UTM parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the UTM parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing UTM parameters (Zone and Zone Override) to be set (input).

Example:

```
status = Set_UTM_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(31, 0)

Outputs:

None.

5.108.3 DECLARATIONS

5.108.3.1 TYPES

The structure type UTM_Parameters is used as defined in Section 4.1.2.

5.108.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.108.3.3 VARIABLES

Not applicable.

5.108.4 DEPENDENCIES

None.

5.108.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.109 GET_UTM_PARAMS

5.109.1 DESCRIPTION

This function returns the current input or output Universal Transverse Mercator (UTM) parameters.

5.109.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_UTM_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            UTM_Parameters            *parameters);
```

State	Indicates whether the UTM parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the UTM parameters are to be returned for the input or output coordinate system or projection (input),
parameters	UTM parameter structure (Zone and Zone Override) to be returned (output).

Example:

```
status = Get_UTM_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive

Direction Input

Outputs:

parameters (31, 0)

5.109.3 DECLARATIONS

5.109.3.1 TYPES

The structure type UTM_Parameters is used as defined in Section 4.1.2.

5.109.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.109.3.3 VARIABLES

Not applicable.

5.109.4 DEPENDENCIES

None.

5.109.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.110 SET_UTM_COORDINATES

5.110.1 DESCRIPTION

This function sets the current input or output Universal Transverse Mercator (UTM) coordinates to the specified values.

5.110.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_UTM_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const UTM_Tuple           coordinates);
```

State	Indicates whether the UTM coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the UTM coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing UTM coordinates (Easting, Northing, Zone, Hemisphere) to be set (input).

Example:

```
status = Set_UTM_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(500000.0, 4000000.0, 31, ‘N’)

Outputs:

None.

5.110.3 DECLARATIONS

5.110.3.1 TYPES

The structure type UTM_Tuple is used as defined in Section 4.1.2.

5.110.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.110.3.3 VARIABLES

Not applicable.

5.110.4 DEPENDENCIES

None.

5.110.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.111 GET_UTM_COORDINATES

5.111.1 DESCRIPTION

This function returns the current input or output Universal Transverse Mercator (UTM) coordinates.

5.111.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_UTM_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            UTM_Tuple                 *coordinates);
```

State	Indicates whether the UTM coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the UTM coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	UTM coordinate structure (Easting, Northing, Zone, Hemisphere) to be returned (output).

Example:

```
status = Get_Transverse_Mercator_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(500000.0, 4000000.0, 31, ‘N’)

Outputs:

None.

5.111.3 DECLARATIONS

5.111.3.1 TYPES

The structure type UTM_Tuple is used as defined in Section 4.1.2.

5.111.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.111.3.3 VARIABLES

Not applicable.

5.111.4 DEPENDENCIES

None.

5.111.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
```

```
ENGINE_INVALID_TYPE              : Invalid coordinate system type
```

5.112 SET_VAN_DER_GRINTEN_PARAMS

5.112.1 DESCRIPTION

This function sets the current input or output Van der Grinten projection parameters to the specified values.

5.112.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Van_der_Grinten_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Van_der_Grinten_Parameters parameters);
```

State	Indicates whether the Van der Grinten projection parameters are to be set for interactive or file processing (input),
Direction	Indicates whether the Van der Grinten projection parameters are to be set for the input or output coordinate system or projection (input),
parameters	Structure containing Van der Grinten projection parameters (Central Meridian, False Easting, False Northing) to be set (input).

Example:

```
status = Set_Van_der_Grinten_Params (State, Direction, parameters)
```

Inputs:

State	Interactive
Direction	Input
parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

Outputs:

None.

5.112.3 DECLARATIONS

5.112.3.1 TYPES

The structure type Van_der_Grinten_Parameters is used as defined in Section 4.1.2.

5.112.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.112.3.3 VARIABLES

Not applicable.

5.112.4 DEPENDENCIES

None.

5.112.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.113 GET_VAN_DER_GRINTEN_PARAMS

5.113.1 DESCRIPTION

This function returns the current input or output Van der Grinten projection parameters.

5.113.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Van_der_Grinten_Params (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
```

```
Van_der_Grinten_Parameters *parameters);
```

State	Indicates whether the Van der Grinten projection parameters are to be returned for interactive or file processing (input),
Direction	Indicates whether the Van der Grinten projection parameters are to be returned for the input or output coordinate system or projection (input),
parameters	Van der Grinten projection parameter structure (Central Meridian, False Easting, False Northing) to be returned (output).

Example:

```
status = Get_Van_der_Grinten_Params (State, Direction, &parameters)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

parameters	(0.0*PI/180.0, 4000000.0, 4000000.0)

5.113.3 DECLARATIONS

5.113.3.1 TYPES

The structure type Van_der_Grinten_Parameters is used as defined in Section 4.1.2.

5.113.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.113.3.3 VARIABLES

Not applicable.

5.113.4 DEPENDENCIES

None.

5.113.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.114 SET_VAN_DER_GRINTEN_COORDINATES

5.114.1 DESCRIPTION

This function sets the current input or output Van der Grinten projection coordinates to the specified values.

5.114.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Set_Van_der_Grinten_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            const Sinusoidal_Tuple    coordinates);
```

State	Indicates whether the Van der Grinten projection coordinates are to be set for interactive or file processing (input),
Direction	Indicates whether the Van der Grinten projection coordinates are to be set for the input or output coordinate system or projection (input),
coordinates	Structure containing Van der Grinten projection coordinates (Easting, Northing) to be set (input).

Example:

```
status = Set_Van_der_Grinten_Coordinates (State, Direction, coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.114.3 DECLARATIONS

5.114.3.1 TYPES

The structure type Van_der_Grinten_Tuple is used as defined in Section 4.1.2.

5.114.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.114.3.3 VARIABLES

Not applicable.

5.114.4 DEPENDENCIES

None.

5.114.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.115 GET_VAN_DER_GRINTEN_COORDINATES

5.115.1 DESCRIPTION

This function returns the current input or output Van der Grinten projection coordinates.

5.115.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Van_der_Grinten_Coordinates (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            Van_der_Grinten_Tuple     *coordinates);
```

State	Indicates whether the Van der Grinten projection coordinates are to be returned for interactive or file processing (input),
Direction	Indicates whether the Van der Grinten projection coordinates are to be returned for the input or output coordinate system or projection (input),
coordinates	Van der Grinten projection coordinate structure (Easting, Northing) to be returned (output).

Example:

```
status = Get_Van_der_Grinten_Coordinates (State, Direction, &coordinates)
```

Inputs:

State	Interactive
Direction	Input
coordinates	(4000000.0, 4000000.0)

Outputs:

None.

5.115.3 DECLARATIONS

5.115.3.1 TYPES

The structure type Van_der_Grinten_Tuple is used as defined in Section 4.1.2.

5.115.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive and Input_or_Output are used as defined in Section 4.1.1.

5.115.3.3 VARIABLES

Not applicable.

5.115.4 DEPENDENCIES

None.

5.115.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_TYPE             : Invalid coordinate system type
```

5.116 CONVERT

5.116.1 DESCRIPTION

This function converts the current input coordinates, according to the current input datum, coordinate system, and parameters, and the current output datum, coordinate system, and parameters.

5.116.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Convert (
            const File_or_Interactive State);
```

State — Indicates whether the interactive or file processing state information is to be used (input).

Example:

```
status = Convert (State)
```

Inputs:

State — Interactive

Outputs:

None.

5.116.3 DECLARATIONS

5.116.3.1 TYPES

Not applicable.

5.116.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive are used as defined in Section 4.1.1.

5.116.3.3 VARIABLES

Not applicable.

5.116.4 DEPENDENCIES

This function depends on all of the functions in the ALBERS EQUAL AREA CONIC, BONNE, CASSINI, CYLINDRICAL EQUAL AREA, ECKERT IV, ECKERT VI, EQUIDISTANT CYLINDRICAL, GEOCENTRIC, GEOREF, LAMBERT CONFORMAL CONIC, LOCAL CARTESIAN, MERCATOR, MGRS, MILLER CYLINDRICAL, MOLLWEIDE, ORTHOGRAPHIC, POLYCONIC, SINUSOIDAL, TRANSVERSE CYLINDRICAL EQUAL AREA, TRANSVERSE MERCATOR, UPS, UTM, and VAN DER GRINTEN components, as well as on most of the functions in the DATUM, ELLIPSOID, and GEOID components.

5.116.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INPUT_WARNING            : Warning returned by 1st conversion
ENGINE_INPUT_ERROR              : Error returned by 1st conversion
ENGINE_OUTPUT_WARNING           : Warning returned by 2nd conversion
ENGINE_OUTPUT_ERROR             : Error returned by 2nd conversion
```

5.117 GET_CONVERSION_ERRORS

5.117.1 DESCRIPTION

This function returns the 90% horizontal (circular), vertical (linear), and spherical error values for the most recent conversion.

A value of -1.0 indicates that the standard error value is unknown.

5.117.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Conversion_Errors (
            const File_or_Interactive State,
            double *CE90,
            double *LE90,
            double *SE90);
```

State	Indicates whether the interactive or file processing conversion error values are to be returned (input),
CE90	90% circular (horizontal) error value, in meters,
LE90	90% linear (vertical) error value, in meters,
SE90	90% spherical error value, in meters.

Example:

```
status = Get_Conversion_Errors (State, &CE90, &LE90, &SE90)
```

Inputs:

State	Interactive

Outputs:

CE90	5.0
LE90	3.0
SE90	5.0

5.117.3 DECLARATIONS

5.117.3.1 TYPES

Not applicable.

5.117.3.2 CONSTANTS

The enumerated constants of the types File_or_Interactive are used as defined in Section 4.1.1.

5.117.3.3 VARIABLES

Not applicable.

5.117.4 DEPENDENCIES

None.

5.117.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_STATE            : Invalid state parameter value
```

5.118 GET_CONVERSION_STATUS

5.118.1 DESCRIPTION

This function returns the current input or output conversion status. The conversion status is the logical or of one or more Engine conversion status codes.

5.118.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Conversion_Status (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            long                      *Conversion_Status);
```

State	Indicates whether the interactive or file processing conversion status is to be returned (input),
Direction	Indicates whether the input or output conversion status is to be returned (input),
Conversion_Status	Conversion status code(s) for the most recent conversion operation (output).

Example:

```
status = Get_Conversion_Status (State, Direction, Conversion_Status)
```

Inputs:

State	Interactive
Direction	Input

Outputs:

Conversion_Status	ENGINE_LON_ERROR

5.118.3 DECLARATIONS

5.118.3.1 TYPES

Not applicable.

5.118.3.2 CONSTANTS

Not applicable.

5.118.3.3 VARIABLES

Not applicable.

5.118.4 DEPENDENCIES

None.

5.118.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
```

5.119 GET_CONVERSION_STATUS_STRING

5.119.1 DESCRIPTION

This function returns a character string that corresponds to the current input or output conversion status.

5.119.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Conversion_Status_String (
            const File_or_Interactive State,
            const Input_or_Output     Direction,
            char                      *Separator,
            char                      *Message_String);
```

State — Indicates whether the interactive or file processing conversion status is to be returned (input),

Direction — Indicates whether the input or output conversion status is to be returned (input),

Separator	String to be used to separate individual status messages (input),
Message_String	String containing one or more concatenated error messages, separated by the specified string (output).

Example:

```
status = Get_Conversion_Status_String (State, Direction, Separator,
            Message_String)
```

Inputs:

State	Interactive
Direction	Input
Separator	"; "

Outputs:

Message_String	"Input MGRS Coordinates: Invalid MGRS String"

5.119.3 DECLARATIONS

5.119.3.1 TYPES

Not applicable.

5.119.3.2 CONSTANTS

Not applicable.

5.119.3.3 VARIABLES

Not applicable.

5.119.4 DEPENDENCIES

None.

5.119.5 ERROR HANDLING

This function returns the following status codes:

```
ENGINE_NO_ERROR                 : No errors occurred in function
ENGINE_NOT_INITIALIZED          : Initialize_Engine has not been called
ENGINE_INVALID_STATE            : Invalid state parameter value
ENGINE_INVALID_DIRECTION        : Invalid direction parameter value
```

5.120 GET_RETURN_CODE_STRING

5.120.1 DESCRIPTION

This function returns a character string that corresponds to a specified engine return code.

5.120.2 INTERFACES AND EXAMPLES

The following is a list of the formal arguments required to use this function.

```
long Get_Return_Code_String (
            long Error_Code,
            char *Separator,
            char *Message_String);
```

Error_Code	Code returned by previous call (input),
Separator	String to be used to separate individual status messages (input),
Message_String	String containing one or more concatenated error messages, separated by the specified string (output).

Example:

```
status = Get_Return_Code_String (Error_Code, Separator, Message_String)
```

Inputs:

Error_Code	ENGINE_INVALID_TYPE
Separator	"; "

Outputs:

Message_String	"Invalid Coordinate System Type"

5.120.3 DECLARATIONS

5.120.3.1 TYPES

Not applicable.

5.120.3.2 CONSTANTS

Not applicable.

5.120.3.3 VARIABLES

Not applicable.

5.120.4 DEPENDENCIES

None.

5.120.5 ERROR HANDLING

None.

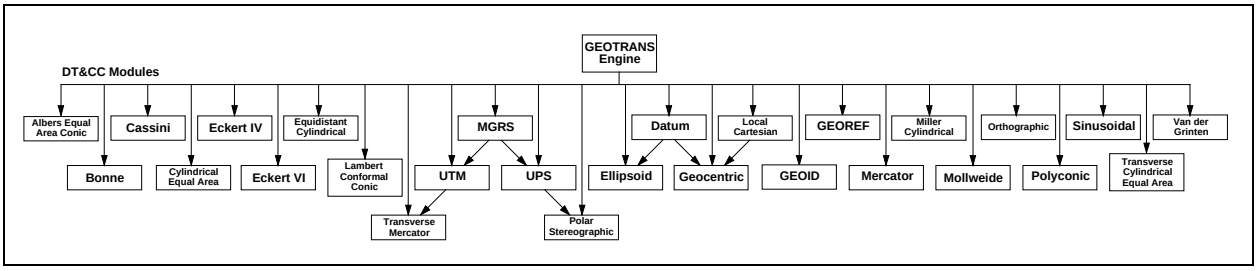
APPENDIX A STRUCTURE/DEPENDENCY DIAGRAMS

This component depends on a collection of other components that perform individual coordinate conversions, map projections, and datum transformations, including:

- ALBERS – which performs forward and inverse Albers Equal Area Conic projections,
- BONNE – which performs forward and inverse Bonne projections,
- CASSINI – which performs forward and inverse Cassini projections,
- CYLINDRICAL EQUAL AREA – which performs forward and inverse Cylindrical Equal Area projections,
- DATUM – which performs datum transformations between WGS 84, WGS 72, and all standard local datums,
- ECKERT4 – which performs forward and inverse Eckert IV projections,
- ECKERT6 – which performs forward and inverse Eckert VI projections,
- ELLIPSOID – which provides parameter values for all standard ellipsoids,
- EQUIDISTANT CYLINDRICAL – which performs forward and inverse Equidistant Cylindrical projections,
- GEOID – which performs conversions between WGS 84 ellipsoid heights and geoid heights,
- GEOREF – which performs conversions between geodetic and World Geodetic Reference System coordinates,
- LAMBERT – which performs forward and inverse Lambert Conformal Conic projections,
- LOCAL CARTESIAN – which performs conversions between geodetic, geocentric, and Local Cartesian projections,
- MERCATOR – which performs forward and inverse Mercator projections,
- MGRS – which performs conversions between geodetic, UTM, UPS, and Military Grid Reference System coordinates,

- MILLER – which performs forward and inverse Miller Cylindrical projections,
- MOLLWEIDE – which performs forward and inverse Mollweide projections,
- ORTHOGRAPHIC – which performs forward and inverse Orthographic projections,
- POLAR STEREO – which performs forward and inverse Polar Stereographic projections,
- POLYCONIC – which performs forward and inverse Polyconic projections,
- SINUSOIDAL – which performs forward and inverse Sinusoidal projections,
- TRANSVERSE CYLINDRICAL EQUAL AREA – which performs forward and inverse Transverse Cylindrical Equal Area projections,
- TRANSVERSE MERCATOR – which performs forward and inverse Transverse Mercator projections,
- UPS – which performs conversions between geodetic and Universal Polar Stereographic coordinates,
- UTM – which performs conversions between geodetic and Universal Transverse Mercator coordinates,
- VAN DER GRINTEN – which performs forward and inverse Van der Grinten projections.

This component also depends on the standard C character handling, math, input/output, string handling, and general utility libraries.



APPENDIX B DEFINITIONS/GLOSSARY

Central Meridian – Longitude at the horizontal center of a projection; Origin Longitude.

Coordinate – Linear or angular quantities that designate the position that a point occupies in a given reference frame or system. Also used as a general term to designate the particular kind of reference frame or system, such as Cartesian or spherical coordinates.

Ellipsoid – The surface generated by an ellipse rotating about one of its axes.

False Easting – A coordinate value (in meters) assigned to the central meridian of the projection to avoid the inconvenience of using negative coordinates.

False Northing – A coordinate value (in meters) assigned to the origin latitude of the projection to avoid the inconvenience of using negative coordinates.

Geodetic Coordinates – The quantities of latitude and longitude that define the position of a point on the surface of the earth with respect to the reference ellipsoid. Also, imprecisely called geographic coordinates.

Geodetic Latitude – The angle between the plane of the equator and the normal to the ellipsoid through the computation point. Geodetic latitude is positive north of the equator and negative south of the equator.

Geodetic Longitude – The angle between the plane of a meridian and the plane of the prime meridian. A longitude can be measured from the angle formed between the local and prime meridians at the pole of rotation of the reference ellipsoid, or by the arc along the equator intercepted by these meridians.

Map Projection – A function relating coordinates of points on a curved surface (usually an ellipsoid or sphere) to coordinates of points on a plane. A map projection may be established by analytical computation or, less commonly, may be constructed geometrically.

Map Scale – The ratio between a distance on a map and the corresponding actual distance on the earth's surface.

Meridian – A north-south reference line, particularly a great circle through the geographical poles of the earth, from which longitudes and azimuths are determined; or the intersection of a plane forming a great circle that contains both geographic poles of the earth, and the ellipsoid.

Origin Latitude – Latitude at which the northing coordinate of the projection is zero.

Parallel – A line on the earth, or a representation thereof, that represents the same latitude at every point.

Scale Factor (Projection) – A multplier for reducing a distance in a map projection to the actual distance on the chosen reference ellipsoid.

Standard Parallel– Latitude at which the scale factor of the projection is 1.0.

APPENDIX C REFERENCES

(1) Topographic Engineering Center, TEC-SR-7,.**Handbook for transformation of DATUMS, PROJECTIONS, GRIDS, AND COMMON COORDINATE SYSTEMS**, January 1996.