Legal Statement

1. The copyright of this document is covered by SuperMap Software Co., Ltd. in accordance with the Copyright Law of the People’s Republic of China and the Universal Copyright Convention. If, without the written permission of the company, any part of the document shall not in any way or any reason be used, copied, modified, transmitted, or bundled with other products to be used, sold, tort reserved.

2. “超图”, “SuperMap”, and 

3. This document represents no responsibilities of any supplier or agent. Without statement, SuperMap Software Co., Ltd. has right to do modifications to this document.

4. The copyright of trademarks mentioned in this document belongs to the corresponding companies. Without the written permission of these companies, the trademarks shall not in any way or any reason be used, copied, modified, or transmitted.

5. The concerned software products and the updated products hereinafter in this document are developed and sold by SuperMap Software Co., Ltd.

Hereby declare

SuperMap Software Co., Ltd.:

Add: 7/F Tower B, Technology Fortune Center, No. 8 Xueqing Road,
Haidian District, Beijing, 100192, P. R. China

Tel: +86-10-82736655-4170
Fax: +86-10-82734630
HomePage: www.supermap.com

Sales: request@supermap.com

Tech Support: globalsupport@supermap.com

SuperMap Software welcomes all advices and suggestions from you.
Content

1 Summary ........................................................................................................................................ 1

2 Introduction to Component Structure ................................................................................................1
   2.1 SuperMap Objects Java/.NET Class Types ............................................................................ 1
   2.2 SuperMap Objects Java/.NET Class Relations ..................................................................... 3

3 Module Classes .................................................................................................................................. 7
   3.1 Data Module ............................................................................................................................. 7
      3.1.1 Workspace, Datasources, Maps, Layouts, Resources, Scenes ........................................ 8
      3.1.2 Datasource, Datasets, Dataset ......................................................................................... 11
      3.1.3 PrjCoordSys ....................................................................................................................... 15
      3.1.4 DatasetVector ..................................................................................................................... 21
      3.1.5 Recordset ............................................................................................................................ 23
      3.1.6 Geometry, Geometry3D .................................................................................................... 28
      3.1.7 Other Classes ...................................................................................................................... 32
   3.2 Mapping Module ......................................................................................................................... 38
      3.2.1 MapControl, Map, Layers ................................................................................................ 38
      3.2.1 Layers, layer, Theme ......................................................................................................... 44
      3.2.2 Theme ................................................................................................................................. 47
   3.3 SpatialAnalyst Module ................................................................................................................. 51
      3.3.1 BufferAnalyst .................................................................................................................... 52
      3.3.2 OverlayAnalyst ................................................................................................................... 52
      3.3.3 ProximityAnalyst ............................................................................................................... 53
      3.3.4 Interpolator ......................................................................................................................... 54
      3.3.5 SurfaceAnalyst .................................................................................................................. 56
   3.4 Layout Module ............................................................................................................................. 56
   3.5 Realspace Module ........................................................................................................................ 60
      3.5.1 Scene, SceneControl .......................................................................................................... 60
3.5.2 Layer3Ds, Layer3D , Theme3D ................................................................. 64
3.5.3 Theme3D ............................................................................................... 67

3.6 NetworkAnalyst Module ............................................................................. 69
  3.6.1 NetworkBuilder..................................................................................... 69
  3.6.2 TransportationAnalyst........................................................................... 69
  3.6.3 FacilityAnalyst ...................................................................................... 73
  3.6.4 TrafficTransferAnalyst ......................................................................... 74

3.7 Topology Module ......................................................................................... 77

3.8 Processing Module ...................................................................................... 78

3.9 Conversion Module ...................................................................................... 79
SuperMap Objects Java/.NET are .NET based development platforms in the SuperMap GIS Universal series. They adopt the same Universal GIS Core (UGC) concept, the same frame and architecture and each has ten modules, or ten components. Please see in the following diagram. In Objects .NET, each module corresponds to a .NET assembly (.dll) while in Objects Java, each module corresponds to a jar package(JavaBean). Please note that the data module is the core module.

![Diagram 1 SuperMap Objects Java/.NET namespace structure](image)

SuperMap Objects .NET 6R provides the following independent modules with the Data Module being the core module, which provides data access support for other modules:

1. Data Module: Spatial data are the blood of a GIS system. In SuperMap Objects .NET, a module is created for common operations on spatial data and their attributes, e.g. creating, managing, accessing, and querying data. The data module supports updating data in batch, editing
the historical records and calculating of geometry objects, etc. It also supports the management of data version and metadata.

2. Mapping Module: Maps are a good representation of geographical and spatial data and the spatial relationship, and mapping is one of the fundamental functions of GIS. The mapping module in SuperMap Objects .NET has exposed mapping related functions including map displaying, rendering, editing, and laying out. This module provides the function of creating thematic maps, including label thematic maps, graph thematic maps, ranges thematic maps, dot density maps and so on. Meanwhile it also supports mapping presentation.

3. Realspace Module: This module is for the integrated display of 2D and 3D data. Global terrain data and high-resolution images can be loaded and displayed in the 3D model. Operations including browsing, panning, selecting, querying, and position can be performed in this module.

4. SpatialAnalyst Module: Spatial analysis is the basic component of integrated geographical analysis model. In SuperMap Objects Java/.NET products, spatial analyst functions are provided in the SpatialAnalyst Module. This module covers most of the spatial analysis functions, including buffering, overlaying, neighborhood, surface, interpolating, etc.

5. NetworkAnalyst Module: Network analysis is the important composition of spatial analysis. Because its common but complicated using, Network module is technically designed for network analysis applications. While you use this module, you can perform public transport transfer analysis, traffic network analysis, which includes resource allocation and location, TSP analysis, logistics vehicle routing analysis, path finding, closest facilities analysis, and facilities analysis.

6. Conversion Module: Data formats are a great challenge for any GIS systems since different GIS vendors have created their own data formats. To facilitate the interoperability of GIS data in SuperMap Objects Java/.NET, the conversion module supports conversion between different data formats, both vector and raster.

7. Layout Module: This module is for map laying out, presentation, and printing. The layout in SuperMap Objects Java/.NET shares the same object model with 2D map. This module supports various color models, such as CMYK, and RGB high-performance printing of large-volume data. It also provides standard map sheets and frames for your professional map layout and specific mapping needs.
8. Topology Module: This module provides topology validating and topology processing functions.

9. Processing Module: This module provides a cache generating function for 3D data, including 3D image data, terrain data and 3D model data.
Introduction to Component Structure

By learning the class structure of SuperMap Objects Java/.NET, also named the Component Objects Model Diagram (COMD), you will quickly get familiar with the relations between different classes in SuperMap Objects Java/.NET.

The Component Objects Model Diagram is created based on the Unified Modeling Language (UML), from which developers can learn the relations between different classes and how to get one object from another object, consequently help they choose the appropriate interfaces, get the proper properties and methods, and finally to an application system. The Component Objects Model Diagram is provided in the Help document.

Before you learn the class structure in SuperMap Objects Java/.NET, you need to get a glimpse of the different classes in SuperMap Objects Java/.NET and their relations:

2.1 SuperMap Objects Java/.NET Class Types

In SuperMap Objects Java/.NET, there are six types of classes, as shown inDiagram 2: Abstract Class, Creatable Class, Non-creatable Class, Static Class, Structure Class, and Enumeration Class.
Abstract Class: An abstract class cannot be instantiated. An abstract class is designed only as a parent class from which child classes may be derived. For example, Geometry is an abstract class which has no Geometry objects. GeoPoint, GeoLine, GeoRegion, and many other concrete types of classes are derived from Geometry. Members defined by an abstract class can be inherited by its subclasses.

Creatable Class: A creatable class can have its instance objects created by using the "new" keyword and initialized. And its instance objects exist independently of other objects, and its life cycle of the instance objects is not managed by other objects. For instance, Workspace is a creatable class. You can create a Workspace object using the "new" keyword.
Workspace workspace = new Workspace(); (.NET)
Workspace workspace = new Workspace(); (Java)
Where, workspace is an instance of Workspace.

Non-creatable Class: A non-creatable class cannot create its instance object directly, but can getting it by the method of another class. You cannot use the "new" keyword or a constructor to
create objects of a non-creatable class. For instance, Datasources is a non-creatable class. It is created through the Datasources methods of the Workspace class.

Datasources datasources = workspace.Datasources(); (.NET)
Datasources datasources = workspace.getDatasources(); (Java)

Where, workspace is an object of Workspace; datasources is an object got from the Workspace object.

**Static Class** : A static class and its members are used to create data and functions that can be accessed without creating an instance of the class. Static classes work like a set of geographical processing tools, each method being one tool with which you can input parameters and get the results. For example, Geometrist is a static class and can be used in the following way:

GeoRegion geoRegion3 = Geometrist.Intersect(geoRegion1, geoRegion2); (.NET)
GeoRegion geoRegion3 = Geometrist.intersect(geoRegion1, geoRegion2); (Java)

Where, geoRegion1, geoRegion2 and geoRegion3 are three GeoRegion objects. You can see from the example that Geometrist Class can be used directly, without creating an object of this class.

**Structure Class** : The structure class is a concept in .NET, rather than in Java. In Java, the structure class is the creatable class.

**Enumeration Class** : An enum class defines a number of constants and has no instance objects.

For example, EncodeType is an enum class:

datasetVectorInfo.EncodeType(EncodeType.Int32); (.NET)
datasetVectorInfo.setEncodeType(EncodeType.Int32); (Java)

Where, datasetVInfo is an object of DatasetVectorInfo.

## 2.2 SuperMap Objects Java/.NET Class Relations

In SuperMap Objects Java/.NET, three main relations exist between classes: Aggregation, Inheritance and Association, while there are three types of aggregation relations. All these relations are described in the UML diagram.

1) **Aggregation**

Diagram 3 shows the three types of aggregation relations.
Object A includes one Object B: it means that Object A must include or associate with one and only one Object B. For example a Datasource object is sure to associate with a Datasets object.

Object A includes no Object B or one Object B: it means that Object A can only include or associate with one or no Object B. For example, a Layer object can include the Theme object or not.

Object A includes no Object B or more than one Object B. In this relation, Object A is usually a collection of Object B. For example, a Datasources object can include more than one Datasource object.

Diagram 3 Aggregation

2) Inheritance

An abstract class cannot create an object directly. The properties and methods of an abstract class can only be implemented by its subclass which will inherit all the non-private methods and properties of the parent class. And it can be seen as a type of parent class. The subclass and its parent class is an inheritance relation.

As shown in Diagram 4, B inherits from A. A is the parent class while B is the subclass of A. The inheritance between classes is as an extension of class functions. When inheriting from its parent class, a subclass adds its own behavior and properties while reserving that of the parent class. E.g. GeoPoint, GeoLine, GeoRegion and GeoText all inherit from the abstract class Geometry. Inheritance relation does not restrict only to abstract classes and their subclasses. However, this
kind of inheritance relation is commonly used in SuperMap Objects Java/.NET.

![Diagram 4 Inheritance](image)

3) Association

A and B have some association with each other, and the objects in Class B can be accessed via the objects in Class A. It is that A and B has an association relation.

![Diagram 5 Association](image)

Association relation shows a loose connection between classes in general. In SuperMap Objects Java/.NET, there are two types of association relations.

- The first type is the relation of using and being used. For example, the main class uses the enumeration class, the main class uses its property class and the main type uses other information classes while being created; Datasource and EngineType have an association relation in that Datasource sets its engine type via EngineType; And Workspace and WorkspaceConnectionInfo have an association relation in that Workspace creates, opens and saves workspaces via WorkspaceConnectionInfo.

- The other type is the operating relation. One class operates on another class. For example, you can get a Recordset object by doing a query option on DatasetVector, and you can also get a corresponding DatasetVector object by a Recordset object.
Module Classes

SuperMap Objects Java/.NET contains the following modules:

3.1 Data Module

In Data module, you can open, connect, edit, delete and save data. Workspace, datasource, dataset are very important conceptions and the main units for data organizing and management in SuperMap data structure. Because different classes in data module manage different contents, their functions and operations are different. Thus it is a great help for users to known the functions provided by each class and the relationships between classes. The basic GIS functions include the
query, edit and other operations to the geometry and attribute data of vector dataset. These operations perform by the DatasetVector, Recordset and Geometry classes as well as the cooperation of them. In data module, the Maps class is a connection to Mapping class, which helps data visualization. Besides, the Data module provides various 3D geometry objects that used in 3D module.

### 3.1.1 Workspace, Datasources, Maps, Layouts, Resources, Scenes

The workspace is the working environment of the users. The workspace of an application stores the connection information of all the datasources, the organization relation of maps, layout information, 3D scene information and sources used. A workspace manages its datasources, maps, layouts, 3D scenes, and resources through the Datasources object, Maps object, Layouts object, Scenes object, and Resources object. The Datasources object in the workspace only stores the connection information, the datasource actually stores in database or SDB file. The Maps object in a Workspace object stores some configuration information, such as the number of layers in a map, the dataset used by the layer, the bounds of the map, and the style of the background.

Diagram 7 shows the relation of classes related to the workspace.

![Diagram 7 Workspace Class Structure](image)

The Workspace class allows you to create a new workspace, and open and close an existing workspace. While doing these operations, the WorkspaceConnectionInfo is needed as a parameter. WorkspaceConnectionInfo stores all the information to connect or open a workspace, such as the type of the workspace, the name of the server, the user's name and password. Different
information needs to be set in different workspaces.

Through the Workspace objects, users can access Datasources object, Maps object, Layouts object, Resources object, and Scenes object and then to manage the datasources, maps, layouts, resources, and scenes in the workspace. Each workspace has a Datasources class, a Maps class, a Layouts class and a Scenes class. The Datasources class is used to manage all the datasources in the workspace. The Maps class is used to manage all the maps saving in the workspace. The Layouts class is used to manage all the layouts in the workspace. The Resources class is used to manage all the resources used by the workspace. The Scenes class is used to manage all the scenes in the workspace. In addition, in Objects .NET, users can also get or set the properties of the workspace, including get or set its type, version, name, etc. In Objects Java, users can return or set the methods for the Workspace class, including return or set its type, version, name, etc of the workspace.

A workspace can be saved in a file or in a database. The enumeration class WorkspaceType defines the five types of workspaces users can save, including defaulted unsaved workspace, SXW workspace(in XML format), SMW workspace, Oracle workspace, and SQL Server workspace.

WorkspaceVersion defines the constants of the version types of the Workspace. It is not only supports the workspace in its own version, UGC 2.0 version, but also supports SuperMap 5 series and subsequent version of workspaces.

There can be more than one workspace in an application, and each workspace manages the datasources, maps, etc. in it. Note that you cannot do any operation cross workspaces.

The Datasources object of a workspace allows you to create, open and close a datasource, get a Datasource object by its index or alias name, get the count of the datasources. Note that when you create or open a datasource, a datasource created or opened will be returned, and the datasource object obtained through the method Create and Open is an opened datasource.

The Maps object manages the maps in a workspace by adding, removing, and inserting map, getting the names and index numbers of the map, getting the XML string of the map, and replacing map by XML strings. The GetAvailableMapName method can be used to get any unique map name to avoid the case that a map is failed to be loaded because its name is not unique. Note that unlike the Datasources object, the Maps object cannot get any map but their name, and a map is opened using its map name as a parameter.
The Layouts object manages the layouts in a workspace by adding, removing, and inserting layouts, getting the names and index numbers of the layouts, getting the XML string of the layouts, and replacing layouts by XML strings. The GetAvailableLayoutName method can be used to get any unique layout name to avoid the case that a layout is failed to be loaded because its layout name is not unique.

The Scenes object manages the scenes in a workspace by adding, removing, and inserting scenes, getting names and index numbers of the scenes, getting the XML string of the scenes, and replacing scenes by XML strings. The GetAvailableSceneName method can be used to get any unique scene name to avoid the case that a scene is failed to be loaded because its scene name is not unique.

The Resources object manages the resources in the workspace, mainly the symbol resources. The symbol resources are in essence the symbol libraries, including the point symbol library, line symbol library, and fill pattern library. And the each symbol library manages its own symbols. The relations of the Resources class and its symbol libraries are shown below:

Symbol

The Symbol class is the base class for all the symbols, including SymbolMarker class, the SymbolLine class and the SymbolFill class. They are all inherited from the Symbol class.

SymbolLibrary

The SymbolLibrary class is the base class for all the SymbolLibrary class, including the SymbolMarkerLibrary class, the SymbolLineLibrary class and the SymbolFillLibrary class. They are all inherited from the SymbolLibrary class.

SymbolGroups, SymbolGroup

SymbolGroups is the set class for the SymbolGroup classes. These classes are used to manage the logical structure of symbols in the symbol library. Symbols of the same category and use are managed in the same group.
3.1.2 Datasource, Datasets, Dataset

Datasource is used to manage the projection information, the connection information of datasource and database, and the related operations on the datasets in this datasource.

Diagram 9 shows the classes related with the Datasource class. A new dataset can be created by copying the existing dataset. The datasource objects work as an element of the datasource in workspace. The datasource objects can be saved and the dataset can be copied.

Datasource defines the uniform data access interface and criterion. And its physical storage can be in file or database mode. Several datasource types are provided in SuperMap products, including the file datasource and the database datasource. The SDB datasource is the file datasource type, and the oracle and SQL Server datasource is the database datasource type.

One datasource is corresponding with a spatial data engine (EngineType). The spatial database engine is built on the base of regular DBMS. Besides the basic functions of the regular DBMS, the spatial database engine provides specific storage and management capabilities aiming at spatial data. SDBPlus, OraclePlus, SQLPlus, ImagePlugins are supported engine types in this version.

Multiple datasources can be open in one workspace. Different databases are indentified by different aliases. For file mode datasource (SDB datasource), all the data in the same datasource use the same projection coordinate system (PrjCoordSys); for other types of datasource, they can use their own projection coordinate system. Because the base class, Dataset class, can defines the custom projection coordinate system of Dataset object by the provided PrjCoordSys property. If the
dataset has no projection coordinate system, it will use the same projection coordinate system with the datasource it belonging to; otherwise it will use its own projection coordinate system. The projection coordinate system is composed of map projection mode, projection parameters, coordinate unit and geographic coordinate system. SuperMap Objects .NET /Java provides many predefined projection systems which can be used by users directly. In addition, users can customize projection system of their own. More details about PrjCoordSys class, please refer to Section 3.1.3.

All the connection information about a datasource will be stored in DatasourceConnectionInfo class. This information includes the name of the server that the datasource will connect to, the name of the database, the user’s name and password needed to login the database, etc. The connection information of the data source in a workspace will be stored in the file of this workspace. For different datasources, the connection information will be different.

All the datasets in the datasource are managed by a Datasets object. The Dataset is mainly used to manage datasets and performs some operations on datasets, such as create dataset, rename and delete dataset.

The Datasets can contain one or more types of dataset, such as the vector dataset, the raster dataset or the image dataset.

**DataSetType**
Some types of the datasets are supported in this version including tabular dataset, point dataset, line dataset, region dataset, text dataset, CAD dataset, LineM dataset, network dataset, topology dataset, image dataset and grid dataset.

EncodeType

EncodType is the compression encoding type of dataset storage. It supports four encode types for vector datasets. They are single-byte, double-byte, triple-byte and four-byte encode type.

Diagram 10 Dataset Class Structure

Dataset

Dataset is the base class of all the datasets, such as vector dataset and grid dataset. It provides the common properties, methods and events for different datasets. A dataset is a collection of the same data type, which is the minimum unit of GIS data organization. According to the types of data, there are Dataset Vector, DatasetGrid, DatasetImage, and the dataset designed for specific problems, such as topology dataset, network dataset, etc. The vector dataset is a collection of
spatial features with the same type; it is also called feature dataset. For different features may have different spatial representations, the vector dataset includes point dataset, line dataset, region dataset and so on. Each vector dataset is a collection of features with the same spatial representation and characteristic. The image dataset and the grid dataset are composed of cell arrays. They are good in presenting the location relationship of spatial phenomena, though not so satisfied in representing features.

**DatasetVector**

For the introduction to vector dataset class, please refer to Section 3.1.4.

**DatasetImage**

DatasetImage class. This class is used to describe the image data, such as grid map, multi-band image and photos, without any property information. Before performing any operations, you need to call Dataset.open() method to open the dataset.

**DatasetGrid**

DatasetGrid class. This class is used to describe the grid data, such as DEM dataset and land use map. Before performing any operations, you need to call Dataset.open() method to open the dataset.

**DatasetGridInfo**

DatasetGridInfo class. This class contains the information about grid dataset, such as the name, the width, the height, the encode type, block size, null value and so on.

**DatasetImageInfo**

DatasetImageInfo class. This class contains the information about image dataset, such as the name, the width, the height, the encode type, block size, etc.

**PixelFormat**
PixelFormat class. This class defines the pixel format of the grid dataset and the image dataset. For detailed information about PixelFormat, please refer to Programming Reference.

**DatasetTopology**

Dataset Topology Class.

**DatasetTopologyInfo**

DatasetTopologyInfo class. This class contains the name information about topology dataset. By this class, you can get the string of the name of the topology dataset.

**TopologyDatasetRelationItems, TopologyDatasetRelationItem**

TopologyDatasetRelationItems class is a collection of TopologyDatasetRelationItem class. TopologyDatasetRelationItem is the class for providing necessary parameters in topology preprocessing. The parameters include the dataset to be preprocessed and the precise index of the preprocessed dataset.

**TopologyValidatingItem**

The TopologyValidatingItem class is used for the topology validating, providing the necessary parameters. These parameters include the dataset to be validated, the reference dataset and the topology rule.

**TopologyRule**

TopologyRule class. This class performs the validation of point, line and region data. And it can return the dataset object which has the topology error according to the topology rules.

### 3.1.3 PrjCoordSys

Because all the data in the same datasource use the same projection coordination system, SuperMap Objects .NET /Java provides the corresponding class and enumerations of projection
coordination system. Diagram 11 shows the Class Structure about projection. The following is the introduction to functions of each class.

**PrjCoordSys**

PrjCoordSys class. A projection coordinate system is composed of map projection mode, projection parameters, coordinate unit and geographic coordinate system. SuperMap Objects .NET provides many predefined projection systems which can be used directly. In addition, users can customize projection systems of their own. The projection coordinate system is defined on a two dimensional surface. Unlike geographic coordinate system by using Lon/Lat to identify the location, the projection coordinate system identify locations by x, y coordinate. Each projection coordinate system is based on a geographic coordinate system.

**CoordSysTranslator**

CoordSysTranslator class. This class is mainly used to transform the different projection coordinates and projection coordinate systems. It provides three methods to transform different projection coordinate systems. The Forward method is for the transformation between geographic coordinate systems (Lon/Lat) and the projection coordinate systems. The inverse method is for the transformation between the projection coordinate systems and geographic coordinate systems (Lon/Lat). The Convert method is for the transformation between two different projection coordinate systems.

**CoordSysTransParameter**

CoordSysTransParameter class. The projection parameters contains translation factor, rotation factor and scale factor. This class is used for the transformation of the reference coordinate systems, including the translation of the reference coordinate, the rotation of the reference coordinate system and the scaling up/down of the reference coordinate system. If the source and target datum are different in a projection transformation, it may be necessary to transform from one to another. The three-parameter and the seven-parameter methods are commonly used for projection transformation.

**CoordSysTransMethod**
The types of the projection transformation methods. If the source and target geographic coordinate system is different, it may be necessary to transform the datum from one to another. There are two methods commonly used, they are the grid-based method and the equation-based method. This class provides the method based on the equation. According to the number of the parameters, the equation-based method can be divided into three-parameter and seven-parameter methods. The seven-parameter method is the most commonly used. GeocentricTranslation, MolodenskyMolodensky, MolodenskyAbridged are three-parameter methods. GeocentricTranslations, Molodensky, MolodenskyAbridged are seven-parameter methods.

Diagram 11 PrjCoordSys Class Structure

GeoCoordSys

The GeoCoordSys class. The geographic coordinate system is composed of Datum, Central Meridian
and an angular unit of measure. In a geographic coordinate system, the unit traditionally measures in degrees, minutes, and seconds degree. The horizontal direction or east–west direction ranges from -180 degree to 180 degree and the vertical direction or north–south direction ranges from -90 degree to 90 degree. The geographic coordinate system is a sphere coordinate system defined the locations of the surface by longitude and latitude.

**GeoCoordSysType**

Constant that defines the type of geographic coordinate systems.

**GeoSpatialRefType**

Geo-spatial coordination type is used to distinguish the planar coordinate system, the geographic coordinate system and the projection coordinate system.

**Unit**

Specifies the unit constants. For more information about the members of Unit enumeration, please refer to the Programming Reference.

**GeoPrimeMeridian**

The prime meridian class. The object of this class is mainly used in geographic coordinate system, which is composed of prime meridian, reference system or datum and angle unit.

**GeoPrimeMeridianType**

The prime meridian enumeration type. For more information about GeoPrimMeridianType, please refer to the Programming Reference.

**GeoDatum**

The geodetic reference system (GeoDatum) class. The class includes the earth ellipsoid parameters. The earth ellipsoid is only a description to the size and shape of the earth. For more accurate
description of the real position of objects on the earth, datum is needed. The geodetic reference system (GeoDatum) decides the relative position between the earth spheroid and the centroid of the earth. It provides a frame for surveying the objects on the earth and also determines the origin and direction of Lat/Lon Grid in the earth surface. The Datum sets the centroid of ellipsoid as its origin. The earth ellipsoid of the datum in a region may offset the true centroid of the earth more or less, but it is the best description to the surface of this region. The coordinates of all the objects in the surface are relative to this ellipsoid centroid. Now, the commonly used datum is WGS84, which acts as the basic frame for earth surveying. Different datum is fit for different countries and regions. One Datum is not fit to any regions.

**GeoDatumType**

The datum enumeration class. The earth spheroid parameter type. This type is used to describe the major radius and flattening of an ellipsoid.

**GeoSpheroid**

The earth spheroid parameter class. This class is used to describe the major radius and flattening of an ellipsoid. Although the earth is best represented by a spheroid, the earth is sometimes treated as a sphere to make mathematical calculations easier. The assumption that the earth is a sphere is possible for small-scale maps, those smaller than 1:1,000,000. At this scale, the difference between a sphere and a spheroid is not detectable. However, to maintain accuracy for larger-scale maps (scales of 1:1,000,000 or larger), a spheroid is necessary to represent the shape of the earth. A sphere is based on a circle, while a spheroid (or ellipsoid) is based on an ellipse. The shape of an ellipse is defined by two radiuses. The longer radius is radius of the equator, and the shorter radius is the radius of the polar.

Because of the same projection and different ellipsoid parameters, the same data can be projected differently. Therefore, it is required to select appropriate ellipsoid parameters. The ellipsoid parameters are different at different times, countries and regions. At present, the Beijing 54 Coordinate system, the Xi’an 80 Coordinate system and the State Geodetic System in 2000 are mainly used in China, corresponding to the Krasovsky ellipsoid, IAG75 ellipsoid and CGCS2000 ellipsoid respectively. However in the worldwide, the WGS84 ellipsoid is commonly used.
GeoSpheroidType

Defines the GeoSpheroid parameters object type. For detailed information, please refer to the Programming Reference.

PrjCoordSysType

Projection coordinate system enumeration class. For detailed information, please refer to Namespace.

PrjParameter

The projection parameters class. This class defines the parameters of map projection, such as meridian, center latitude, and standard parallel of projection with two standard parallels. For detailed information, please refer to the Programming Reference.

Projection

Map projection class. A map projection converts data from the round earth onto a flat plane. Many common map projections are classified according to the projection distortions: conformal projection, equidistant projection and equal-area projections. Each map projection is designed for a specific purpose and distorts the data differently. For example, the conformal projections are commonly used in sea map. There is another arbitrary projection whose distortion is intervenient between the above three type projections. The arbitrary projection are usually used for reference and teaching. And map projections are classified according to their component ways: the geometrical projection and non-geometrical projection. The geometrical projections project a gridded network called a graticule on a geometric surface and unfolding this surface into a plane. This kind of projections contains azimuth projections, cylindrical projections and conical projections. The non-geometrical projections do not rely on geometric surface. It builds a function relation between the points in spheroid and plane by mathematical formulas according some conditions. This kind of projections contains pseudoazimuthal projections, pseudocylindrical projections, pseudoconic projections and mutil-conic projections.

ProjectionType
The projection type of the projection coordinate system. For more details, please refer to the Programming Reference.

### 3.1.4 DatasetVector

The DatasetVector class (DatasetVector), is inherited from Dataset. It can manage and do some operations on vector dataset. The operations include querying, modifying, deleting, building index and so on. The methods of DatasetVector class, such as Open and Close methods, are inherited from the methods of Dataset class. Diagram 12 shows the Class Structure of DatasetVector class. The following is the introduction to the functions and relations of these classes.

![Diagram 12 DatasetVector Class Structure](image)

**Tolerance**

The Tolerance class is used to set all the tolerance value of vector datasets. Except the minimum polygon tolerance, other tolerance can be used in the topology processing.

**FieldInfos**

The FieldInfos class. The class can be created individually, and can also be obtained from a vector dataset or a recordset. The collection of the fields in a vector dataset or in a record set is an instance of the FieldInfos class.
FieldInfo

The FieldInfo class. This class stores the information of the fields, such as name, type, default value and length. Each field is corresponding with a FieldInfo object. For every field in the vector dataset, you can only modify the Caption property of the field, and other information of the field cannot be modified.

You can get the FieldInfo objects in two ways:

One is from the Recordset, which is generated by performing the query operation.

The other is from the DatasetVector. And you can add new fields in a vector dataset.

FieldInfos contains one or multiple FieldInfo objects.

FieldType

The Field type constant. You can get or set the field type by FieldInfo.Type.

SpatialIndexType

It defines the types of spatial index of vector datasets. A spatial index is a data structure used to improve the efficiency of spatial data query. Four types of spatial index are supported in SuperMap, namely R-tree spatial index, quadtree index, tile spatial index and multi-level grid spatial index.

SpatialIndexInfo

The SpatialIndexInfo class. This class provides the information of building spatial index for vector datasets, such as the type of spatial index, counts of leaf nodes, field of tile, the width and height of tile, and the size of multi-level grid.

Only one of spatial index can be used at one time in a vector dataset, but it can be switched to another kind of spatial index. For completing the switch, you should delete the former spatial index before building a new spatial index. When a dataset is in editing status, the current index will be maintained automatically. Specially, if a dataset has been edited several times, the efficiency of spatial query index will be affected. SuperMap provides a method to notify users whether to build
a new spatial index or not in this situation.

**DatasetVectorInfo**

The DatasetVectorInfo class. This class contains the information about vector dataset, such as the name of a dataset, the type of a vector dataset, the encoding type, whether using a file cache, etc.

**StatisticMode**

This class provides the statistic mode of the fields, including the standard variation (StdDeviation), variance (Variance), the sum (Sum), the average value (Average), the maximum (Min) and the minimum value (Max).

**Charset**

Defines a character set constant. This class provides the character set type for the text of the vector dataset.

### 3.1.5 Recordset

The Recordset class. You can perform operations on the vector data by this class. Diagram 13 shows the structure of the Recordset and its relative classes. The following is a simple introduction to their functions and relations.

![Diagram 13 Recordset Class Structure](image)

There are two types of datasource, the file datasource and the database datasource. In a database, the spatial geometry information and the attribute information of the feature are stored together.
A vector dataset is stored in a DBMS table, in which the geometry information and the attribute information of the vector data are stored. The geometry field stores the geometry information of the features in the table. A row in the table is called a piece of record.

For the attribute dataset in a vector dataset, there is no geometry field. The record is a subclass of DBMS table. In a file dataset, the spatial geometry information and the attribute information are stored separately. Actually, the recordset treats the file data the same as the database data, regarding both as tables storing the spatial geometry information and the attribute information together. However, the recordset is a subclass getting out of the attribute dataset for operations. A piece of record in recordset, or a row, is corresponding to a feature, including the spatial geometry information and the attribute information of this feature. And a column of the recordset is called a field.

Mapcontrol and converting the selection to the Recordset; the other is getting a Recordset by vector dataset. The users can use the DatasetVector.GetRecordset() method to get the Recordset directly, or by query statements. The difference is that the former gets a recordset contains all the geometry and the attribute information of the dataset, the latter gets a recordset selected by the query conditions.

**QueryParameter**

The QueryParameter class. It describes the limiting conditions of conditional query, such as the SQL statement, cursor type, condition setting of spatial location relationship, etc. Conditional query is selected a recordset of all the features satisfying certain conditions. And the result of this query is a Recordset.

The QueryParameter class is used to set the conditions in a query. There are two main query manners: one is SQL query or attribute query, the other is spatial query. The SQL query selects records by building conditional statements, including attribute fields, operators and numerical values and gets a recordset. The spatial query selects features by their spatial relations and gets a recordset.

In .NET, the AttributeFilter, OrderBy and GroupBy properties of QueryParameter class are used to get and set the information of attribute query, while SpatialQueryObject and SpatialQueryMode are used for spatial query. In addition, the JoinItems and LinkItems properties are used for get and
set the joint or link with an exterior table, and perform a query on the table.

In Java, the getAttributeFilter(setAttributeFilter), getOrderBy(setOrderBy) and getGroupBy(setGroupBy) properties of the QueryParameter class are used to return and set the relative information of attribute query; the getSpatialQueryObject(setSpatialQueryObject), and getSpatialQueryMode(setSpatialQueryMode) properties are used to return and set the relative information of spatial query. In addition, the JoinItems and LinkItems properties can get or set the join or link information with an exterior table, perform a query on the table.

**CursorType**

The cursor is generally used as a pointer which points a record in a recordset. The users can locate the record by moving the cursor forward and backward. There are two types of cursors in SuperMap; they are dynamic cursor and static cursor. The dynamic cursor is used to save operations on a recordset, such as editing, updating. But it will occupy a lot of system resources and the cost is high. The static cursor is used to query data, which has a high efficiency. In .NET, you can set the cursor type by the QueryParameter.CursorType property, while in Java, you set it by the setCursorType() method of QueryParameter class.

**SpatialQueryMode**

The spatial query mode constants. The spatial query is a query method that selects geographic features based on their locations or spatial relationship. For example, a spatial query may find the features that contained within a region, or the features separated from each other, or the features that are adjacent to each other.

**JoinItems**

The JoinItems class. The JoinItems object is used to manage all the JoinItem objects when a vector dataset joins with several exterior tables. In .NET, the join information is gotten or set by the QueryParameter.JoinItems property. In Java, the join information is returned or set by the getJoinItems() and setJoinItems() methods of QueryParameter class.

**JoinItem**

The JoinItem class. The JoinItem class is used to join a vector dataset and an exterior table. The exterior table can be the DBMS table of another vector dataset (including the tabular dataset,
which has no geometry information) or the table in a database defined by the users. Tables can be connected with each other in two manners, one is ‘join’, and the other is ‘link’. ‘join’ actually adds an exterior table to the specific table based on the same field (fields) the two tables have. ‘link’ defines the links between two tables, not adding one to another. Please refer to LinkItem. The names of the fields to link two tables can be different, but the data type of the fields must be consistent.

Once the two tables are joined together, you can perform query, mapping and analysis on the exterior table via the main table. When the relationship between two tables is one-to-one or many-to-one, you can use join to connect them. And if the relationship between two tables is many-to-one, the users are allowed to specify multiple connections between fields.

**LinkItems**

The LinkItems class. When a vector dataset connects with multiple exterior tables, a LinkItems objects manage all the LinkItem objects. In.NET, the QueryParameter.LinkItems property is used to get or set the connection information with the exterior tables; in Java, getLinkItems() and setLinkItems() methods of the QueryParameter class are used to return and set the connection information with the exterior tables.

**LinkItem**

The LinkItem class. The LinkItem class is used to link a vector dataset and an exterior table. The exterior table can be the DBMS table of another vector dataset (including the tabular dataset, which has no geometry information) or the table in a database defined by the users. Tables can be connected with each other in two manners, one is ‘join’, and the other is ‘link’. ‘join’ actually adds an exterior table to the specific table based on the same field the two tables have. Please refer to the JoinItem class. ‘link’ defines the links between two tables, not adding one to another. Please refer to LinkItem. The names of the fields to link two tables can be different, but the data type of the fields must be consistent.

The users can all the relative data saved in any non-spatial databases, can also perform some setting, such as thematic map setting in GIS. The constraint conditions of using LinkItem: the GIS data and the attribute data must have link conditions, that is the main spatial dataset and the exterior attribute table must have a link field.
JoinType

The join type between two tables. Join type is used when performing query on two joined tables, and it determines the records in the result, including LeftJoin type and InnerJoin type. See in Table 1.

<table>
<thead>
<tr>
<th>MEMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeftJoin</td>
<td>Left join. When the two tables are joined with left join type, all the records that meet query condition in the left table will be in the result dataset, and if there is no relevant record in the right table, the corresponding filed values will be empty.</td>
</tr>
<tr>
<td>InnerJoin</td>
<td>Inner join. When the two tables are joined with inner joint type, only the records exist in both tables will be returned as the result, besides meeting the query condition.</td>
</tr>
</tbody>
</table>

RepresentationElement

The RepresentationElement class. This class is used to build a RepresentationElement object, and display a Geometry object in the vector dataset. The representation is the information linking to the geometry objects. It can make the objects displaying in the layer with another style while the former geometry objects will not be displayed. It did not modify the objects in the vector dataset, only change the display mode.

The RepresentationElement object defines the display mode, contains geometry objects and the style of the objects. In addition, the representation type controls the visibility of the original geometry object, the setting style and the objects represented.

RepresentationType

The representation type. The representation type controls the visibility of the original geometry object, the setting style and the objects represented.
EditHistory

The EditHistory class. This class includes the history information of the edit operations, such as the adding, modifying and deleting operation.

EditType

Edit operation type, including the adding, modifying and deleting operation.

3.1.6 Geometry, Geometry3D

A geographical information system (GIS) differs from a general information system in that it can not only store general attribute information but also geographic information. A GIS can use graphics to display the information it stores, to achieve a better simulation of a region in the real world.

The Geometry class is the base class of all the geometric objects, abstract class. It provides the common basic properties and methods of the geometric objects. For example, in .NET, Geometry.Bounds gets the minimum rectangle that contains the Geometry object; Geometry.Style gets the style of the Geometry object. And in Java, for the Geometry class, the method getBounds() gets the minimum rectangle that contains the Geometry object; getStyle() returns the style of the Geometry object; getType() gets the geometric object type.

Diagram 14 Geometry Class Structure shows the Geometry class structure.

This Geometry class is used to describe the spatial characteristics of the geographical entity and provides the corresponding processing methods. For different geographical entities have different spatial characteristics, so the GeoPoint, GeoLine, and GeoRegion class, etc. are used to describe different geographical entities respectively.

In Objects .NET, you can use Geometry.IsEmpty to determine whether the geometric object is null. Use Geometry.SetEmpty to clear the spatial data from this geometric object, and the styles and ID of this geometric object remain unchanged. Similarly, in Objects Java, the method isEmpty of the Geometry class is used to determine whether the geometric object is null and the method setEmpty is used to clear the spatial data from this geometric object, and keep the styles and ID of this geometric object unchanged.
GeometryType

The GeometryType class. It defines a series of 2D and 3D geometries and parameterized 2D and 3D geometries, as well as other objects, such as the north arrow, scale bar, images, and maps.

Geometrist

The Geometrist class is a class for basic geometric operations that only support the operations of 2D geometry objects. There are two uses of this class. First this class can be used to determine the spatial relationship of the two geometry objects, such as determining the relationship of the two geometry objects is disjoint or contain. The other use of this class is to perform operations on the geometry objects. The current version supported operations including clip, intersect, erase, union, symmetric difference, update and identity. Different operation has different request for both the operate and operated geometry objects. You should pay attention to these requests.

GeoStyle

The Geostyle class. This class defines the symbol styles, line styles, fill patterns and their properties. For various geometric objects, their styles can be set using the GeoStyle, except for the GeoText objects, whose style can only be set with the TextStyle. This instance of this class only stores the ID value of the specified style, the detailed information of the style is stored in the marker symbol library, line symbol library and fill symbol library correspondingly. For more details, please refer to SuperMap Objects resource library in the Programming Reference. Except for the compound datasets (CAD datasets), other datasets will not store the style information of the geometric objects. There are two kinds of fill mode, one is the common fill mode and the other is the gradient fill mode. In common fill mode, the shape can be filled with picture or vector symbols, etc.; while in gradient fill mode, there are four gradient fill types: linear gradient fill, radial gradient, conical gradient and foursquare gradient.

FillGradientMode

FillGradientMode defines the gradient types of the gradient fill mode. All the gradient types are gradient between two colors, from the starting color to the ending color. All the supported gradient fills are performed based on the boundary rectangle of the filled shape, that is, the bounds of the filled shape.

TextStyle
The TextStyle class. The class is used to set the style of the instance of the GeoText class.

Diagram 14 Geometry Class Structure
The TextAlignment class specifies the alignment methods of each subobject of text. The position of each subobject is determined by the anchor point and the alignment methods. When the anchor point is fixed, the alignment method determines the relative position of anchor point and subobjects, and thus the position of subobject is located.

Generally, the anchor point of subobject controls the position of label. You can set the relationship between text subobject and its anchor point in TextStyle. Now, 12 kinds of relationships between text subobject and its anchor point are provided. To the anchor point, a label of text can be aligned to its Top Left, Top Middle, Top Right, Middle Left, Center, Middle Right, Baseline Left, Baseline Center, Baseline Right, Bottom Left, Bottom Middle, and Bottom Right. The classes, GeoArc, GeoPoint, GeoLine, GeoRegion, GeoLineM, GeoPie, GeoChord, GeoCurve, GeoCardinal, GeoText, GeoCompound, GeoBSpline, GeoPicture, GeoNorthArrow, GeoMapScale, GeoMap, GeoRectangle, GeoRoundRectangle, GeoCircle, GeoEllipse, and GeoEllipticArc all inherit from the Geometry class. GeoArc, GeoPie, GeoChord, GeoCurve, GeoCardinal, GeoText, GeoCompound, GeoBSpline, GeoRectangle, GeoRoundRectangle, GeoCircle, GeoEllipse, and GeoEllipticArc, are parameterized 2D geometries, inherited from Geometry class. Among them, GeoLine and GeoRegion object are expressed by one or multiple ordered set of coordinate. GeoPoint is a 0-dimensional geometry with its x, y coordinates; GeoLineM is a linear referenced line object with its x, y coordinates, and its measure (M) value. The M value is the linear measurement of a line object, which represents the positions of different points in a line for transport network analysis. GeoText is used to annotate geometries and necessary explanations; GeoMap is used to add maps to layouts; GeoNorthArrow is used to add north arrows to layouts; GeoMapScale is used to add scale bars to layouts; GeoPicture is a picture.

In addition, the 3D geometry object inherits from Geometry class and its base class is Geometry3D, which inherits from Geometry class. Their relationship is shown in Diagram 15.

**GeoStyle3D**

The GeoStyle3D class. This class is used to set the display style of the Geometry3D object, including the height methods, the base height, the stretch height, textures and the surface fill method, etc.

**GeoModel**

The GeoModel class. This class is used to build a model of the Geometry3D objects and to set display styles of the Geometry3D objects.
The classes GeoPoint3D, GeoLine3D, GeoRegion3D, GeoBox, GeoPlacemark, GeoPie3D, GeoCircle3D, GeoPyramid, GeoPieCylinder, GeoCylinder, GepCone, and GeoText3D all inherit from the class Geometry3D. GeoBox, GeoPie3D, GeoCircle3D, GeoPyramid, GeoPieCylinder, GeoCylinder, and GepCone are all parameterized geometries; GeoLine3D and GeoRegion3D are represented using one or more ordered sets of coordinates \((x, y, z)\); GeoText3D is used to annotate 3D geometries and necessary explanations; GeoPlacemark is used to lay the geometry objects in specified locations and label their name in a 3D window.

Diagram 15 Geometry3D Class Structure

### 3.1.7 Other Classes

Point2D, PointM, BoundingBox, Rectangle2D and Size2D are construction classes.
Diagram 16 Other Classes Structure
Point2Ds

The Point2Ds class. This class manages all the nodes of 2D line geometry objects and their subobjects. The 2D line geometry objects and their subobjects have directions, so the Point2D object is an ordered set of points. Point2D is in an aggregation relationship with Point2Ds. And Point2Ds object contains zero or multiple Point2D objects.

Point2D

This class is used to represent a two-dimensional coordinate of point in Double precision. The Point2D class, whose insane are points, is used to represent a two-dimensional coordinate of point, which is the Double type. The coordinate values of the Point2D objects can range from ±5.0*1E-324 to ±1.7*1E308, and with a precision of 15-16 significance digits.

The Point2D object is a basic for various geometric objects. For example, a GeoLine object is formed of a bunch of Point2D objects, while a GeoRegion object is formed of a bunch of Point2D objects with the same startpoint and the endpoint. So when describing the location and shape of a geometric object, the Point2D object is very important.

Besides the methods for getting and setting the values of coordinates of a point, this class also provides method that used to create new point with integer x-coordinate and y-coordinate with given point, and the method to offset this point.

PointMs

The PointMs class. It is a set of PointM objects in Double precision. The PointMs class is used to manage all the nods on a GeoLineM or a part of a GeoLineM object. PointMs is in an aggregation relationship with PointM. And PointMs object contains zero or multiple PointM objects.

PointM

The PointM class. The pointM is the point with linear Measure value. M indicates the Measure value.

Point3Ds

The Point3Ds class. This class is used to manage all the nods on a GeoLine3D or a part of a GeoLine3D object. As GeoLine3D objects or parts of a GeoLine3D object have directions, so the
Point3Ds object is a set of ordered Point3D objects. Point3Ds is in an aggregation relationship with Point3D. Each Point3Ds object contains zero or multiple Point3D objects.

**Point3D**

The Point3D class. It is a set of Point3D objects in Double precision. The Point3D class, whose insane is Point3D object, represents an ordered pair of double x-coordinate, y-coordinate and z-coordinate that defines a point in a three-dimensional plane.

Besides the methods for getting and setting the values of coordinates of the point, this class also provides methods that used to create new point with integer x-coordinate, y-coordinate and z-coordinate with given point3D, the method to offset this point3D, etc.

**Colors**

The Colors class. This class is mainly provides the color schemes and the generation of all kinds of gradient and random colors as well as the predefined gradient colors in SuperMap. This colors object can be used to set colors in thematic maps, such as ThemeUnique, ThemeRange, ThemeGraph, ThemeGraduatedSymbol, ThemeLabel, ThemeDotDensity and the properties of geometric objects.

**ColorGradientType**

The ColorGradientType class defines the types of the color gradient. The color gradient is a mixture of the various colors, which can be the gradient from start color to ending color or among the gradient colors. This color gradient can be used to set color schemes in thematic maps, such as ThemeUnique, ThemeRange, ThemeGraph, ThemeLabel, ThemeGridUnique and ThemeGridRange maps.

**LogFile**

The LogFile class. This class is used to manage the log file, including defining the log level, the storage directory of log files, the maximum storage of log files and the time interval to create log files. There are two levels of log files. Different levels record different contents. A Higher level of log file means detailed information.
Specifies the constants that define the level of the error log records

Table 2 LogLevel enumeration members

<table>
<thead>
<tr>
<th>MEMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>This level of the error log record specifies the errors brought by invalid operations.</td>
</tr>
<tr>
<td>Debug</td>
<td>This level of the error log record aims at software developers, which includes all the information of the error.</td>
</tr>
</tbody>
</table>

LogTimeInterval

Log file storage mode. It defines the storage mode of log file according to the time interval, that is, the content of log file is generated from the ending of last time to the ending of current time interval. SuperMap Objects .NET provides five modes of storage log file.

Table 3 LogTimeInterval enumeration members

<table>
<thead>
<tr>
<th>MEMBER</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Month</td>
<td>The mode that generates a storage log file per month.</td>
</tr>
<tr>
<td>Week</td>
<td>The mode that generates a storage log file per week.</td>
</tr>
<tr>
<td>Day</td>
<td>The mode that generates a storage log file per day.</td>
</tr>
<tr>
<td>Hour</td>
<td>The mode that generates a storage log file per hour.</td>
</tr>
<tr>
<td>None</td>
<td>The mode that generates a storage log file not according to the time interval.</td>
</tr>
</tbody>
</table>

License
The license class. This class provides the information about license, such as the identify code of the compute, the name of the user, the current number of the logins, and etc. Users can export the information to strings.

LicenseMode

Three types of the license mode are provided, hardware key, license file and no license.

Environment

This environment class is used to manage the configuration information about develop environment, such as setting the location folder to save the cache file and setting the equal-zero precision.

Size2D

Stores an ordered numerical pair in double precision. Usually, it stores the height and width of a rectangle.

Rectangle2D

The Rectangle2D class. It is used to represent a rectangle whose coordinates are double-precision values, that is, the x-coordinate of the left edge, the y-coordinate of the bottom edge, the x-coordinate of the right edge and the y-coordinate of the top edge are all double-precision values. The Rectangle2D object is often used to determine bounds, and often acts as the minimum rectangle that contains a geometric object, the visible bounds of a map window, the bounds of a dataset, etc. Besides, when performing some operations like selecting by rectangle or querying by rectangle, the Rectangle2D object will be useful.

This class provides methods for determining the relationship between rectangles, and methods to get the intersection and union of two rectangles.

You can zoom in/out and offset the rectangle, and also can create a new rectangle with integer boundary coordinates by performing some calculations method on the boundary coordinates of the given rectangle.

ToolKit

The toolkit class provides some assistant calculating functions, such as reloaded method to
determine whether the given value is zero.

**ProductType**

The SuperMap products. Provides the SuperMap product list for the enumeration class.

**ProductVersion**

The version of SuperMap GIS product. This class is used to get the version information of products.

**HardwareKeyType**

Hardware key type. There are five kinds of hardware key: stand-alone key, the stand-alone time key, the network key, the network time key and the invalid type.

**ColorSpaceType**

The colors in the display device, such as display, projector, which is composed of sun light, is different from these in the printing system, such as printer, printing machine, which is using paint. For the different form mode of colors, SuperMap provides two types of color space. They are RGB and CMYK. RGB is used for display system while CMYK is used for printing system.

**BoundingBox**

The BoundingBox class. This class is used to get the boundingbox of 3D objects.

### 3.2 Mapping Module

The Mapping Module supports functions including map displaying, rendering, editing, thematic mapping, etc.

#### 3.2.1 MapControl, Map, Layers

**MapControl**

The MapControl class. This class is used to provide an interface for displaying the map. A MapControl is a control that is used to display a map. It also provides an approach for the interoperability between maps and data. Each MapControl object contains only one Map object. By
MapControl, you can not only display a map, but also edit the visibility of the map, thereby edit the data referred by the map.

When a map links to a MapControl, the MapControl can display the map, and set its display properties and operations in the map object. But the setting and operations about the edits of map’s visibility will be done in the MapControl object. The difference is that the browsing operation provided in the Map class is used when the map is displayed without a Mapcontrol, known as displaying a map without an interface, and only simple panning and zoom by specified ratio are supported operation, such as zooming and panning, can be set both in the Map object or a MapControl object. The difference is that the map operations in Map object can browse the map while displaying it without an interface, and only provide simple panning and zoom by specified ratio operations; the MapControl provides more operations on zooming and panning by Action property, and also provides various selecting mode (SelectionMode), like selecting by point, rectangle, circle, etc. Diagram 17 shows the class structure related with the MapControl class In the following we will briefly introduce the functions and the relations of the classes.

**EditHandleOptions**

The EditHandleOptions class. This class is used to control the state of the handles when editing in the map window. In .NET, the class gets or sets the options of edit handle by the HandleOptions property of MapControl class. In Java, the class returns or sets the options of edit handle by the getHandleOptions() or setHandleOptions() methods of MapControl class.

**Action**

The enumeration of map operation status. It is used to complete the operation mode related with map display, including zoom in, zoom out, zoom free, pan and simple edit operations, such as create point, create line, create polygon, etc. In .NET, this class can get or set the current map operation status by the Action property of the MapControl class; In Java, this class return or set the current operation status by the getAction() or setAction() methods of the MapControl class.

**SelectionMode**

It defines the select mode when selecting features. Whether a feature is selected or not depends on the select mode and the spatial relationship between the feature and the selected region. When the feature has the same spatial relationship with the selected region, different select mode
will lead to different result. In .NET, it gets or sets the object selection mode by the SelectionMode property of the MapControl class; In Java, it returns or sets the object selection mode by the getSelectionMode() or setSelectionMode() methods of the MapControl class.

**InteractionMode**

The InteractionMode class. This class is used to define the interaction mode of the mouse and key operation by users, that is, responds to the users’ interaction operations. In .NET, it gets or sets the interaction mode by the InteractionMode property of the MapControl class; In Java, returns or sets the interaction mode by the getInteractionMode() or setInteractionMode() methods of the MapControl class.

![Diagram 17 MapControl Class Structure](image)

**TrackMode**

The TrackMode class. This class defines whether to create a feature in a layer or in the RAM, or create a GeoMap object in the CAD layer. The users can access this class by the TrackMode property of the MapControl class.

**EditHandleType**
The constant of the edit handle type.

**VectorizationSetting**

The VectorizationSetting class. This class is used to set the parameters before or in the process of tracing vectorization.

The interactive vectorization mode, or the semi-automatically tracing vectorization, traces each line of the raster data manually. If the line object is of good quality, the system will automatically trace the line until it cannot trace, then you need to trace manually continuously. In this way you can interactively digitize the rest lines until it is finished. It applies to the contour map, the hydrography map, the road network map and other maps with more line objects.

It is a non-creatable class. In .NET, accesses this class by the VectorizationSetting property of the MapControl class; In Java accesses this class by the getVectorizationSetting() or setVectorizationSetting() methods of the MapControl class.

**SnappedElement**

The SnappedElement class. This class provides the information of the snapped element. The snapped object can be the existing Geometry object, or the drawing object. The class is non-creatable class and returns the collection of the snapped objects by the GetSnappedElements method of the MapControl Class.

**SnapSetting**

The SnapSetting class. The snapping is used to snap nodes to the object in special relation with the node and within the snap tolerance. The snapping is commonly used in the vectorization of the raster data or the editing of vector layer. The proper snap tolerance is help for reducing the mistakes, such as the dangles, cutting lines, in vectorization and feature editing. Please note that the snap priority is decided by the order of the snapping mode in the object. That is, the ID value of the snapping mode, the smaller the value is, the higher priority will be. You can accesses this class by the SnapSetting property of the MapControl class.

**SnapMode**

The SnapMode class. This class includes the intelligent snapping methods provided in the SuperMap. These intelligent snapping methods can enhance the editing precision and efficiency.
Map

This Map class manages the display environment of map. A map is a dynamic presentation of geographic data, and it is often composed of one or multiple layers. A map object manages all the layers by Layers object, and a map lay must be associated with a Workspace so as to display the data in the Workspace. The setting of the map display property will work on all the layers of the map.

The Map class is a creatable class. In .NET, this class gets the map objects displayed in MapControl by the Map property of the MapControl class. In Java, it returns the map object displayed in MapControl by the getMap() method of the MapControl class. In this class, you can set various display properties of a map, such as the bounds, scale, coordinate and measurement units, coordinate system, text, etc. And you also can do some operations on the map, such as opening and closing a map, zooming and panning the map and outputting the map etc.

The Map class supports layout without an interface, that is, a map can be displayed without being associated with a MapControl. But by this way, you can only display the map and do some simple operations on the map, and cannot edit the data by map displayed by the Map class.

MapColorMode

The MapColorMode class. In .NET, this class gets or sets the color mode of the current map by the ColorMode property of the Map class; In Java, this class returns or sets the color mode of the current map by the getColorMode() or getColorMode() methods of the Map class. The color mode of a map can be multicolor mode, black and white color mode, gray color mode and reverse black and white color mode, etc. It is only for map display and only available for vector features. When the color modes converted, the thematic style will not change. The conversion of color mode accords with the color style of the thematic layer. For more information, please refer to the Programming Reference.

PDFOptions

The PDFOptions class. A map can be exported as a PDF file by the OutputMapToPDF method of the Map class. This class is used to set the parameters when outputting the map as a PDF file.

GridSetting, GridType
The GridSetting class. This class can set the corresponding properties of grid. In .NET, this class gets or sets the grid setting of map object by the Grid property of the Map class; In Java, this class returns or sets the grid setting of map object by the getGrid() or setGrid() method of the Map class.

**TrackingLayer**

The TrackingLayer class. In SuperMap, each map has a TrackingLayer object. The users can get the tracking layer of the current map by the TrackingLayer property of the Map class. To be exactly, each map has a tracking layer. The tracking layer is a blank transparent layer, and always lies on top of other layers of the map. It is used to store some graphics or text and so on temporarily during processing and analyzing. When the map is displayed, the tracking layer will always exist, and you cannot delete the tracking layer or change its position.

This class provides methods to manage the geometric objects on the tracking layer, such as adding, deleting and so on. You can classify the geometric objects by setting tag for each geometric object on the tracking layer. The tag can be think of as a description for the geometric object, and the geometric objects that will be used in the same way or have something in common can have the same tag.

**ScreenLayer**

ScreenLayer class. This class contains the related attribute information of the ScreenLayer class objects.
3.2.1 Layers, layer, Theme

Layers

The Layers class. This class manages all the layers in a Map object, and provides methods of adding, deleting, searching, moving, etc. And you can also set a Layer to editable or not in this class. In addition, the Layers class allows several layers in it to be editable at the same time, in this way, you can edit these layers at one time, and your efficiency will be improved.

A Layers object contains one or multiple type of map objects. And it provides the methods to create the common layer and thematic layer respectively. Note that you need to specify the Theme object of the thematic layer when it is created, just like the Dataset object of the thematic layer, and you cannot change this to another Theme object later. You can only modify and set the specified Theme objects.

Layer

The Layer class. This class provides a series of properties about map display and control, which
facilitate the management of a map. When a dataset is loaded to a map window, a layer is formed, which exists as a medium, not a container. So the layer is the visual display of a dataset. A layer references to one dataset. By editing on the layer, you can manage to edit the features in corresponding dataset. One or more layers overlap each other is called a map, in this sense, a map is composed of overlapped layers.

There are two kinds of layers, common layer and thematic layer. In common layer, the features in a vector data display in the same rendering style, and cells in a raster dataset is displayed using a color table. In a thematic layer, features are rendered using the specified type of thematic map. The raster data can be displayed only by the common layer. In .NET, the common layer gets or sets its style by the Layer.AdditionalSetting property, while the thematic layer gets or sets its style by the Theme class. In Java, the common layer returns or sets its style by the getAdditionalSetting() and setAdditionalSetting() methods of Layer class while the thematic layer gets its style by the Theme class.

This class provides properties (methods) control the state of the layer, like whether the layer is visible, selectable, editable, snapable and so forth, and it also provides methods to control the displaying behavior of the layer, for example, the maximum and minimum display scales and display filter and so on. In .NET, you can create an instance of this class by the Add method of the Layers class; In Java, you can also create an instance of this class by the add method of the Layers class.

**LayerSetting**

The base class for layersetting. This class is the base class that used to set the display styles of layers. The Diagram 18 shows that the LayerSettingImage, LayerSettingVector and LayerSettingGrid are inherited from the LayerSetting class. For the vector dataset, raster dataset and image dataset, the LayerSettingVector, LayerSettingGrid and LayerSettingImage classes are used to set styles for their corresponding layers. All the features in a vector layer are rendered in the same style, and in the raster layer, a color table is used to define the colors of all the cells, while for an image layer, you can set its brightness, contrast and transparency. The following is the simple introduction to the setting classes.

**LayerSettingImage**
The LayerSettingImage class. Layers are used to display dataset visually. There are two kinds of layers, common layer and thematic layer and the image dataset only can be displayed as a common layer. This class is used to set the display style of image layer. It is mainly used to set the opacity rate of the image layer.

**LayerSettingVector**

The LayerSettingVector class. This is mainly used to set the display style for the vector layer. The common vector layer draws all the features using the uniform symbol or style. If you only want to display your spatial data, and to see where each feature locates, but do not care about the difference in quantity or attribute of features, you can display the features using the common layer.

**LayerSettingGrid**

The LayerSettingGrid class. There are two kinds of layers, common layer and thematic layer. The setting of raster layer is only for the common layer. It is displayed using a color table. The ColorTable of SuperMap is displaying the pixel in RGB color system in 8 bits each. You can set the color value according to the pixel value, making the image data more visually.

**LayerSettingType**

Layer setting enumeration. This version provides 3 kinds of layer setting. The settings for vector layer, raster layer and image layer are different. For vector layer, all the elements could be displayed in the uniform style; for raster layer, the color table can be set; for image layer, the transparency can be set.

**Selection**

The Selection class. This class is used to deal with the selected objects in a map. It is closely related to the Recordset. The users can get the selection information or set the selection option on a map by the Selection class. Through the interactive conversion between Recordset and Selection, users can process the selected objects in a map. Selection is a collection which contains one or multiple elements of the selected layer. This class can be created. In .NET, gets the selection object is gotten by the Selection property of the Layer class; In Java, the selection object is returned by the getSelection() method of the Layer class.
### 3.2.2 Theme

The relation between Theme and Layer: each thematic map accords with one layer that is the display order of thematic map is not restricted by the layers. You can change the thematic map display order at will.

![Diagram 19 Theme Class Structure]

*Diagram 19 Theme Class Structure*
Theme

The Theme class. This class is the base class of all the thematic maps, such as ThemeUnique, ThemeLabel, ThemeRanges, which are all inherited from this class.

ThemeType

The ThemeType class. You can create thematic maps for both vector data and raster data, and the difference is the thematic map for vector data is created based on the field values in the attribute table, while the thematic map for raster data is created based on the pixel values. In SuperMap, there are several types of thematic maps for vector data (namely point, line, region and CAD dataset), including unique values map, ranges map, dot density map, graph map, graduated symbol map, label map and user-defined map, and types of thematic maps for raster data, including grid ranges map and grid unique values map. The following is an introduction to the different kinds of thematic maps. For more information about thematic map, please refer to "Technical Document".

ThemeUnique, ThemeUniqueItem

The ThemeUniqueItem class. This class is used to differentiate the thematic elements; the elements with the same thematic values are displayed in the same style. ThemeUnique and ThemeUniqueItem are in an aggregation relation. The ThemeUnique contains zero or multiple ThemeUniqueItem objects. The ThemeUniqueItem provides the caption, style, unique and invisible properties.

ThemeRange, ThemeRangeItem

The range thematic map class. In a Range Map, a field or an expression, which is regarded as the thematic variable, can be separated into several ranges according to a certain range method. Each feature or record is assigned into a certain range in terms of the thematic variable. The features or records in different ranges will have their own display style. Note that when the thematic variable uses the fields from an exterior table, which is connected with the main dataset by Join or Link, if you want to display the thematic map, it is required to set DisplayFilter property, or else the thematic map is unable to display the features in the exterior table. And also the method how to separate the ranges needs to be assigned. The SuperMap Objects Java/.NET provides 6 range methods: Equal interval, Square root, Standard deviation, Logarithm, Quantile and Customer Interval. The thematic variable of Ranges Map should be a numeric value.
**ThemeGraph, ThemeGraphItem**

The ThemeGraph class. The Graph thematic map shows the thematic value of each feature or record by drawing a graph for them. The Graph thematic map can use multiple variables, and present several attributes, which is also means that you can draw several thematic values on a thematic graph. In a thematic map, you can set the display style of labels. Five label formats of thematic graph map (ThemeGraphTextFormat) are supported in this version; they are Percent, Value, Caption, Caption and Percent, Caption and Value. The thematic graph types (ThemeGraphType) provided in SuperMap Objects .NET are as following: Area, Step, Line, Point, Bar, 3D Bar, Pie, 3D Pie, Rose, Stacked bar, 3D Stacked bar, Pyramid bar and Pyramid Polygon and Ring.

The GraduatedMode class is mainly used in graph map and graduate symbol map. There are three classification methods: constant, logarithm and square root. To those negative fields, cannot use logarithm and square root. To those negative fields, the logarithm and square root cannot be used.

ThemeGraph and ThemeGraphItem are in an aggregation relation. A ThemeGraph object contains zero or multiple ThemeGraphItem objects. The ThemeGraphItem class is used to set the item caption, the field or expression used to create the map, the item uniform display style and the range setting for each item.

**ThemeDotDensity**

The ThemeDotDensity class. In a thematic dot density map, a number of dots in the same size, are used to show the distribution range, the properties of the dataset and the distribution density. When creating a dot density map, you specify how many features each dot represents and how big the dots are depending on what pattern of map represented. Note that when the thematic variable uses the fields from an exterior table, which connected with the main dataset by Join or Link, if you want to display the thematic map, it is required to set DisplayFilter property, or else the thematic map is unable to display the features in the exterior table. In .NET, you need to set the Layer.DisplayFilter when display the thematic map; In Java, you need to set the setDisplayFilter() of the Layer class.

**ThemeGraduatedSymbol**

The ThemeGraduatedSymbol class. In SuperMap Objects Java/.NET, graduated symbol map uses a group of grading symbols to show the quantity relation between different features by some grading methods. Note that when the thematic variable uses the fields from an exterior table, which
connected with the main dataset by Join or Link, if you want to display the thematic map, it is required to set DisplayFilter property, or else the thematic map is unable to display the features in the exterior table. In .NET, you need to set the Layer.DisplayFilter when display the thematic map; in Java, you need to set the setDisplayFilter() of the Layer class.

**ThemeGridRange, ThemeGridRangeItem**

The ThemeGridRange class. The raster ranges map is similar to the ranges map, and the difference is that the ranges map is used for vector data while the raster ranges map is used for raster data. Each ThemeGridRange object contains zero or multiple ThemeGridRangeItem objects. The raster ranges map separates the entire raster values into different ranges according to a certain range mode. And each range is specified by the start value, the end value, name and color. The range of each subsection is [Start, End).

**ThemeGridUnique, ThemeGridUniqueItem**

The ThemeGridUnique class. In a raster unique thematic map, the pixels with the same value will in a group, and a color is assigned for each group to distinguish different group. A ThemeGridUnique object contains zero or multiple ThemeGridUniqueItem objects.

**ThemeLabel, ThemeLabelItem**

The ThemeLabel class. The class is used to label the layer in map with the text, that is, displays the data of the attribute table on layer. You can set the style and location of the label in the label map. You can set the uniform style and use the range method to set the style of each item for displaying the label. Each ThemeLabel object contains zero or multiple ThemeLabelItem objects.

In thematic map, the label background shape (LabelBackShape) could be rectangle, roundrectangle, diamond, ellipse, triangle, symbols, etc. In SuperMap Objects .NET, the background shape can be set by the BackShape property of the ThemeLabel class; in SuperMap Objects Java, the background shape can be set and gotten by the setBackStyle and getBackStyle methods of the ThemeLabel class.

The over-length label processing mode (OverLengthLabelMode) provides three displaying methods in the label map. You can get and set the maximum length of the label through the MaxLabelLength property of the ThemeLabel class in SuperMap Objects .NET. And in SuperMap Objects Java, you can get and set the maximum length of the label through the getMaxLabelLength
and setMaxLabelLength methods of the ThemeLabel class. You can also set the label along line
direction through the AlongLineDirection property of the ThemeLabel class.

**MixedTextStyle**

The MixedTextStyle class. This class is used to set the display styles for the texts in the label map. In
SuperMap Objects .NET, you can set the uniform style for the tests in the label map by the
UniformMixedStyle property of the ThemeLabel class. And you also can get and set the uniform
style for the tests in the label map by the getUniformMixedStyle and setUniformMixedStyle of
ThemeLabel class.

**LabelMatrix**

The LabelMatrix class. This class can create a complex label to label object. For each cell of
LabelMatrix, it can be subdivide as a matrix label. Currently, the LabelMatrixImageCell,
LabelMatrixSymbolCell and ThemeLabel are supported. You can get or set the LabelMatrix by the
Labels property of the ThemeLabel class. ThemeLabel contains one LabelMatrix object, and one
LabelMatrix object contains zero or one label map objects.

**LabelMatrixImageCell**

The LabelMatrixImageCell class. The object of this class can be as a cell of the LabelMatrix object,
that is one LabelMatrixs object contains zero or multiple LabelMatrixImageCell objects.

**LabelMatrixSymbolCell**

The LabelMatrixSymbolCell class. The object of this class can be as a cell of the LabelMatrix object,
that is one LabelMatrixs object contains zero or multiple LabelMatrixSymbolCell objects.

### 3.3 SpatialAnalyst Module

The SpatialAnalyst module supports most of the spatial analyses, including buffer analysis, overlay
analysis, proximity analysis, raster surface analysis, etc.
3.3.1 BufferAnalyst

Buffer analyst is the fundamental spatial analyst in GIS. A buffer zone is actually a zone of a specified distance around the point, line or region. Buffer analyst can be used in many fields, such as to determine the area of the road to widen, the influenced area of a radiation centre. As shown in Diagram 20, SuperMap Objects Java/.NET provides the following related classes:

BufferAnalyst

The BufferAnalyst class. This class is used to create buffers for datasets and recordsets.

BufferAnalystGeometry

The BufferAnalystGeometry class. This class is used to create buffer areas for specify the geometry objects.

BufferAnalystParameter

The BufferAnalystParameter class. This class provides necessary parameters for the buffer analyst, including the end type and the radius of a buffer.

BufferEndType

The BufferEndType class. This class defines the shape of the line end points. Two types are provided for the line end points, Round or Flat.

3.3.2 OverlayAnalyst

Overlay analysis is a very important function for spatial analysis in GIS applications. Overlay analysis
means to get a new dataset by doing a series aggregate calculation between two datasets in the same reference system. And overlay analysis is widely used in resource management, evaluation of urban construction, land management, agriculture, animal husbandry, statistics and other fields. Thus you can process and analyze the spatial data, extract the new spatial information you want by using this class and deal with the attribute data.

As shown in Diagram 21, SuperMap Objects Java/.NET provides the following classes related to overlay analysis.

![Diagram 21 OverlayAnalyst Class Structure](image)

**OverlayAnalyst**

This OverlayAnalyst class. This class is mainly used for performing overlay operations between the input and overlay features, for example, union, clip, intersect, erase, identity, union, Symmetrical difference and update.

**OverlayAnalystParameter**

This OverlayAnalystParameter class. This class provides an interface to select the fields, which will be used in the result dataset during the overlay analyst. There are at least three datasets involved when performing the overlay analysis; they are the input dataset, the overlay dataset and the output dataset. The type of the input dataset (also called the first dataset in SuperMap GIS) can be point, line or region, the type of the overlay dataset (also called the second dataset in SuperMap GIS) should be region. The attribute information of the output dataset comes from the input dataset and the overlay datasets. And the users can choose the fields to remain in the attribute tables of the first two datasets at will.

### 3.3.3 ProximityAnalyst

Proximity is another of the most basic spatial analysis utilities in GIS. The proximity analysis is used
to discover the spatial proximity relationship among features. The ProximityAnalyst class provides the methods to create the Thiessen polygons. The Thiessen polygons are constructed based on a set of points and thus get the proximity relation of the points.

The Thiessen polygons divide the available space and allocate it to the nearest point and any location within a polygon is closer to its associated point than to the point of any other polygons.

Diagram 22 ProximityAnalyst Class Structure

ProximityAnalyst

The ProximityAnalyst class. This class is used for the proximity analysis. The ProximityAnalyst class mainly provides some methods to create the Thiessen polygons.

3.3.4 Interpolator

Spatial interpolation is based on the notion that points which are close together in space tend to have similar value attributes. This is known as spatial autocorrelation. For example, it is raining on one side of the street, and then is almost raining on the other side of the street. Put it in a larger region. The weather in a town is more or less like the weather in the neighboring town. Interpolation is used to estimate a value of a variable at unsampled locations from measurements made at other sampled sites.

The classes shown in Diagram 23 are provided in SuperMap Objects Java/.NET to perform spatial interpolation.
Interpolator

The Interpolator class. This class performs interpolation analysis on discrete points, and returns a grid dataset created by the interpolation analysis. InterpolatorAnalyst can interpolate the sampled points, forecast the values near the sample points, and thus to get an understanding of the whole study area. The sampled points do not only tell the data values in the sampled area, but reflects the data whole region.

InterpolationParameter

The InterpolationParameter class. This class is the base class of interpolation parameters class. It supports the necessary parameters of interpolation, such as resolution, search method, search radius, the number of expected points, and the bound of the interpolation.

InterpolationIDWParameter

The InterpolationIDWParameter class. This class inherited from InterpolationParameter class provides the necessary parameters for the Inverse Distance Weighted method. These parameters include the power of the distance, search method, resolution used in the interpolation, search radius, etc.

InterpolationKrigingParameter

The InterpolationKrigingParameter class. This class inherited from InterpolationParameter class
provides the necessary parameters for the Kriging interpolation method, such as VariogramMode, Range, Sill and Nugget.

**InterpolationRBFParameter**

The InterpolationRBFParameter class. This class inherited from InterpolationParameter class provides necessary parameters for Radial Basis Function (RBF) interpolation method, which is inherited from InterpolationParameter class.

### 3.3.5 SurfaceAnalyst

The SurfaceAnalyst class. By performing the surface analysis, you can extract information from the original data, make some specific features clear and easy to analyze.

![Diagram 24 SurfaceAnalyst Class Structure](image)

**SurfaceExtractParameter**

The SurfaceExtractParameter class. This class provides parameters for extracting isolines and isoregions, including the interval, base value, smoothing method, smoothness, resampling tolerance, etc. Two attributes, the interval and base value, controls the value and count of the isolines in the result.

**SmoothMethod**

The SmoothMethod class. This class is used for smoothing the outline of the isoline or the Isoregion generate from Grid or DEM dataset.

### 3.4 Layout Module

SuperMap Objects Java/.NET provides the classes shown in Diagram 25 to carry out the layout.
The MapLayoutControl class. This class provides an interface for display the layout and a method to the layout interaction.

MapLayout

The MapLayout class. This class provides properties and methods to perform the layout setting and print. The users can get the MapLayout object by the MapLayout property of the MapLayoutControl class.

LayoutElements

The LayoutElements class. This class is used to add layout elements, including map, north arrow, legend, scale, some geometry objects. All the objects in the layout are managed by this class. The users can get the LayoutElements object by the .Elements property of the MapLayout class.

LayoutSelection

The LayoutSelection class. You can select any elements of the layout, get the selection set of the layout elements, and carry out operation on layout elements. For example, you can set the elements styles. The users can get the LayoutSelection object by the MapLayout.Selection property.
The RulerSetting class. The ruler is a tool which helps the layout. The RulerSetting class cannot be created. Generally, in .NET, the object of this class can be gotten by the RulerSetting property of the MapLayout class; In Java this class can be returned by the getRulerSetting () method of the MapLayout class.

RulerSetting class defines two properties: IsRulerVisible is used to get or set whether the ruler is visible; Unit is used to get or set the unit of the ruler.

RulerLines

The RulerLines class. This class cannot be created. In .NET, this class can be gotten by the
MapLayout.RulerLines property; In Java, this class can be returned by the MapLayout.getRulerLines() method. One RulerLines object contains zero or multiple RuleLine objects.

RulerLine

The RulerLine class. This class includes the setting information about the RulerLine object. The RulerLine object is used to help layout.

RulerLineType

The RulerLineType class. There are two kinds of rulerline: the horizontal rulerline and the vertical rulerline.

Paper

The Paper class. This class is used to set the paper background, paper size, paper orientation, paper margins and so on. In .NET, the object of this class can be gotten by the Paper property of the MapLayout class. In Java, it can be returned by the getPaper() method of the MapLayout class.

Printer

The printer class. It is used to set the parameters related to printing, including the paper margin, paper orientation, paper size and so on. In .NET, the object of this class can be gotten by the Printer property of the MapLayout class. In Java, it can be returned by the getPrinter() method of the MapLayout class.

PaperMargin

The PaperMargin class. This class defines the top, bottom, left and right margins of the page. In .NET, the object of this class can be gotten by the Margin property of the Paper class and the Printer class. In Java, it can be returned by the getMargin() method of the Paper class and the Printer class.

PaperOrientation

The PaperOrientation enumeration class. There are two kinds of paper orientations: Portrait and Landscape.
3.5 **Realspace Module**

The Realspace module provides several functions, including 3D visualization, 3D editing, 3D thematic mapping, etc.

### 3.5.1 Scene, SceneControl

![Diagram 26 Scene Class Structure](attachment:scene_class_structure.png)
Scene

The scene class. The comprehensive applications of terrain, image and vector are gathered in the 3D scene. In 3D analysis, it contains the 3D data documents that can be viewed in the perspective.

Generally, 2D points and lines, 3D points and lines, regions, text, DEMs, Grids, TINs, images can be viewed in a 3D scene. You can also get and set the camera object, the flying speed, the 3D tracking layer object and 3D layers object, and whether the grid of longitude and latitude, the compass, and the scale are visible. You can save the fog effect, the light source, etc.

This class can be created. In .NET, it can be accessed by the Scene property of the SceneControl class; in Java, it can be accessed by the getScene() method of the SceneControl class.

The following is the introduction to the classes related to the Scene class.

SceneControl

The SceneControl class. The SceneControl is used to view a 3D scene, control 3D layers, perform 3D flying, flooding and other operations, set the properties, etc.

The SceneControl window corresponds to a 3D scene, that is, the 3D scene displays in a SceneControl window.

The SceneControl is used to represent the 3D modes in a visual format. The SceneControl allows you to view data in three dimensions and find information that is not apparent on 2D maps. For this, the 3D visualization is applied in public health, geological survey, etc. Using 3D visualization, you can represent and view spatial data in a more realistic and intuitive way.

Action3D

The Action3D class. This class includes the constant of the operation status in the 3D map. In .NET, you can get or set the 3D map operation status by the Action property of the SceneControl class; in Java, you can return or set the 3D map operation status by the getAction() or setAction() method of SceneControl class.

NavigationControl

The NavigationControl class. By using this class, you can zoom in/out on the 3D map, pan the 3D map, control the flying state, and etc. In .NET, the NavigationControl object can be gotten by the
NavigationControl property of the SceneControl class; in Java, the NavigationControl object can be returned by the getNavigationControl() method of the SceneControl class.

**Camera**

The Camera class. It is mainly used to define the object and manipulate the perspective of a 3D display. According to the principle of camera, the Camera object is used to control the position and the shape of view cone in order to simulate the observer.

**FlyingMode**

The flying mode constant. It is used to get or set the flying mode to the specified point.

**Fog**

The Fog class. This class includes the setting information of the Fog object. The setting may act on the visual effect of the fog. And this class can be created. In .NET, the users can get the fog object by the Fog property of the Scene class. In Java, the users can return the fog object by the getFog() method of the Scene class.

**FogMode**

The FogMode constant class. This class includes three fogmode constants, and each fogmode constant represents a fog density calculating mode.

**Atmosphere**

The Atmosphere class. This class is used to simulate the atmosphere and to make it look realistic like the third dimension. In .NET, the users can get or set the Atmosphere object by the Atmosphere property of the Scene class; in Java, the users can return or set the atmosphere object by the getAtmosphere() or setAtmosphere() methods of the Scene class.

**LatLonGrid**

The LatLonGrid class. This class is used to set whether the latitude longitude grid object and text object on it is visible. In .NET, the users can get the LatLonGrid object by the LatLonGrid property of the Scene; in Java, the user can return the LatLonGrid object by the getLatLonGrid() method of the Scene class.

The latitude longitude coordinate is used to describe the position of the geographical object on the
earth. In the spherical system, ‘horizontal’ or east–west lines are lines of equal latitude or parallels. ‘Vertical’ or north–south lines are lines of equal longitude or meridians. These lines encompass the globe and form a gridded network called a graticule.

**TrackingLayer3D**

The TrackingLayer3D class. In .NET, the users can get the TrackingLayer3D object by the TrackingLayer property of the Scene class. In Java, the users can return the TrackingLayer3D object by the getTrackingLayer() method of the Scene class.

**ScreenLayer3D**

The ScreenLayer3D class contains the related property information of screenlayer3D objects.

**TerrainLayers, TerrainLayer**

The TerrainLayers class is the set of the TerrainLayer class. The TerrainLayers class contains the related information of the TerrainLayer object. You can manage the collection of TerrainLayer objects in one 3D map by setting this class. The TerrainLayer class contains the related property information about the TerrainLayer object.
### 3.5.2 Layer3Ds, Layer3D, Theme3D

**Layer3Ds, Layer3D**

The Layer3Ds class is the set of the Layer3D classes. The Layer3Ds class is used to manage all the 3D layers in one 3D map. The Layer3D class provides a series of properties for the management of the 3D layers.

There is only one 3D map in a 3D map window, that is only one 3D map can be displayed in a 3D map window and a Layer3D object can only in one 3D map window. Therefore the Layer3Ds object manages all the 3D layers in the 3D map.

The Layer3Ds class is used to manage the Layer3D objects in it. You can add, remove, move and
query the Layer3D object. You are allowed to add all types 3D map, remove these layers that do not need to be displayed on the 3D map or find a 3D layer with a specified name, and change the overlay order of the 3D layers.

**Layer3DMap**

The Layer3DMap class is inherited from the Layer3D class. In this layer, a 2D map can be displayed in a 3D scene, showing the integration of 2D and 3D data. This layer can be added to the 3D scenes by the Layer3Ds.Add (String dataName, Layer3Dtype layerType, Boolean addToHead) method.

**Layer3DKML**

The KML layer class. This class is inherited from the Layer3D class.

**Layer3DImageFile**

The image file layer class. This class is inherited from Layer3D class.

**Layer3DModel**

The model cache layer class. This class is inherited form Layer3D class.

**Layer3DDataset**

The Layer3DDataset class.

**Feature3Ds, Feature3D**

The Feature3Ds class. The Feature3Ds object is the collection of the Feature3D objects. The Feature3Ds class is special. It has a tree-like structure. Each Feature3Ds object may contain zero or more subobjects, which are also the objects of the Feature3Ds class. And a Feature3Ds object can zero or more Feature3D objects.

This class can be created. In .NET, the Feature3Ds objects can be gotten by the Features property of the Layer3D class. In Java, the Feature3Ds objects can be returned by the getFeatures() method of the Layer3D class.

**Feature3DSearchOption**

The Feature3DSearchOption class. This class defines the search options when traversing the
subobjects of the Feature3Ds class and the Feature3D objects.

**Layer3DSetting**

The Layer3DSetting class. Currently, the Layer3DSettingVector class, Layer3DSettingDatasetGrid class, Layer3DSettingDatasetImage class, Layer3DSettingKml class and Layer3DSettingScreenImage class are all inherited from the Layer3DSetting class.

**Layer3DSettingVector**

The Layer3DSettingVector class. This class is used to set the display style of the vector dataset when it is added to the 3D map window as a 3D layer or get the relevant information.

**Layer3DSettingImage**

The Layer3DSettingImage class. The geographic information recorded in an image file can be displayed in a 3D map window as a 3D image layer in SuperMap.

**Layer3DSettingGrid**

The Layer3DSettingGrid class. The system loads the grid files, and displays the grid data as a 3D layer in the 3D map window. This layer is called a grid 3D layer.

**Layer3DSettingType**

The Layer3DSettingType enumeration class. Currently, there are five types of the 3D layer settings, the Layer3DSettingVector, Layer3DSettingGrid, Layer3DSettingImage, Layer3DsettingSGD and Layer3DSettingKML.

In .NET, the users can get the Layer3DSettingType objects by the Type property of the Layer3DSetting class; in Java, the users can return the Layer3DSettingType objects by the getType() method of the Layer3DSetting class.

**Selection3D**

The Selection3D class. In .NET, the user can get or set the Selection3D objects by the Selection property of the Layer3D class; in Java, the users can return or set the Selection3D objects by the getSelection() or setSelection methods of the Layer3D class.
Layer3DType

The Layer3DType enumeration class.

3.5.3 Theme3D

Diagram 28 Theme3D Class Structure

Theme3D

The Theme3D class. This class is the base class of all the 3D thematic mapping classes, Theme3DUnique, Theme3DRange, and Theme3DLabel, which are all inherited from the Theme3D class.

Theme3DType

The Theme3DType class. This class is an enumeration class for the types of 3D thematic maps. Currently, SuperMap supports three types of 3D thematic maps, the 3D Unique Values map, 3D Label map, and 3D Ranges map.

Theme3DUnique, Theme3DUniqueItem

Theme3DUnique is the set class of the Theme3DUniqueItem classes.

The Theme3DUnique class. The 3D unique values map classifies features according to the field or expression value. The features with the same value are classified into the same kind, which has the
same display style. Through Join or Link, fields from exterior tables can be used as a thematic variable. When connected with an exterior table by Join or Link, the variable in the exterior table are used to display features in a thematic map, the property Layer3DDataset.DisplayFilter is required to be set in objects .NET, and the method setLayer3DDataset.DisplayFilter() is required to be set in object Java. Or else, it will fail in output this thematic map.

The Theme3DUniqueItem class. The 3D unique values map classifies features according to the field (or filed expression) value. The features with the same value are classified into the same kind, which has the same display style. Each kind is an item of the thematic map.

**Theme3DRange, Theme3DRangeItem**

The Theme3DRange class is the set class for the Theme3DRangeItem classes. In 3D Ranges Thematic maps, the values of a specified thematic field fall into different ranges. Each range has its unique start and end value, name, and style. Each range has its range values that fall into (Start, End).

**RangeMode3D**

The RangeMode class. SuperMap provides six range methods: Equal interval, Square root, Standard deviation, Logarithm, Quantile and Customer Interval.

**Theme3DLabel, Theme3DLabelItem**

The Theme3DLabel class is the set class of Theme3DLabelItem classes.

The ThemeLabel3D class. The class is inherited from the Theme3D class. It is used to display the data in the attribute table on a 3D layer in a text format, which actually labels the 3D layer. The label of the 3D label thematic map can be numbers, letters, characters, such as the geographical name of rivers, lakes, ocean, mountains, cities, villages, the DEM values, the isoline numbers, the velocity of flow, mileage of road, mileage of navigation route, etc.

You can set the style and location of the label in a 3D label thematic map. You can set the uniform style and use the range method to set the style of each item for displaying the label. And the location of each label can be changed.
The Theme3DLabelItem class. The Theme3D label maps use thematic values to label the 3D features, and it allows the users to set the ranges. The labels in the same range are displayed in the same style. Each range is a Theme3DLabelItem object, which has its own name, style, the starting value and the ending value.

### 3.6 Network Analyst Module

Network analysis is an important part of spatial analysis. Because the network analysis is in a wide but complex use, the Network Analyst module offers advance utilities for network modeling and application in SuperMap Objects Java/.NET. TrafficTransferAnalyst, TransportationAnalyst, Facility Analysis and other analysis functions are provided in this module.

#### 3.6.1 NetworkBuilder

**NetworkBuilder**

The NetworkBuilder class. This class is a network modeling tool and can be used to create network datasets from line and point datasets based topological relationships. And the users can perform network analysis on these new datasets.

**NetworkSplitMode**

The NetworkSplitMode class is an enumeration class. It is used to control the different modes for splitting lines by lines or splitting lines by points when creating a network datasets.

#### 3.6.2 TransportationAnalyst

**TransportationAnalyst**

The TransportationAnalyst class. In real world, the network model can be divided into two types,
transportation network and utilities network. Transportation Analyst includes path analysis, Traveling Salesman Problem (TSP) analysis, service area analysis, closest facility analysis, Multiple TSP analysis and finding location analysis. Before performing transportation analyst, you should set the environment of network analyst.

**TransportationAnalystSetting**

The TransportationAnalystSetting class. The settings in transportation network analysis environment will affect the analysis result. Transportation analyst includes the path analysis, Traveling Salesman Problem (TSP) analysis, service area analysis, closest facility analysis and finding location analysis. The network dataset is different from the line dataset because it contains the subdataset of network node. The network dataset has two attribute tables, one is arc information table, and the other is node information table.

![Diagram 30 TransportationAnalyst Class Structure](image)

**TransportationAnalystParameter**

The TransportationAnalystParameter class. This class is used to set the transportation analysis parameters. You can set as follows in this class: the barrier edges, the barrier nodes, the name of
the weight field, the turn weight field, the passed point or nodes in the process of analysis. You can also set whether to keep the nodes set, the edge set, the route set and the stop set in the analysis result.

**WeightFieldInfos**

The WeightFieldInfos class. This class is a set of the `WeightFieldInfo` objects. You can add a WeightFieldInfo object to the WeightFieldInfos or remove a specified WeightFieldInfo from the WeightFieldInfos and other operations.

**WeightFieldInfo**

The WeightFieldInfo class. This class stores the information of the weight field in the network analysis. Two weight fields are provided; they are FTWeightField and the TFWeightField. The weight field is the field that represents the weight of the cost. The FTWeightField represents the cost from the start to the end of an edge while the TFWeightField indicates the cost from the end to the start of an edge.

**TransportationAnalystResult**

The TransportationAnalystResult class. By this class you can get the Route set, the node ID set, the edge set, the path guide set, the stop ID set, the weight set or the cost of each stop. Using these settings, you can make the result of the path analysis, tour path analysis, closest facility analysis and service area analysis more flexible.

**ServiceAreaResult**

The ServiceAreaResult class. This class inherits from the class `TransportationAnalystResult`.

**PathGuide**

The PathGuide class. Path guide records the path from the start stop to the end stop. It is composed of the path guide items. Each item represents an arc, a node or a stop. By this class, you can get the count of the items or get a guide item according to the specified ID number.

**PathGuideltem**

The PathGuideltem class. Path guide records the steps from the start to the end, and each step is a path guide item. By this class, you can get the ID, name, index, weight, and other settings of the
item, also you can judge whether the item represents a stop or an arc, as well as the direction type, the turn type and so on.

**SideType**

The SideType class. The class specifies a constant that defines the road side. Three types are provided; they are on the middle of the road, on the left side of the road, and on the right side of the road.

**TurnType**

The TurnType class. This class is used to get the turn type of the path, including turning around, going ahead, right turn, left turn, path end with no turn, etc.

**DirectionType**

The DirectionType class. This class is used in PathGuidItem. Five directions are available; they are the North, East, West, South and None.

**LocationAnalystParameter**

The LocationAnalystParameter class. This class provides the necessary parameter information for the location analysis, including the number of the resource supply centers, the resource allocation mode, weight, node demand, etc.

**LocationAnalystResult**

The LocationAnalystResult class. This class gives the result of the location analysis.

**SupplyCenters, SupplyCenter**

The SupplyCenters class is a set of SupplyCenter classes. The SupplyCenter class provides the source centers and is used for the location analysis in network analysis.

**SupplyCenterType**

The SupplyCenterType enumeration class. This class defines the types of supply centers, which are used in location analysis. There are three types of centers: non centers, fixed centers and optional centers. Non centers will not be considered in location analysis; fixed centers and optional centers can be used in location analysis.
SupplyResult

The SupplyResult class. This class provides the result of resource supply, including the type of source centers, ID, maximum impedance, the quantity of source and so on.

DemandResult

The DemandResult class. This class is used to return the related information of the demand result, including the ID of the demand node or edge, the ID of the supply center, the actual allocated resource value and the demand result is arc or node.

3.6.3 FacilityAnalyst

![Diagram 31 FacilityAnalyst Class Structure](image)

FacilityAnalyst

The FacilityAnalyst class. The FacilityAnalyst class is one type of the network analysis classes, mainly including the connectivity analysis and the tracing analysis.

FacilityAnalystSetting

The FacilityAnalystSetting class. This class is used to provide the necessary parameters for the facility analysis. The settings in facility analysis will influence the analysis result directly.

FacilityPathResult

The FacilityPathResult class. This class gives the result of the facility path analysis.
3.6.4 TrafficTransferAnalyst

Diagram 32 TrafficTransferAnalyst Class Structure

TrafficTransferAnalyst

TrafficTransferAnalyst class. This class provides the basic methods for traffic transfer. Three method of traffic transfer analysis are provided. The first method is used to find the transfer path according to the specified start stop and end stop; the second is used to find the stops of the specified line; the third one is used to find the lines passing the specified stop.

TrafficTransferAnalystSetting

The TrafficTransferAnalystSetting class. The environment settings of the traffic transfer analysis affects the analysis result directly. It can set the information of lines and stops, the snapping tolerance, the tolerance of two neighboring stops and the walking tolerance.

TransferLineSettings, TransferLineSetting
The TransferLineSettings class is the set of the TransferLineSetting classes. The TransferLineSetting class is used to set the dataset which the lines belong to and the fields of transfer lines information, including the name or alias of the lines, the start time of the first bus of the line, the ending time of the last bus of the line, the interval field, the fare field of the tickets and so on.

**TransferStopSettings, TransferStopSetting**

The TransferStopSettings class is the set of the TransferStopSetting classes. The TransferStopSetting class is used to get the settings of the transfer stop, including the dataset that the stop belongs to, the name field of the stop, and the alias field of the stop.

**TransferWeightFieldInfos, TransferWeightFieldInfo**

The TransferWeightFieldInfos class is a set of the TransferWeightFieldInfo classes. The TransferWeightFieldInfo class stores the related information of weight field in transfer analysis. The information includes the name of the weight field and the name of TransferWeightFieldInfo object.

You can build the bus model with multiple vector datasets which represents the bus lines. In real analysis, different weights are applied in different bus lines. Some lines apply distance weight; others apply time weight or other weight. And, the fields which contain the same weight information may have different names in different datasets. Because the TrafficTransferAnalystParameter class can only have one name which represents a weight, it makes trouble when multiple line datasets exist at the same time. To solve this problem, we use the Name property to represent the weight information of different busline datasets. It is similar to pointer. The name of the weight field is the real field name that represents the weight in the property table of the busline dataset.

In addition, this method can be used for different weights mixed together to participate in a traffic transfer analysis.

**FareFieldInfo**

The FareFieldInfo class. This class is used to get and set the fields which represent the bus fare information. These fields are included the field of the bus fare type, the range of minimum fare, the step of fare increment.

**TrafficTransferAnalystParameter**
The TrafficTransferAnalystParameter class. This class is used to set the traffic transfer analyst parameters. By this class, you can set the identify name of the weight information, the maximum time of the Transfer Guide, the weight ratio of walking to taking a bus, the ID of the start stop, the ID of the end stop, the dataset of the start stop, the dataset of the end stop, the coordinate of the start stop and the coordinate of the end stop.

TrafficTransferAnalystResult

The TrafficTransferAnalystResult class. This TrafficTransferAnalystResult object contains the TransferGuide collection. That is to say that the TrafficTransferAnalystResult class has only one property TransferGuides, and this property stores all the transfer schemes in the traffic transfer analysis result. Every element, the TransferGuide object in the collection, represents a traffic transfer scheme from the start stop to the end stop. One TransferGuide object will contain several TransferGuideltem objects which number will be gained by the Count property of the TransferGuide object. The TransferGuideltem object contains one traffic transfer or walk segment of the traffic transfer scheme.

TransferGuide

The TransferGuide class. Transfer guide records the transfer line from the start stop to the end stop. It is composed of the transfer guide items. Each item represents a transfer line. By this class, you can get the count of the items, a guide item, total distance and total weight.

TransferGuideltem

The TransferGuideltem class. The traffic transfer guide records the transfer lines from the start stop to the end stop. Each transfer line contains a corresponding TransferGuideltem object. The TransferGuideltem class can be used to get the ID of the start point, ID of the end point, the name of the stop, the ID of the transfer line, distance and weight.

TransferLineInfo

The TransferLineInfo class. This class contains the dataset which the line belongs to, the line ID, the name of the line, the time of the first bus of the line, the time of the last bus of the line, the fare of the tickets, the speed of the bus of the line and the interval between every two buses of the line, the distance of the line, the cost of the line and the number of the stops the line has.
TransferLineType

The transfer line type enumeration.

TransferStopInfo

The TransferStopInfo class. This class is used to get the information of transfer stop, including the dataset that the stop belongs to, ID, name, and alias of the stop.

FareInfo,FareType

The FareInfo class. This class is used to get the relative information about the bus fare, including the type of the bus fare, the start fare, the range of the start fare, and the step of the fare. FareType is the bus fare enumeration. This class defines the type of the bus fare; they are charging by distance, charging by stops and the single fare.

3.7 Topology Module

SuperMap Objects Java/.NET provides classes as shown in Diagram 33 to perform topology related functions, including topology validating and processing.

TopologyValidator

The TopologyValidator class. This class is used for topology preprocessing and validating.

TopologyProcessing

The TopologyProcessing class. This class is used for the topology processing.

TopologyProcessingOptions
The TopologyProcessingOptions class. This class is used to set the topology processing options.

### 3.8 Processing Module

Processing is implemented in this module. This module provides cache building for the SCM, SCI3D, and SCT data.

![Diagram 34 Processing Module Class Structure](image)

**CacheBuilderSCM**

The CacheBuilderSCM class. This class builds cache for the SCM model. Several processes are needed to use the SCM module cache. Firstly, build the SCM model, in the following record the information by using the attribute table, which will be displayed in a scene, and then build the cache, and finally add it to a 3D scene, so as to boost the browsing speed with massive 3D models.

The following steps are needed to build a SCM cache file:

1. Build the 3D model displayed in the scene. The building of this model can be finished by 3DMAX.

2. Record the following information in the attribute dataset: the location of the model, the zoom ratio of the x-coordinate and y-coordinate; the rotation angle on the X, Y, and Z direction; the full path of the sgm file. The conversion from the original model file (3ds) to the sgm file can be finished by GeoModel object.

3. Configure the relative attribute of the CacheBuilderSCM class with the specified field name, then perform Build method to create the SCM cache file.

4. Load the cache file by the method of Layer3Ds.add().
CacheBuilderSCI3D

The CacheBuilderSCI3D class. The building of the SCI3D cache is performed by this class. The SCI layer is indispensable in any 3D application. Knowing how to add SCI layer, please refer to the instruction of Layer3Ds.Add() and other methods. This kind of function is used to carry out the building of SCI3D.

The building of the SCI3D cache file includes the following steps:

1. Get the original data, the Image dataset.
2. Build the image cache by this class.

CacheBuilderSCT

The CacheBuilderSCT class is used for the building of the SCT cache file. The terrain layers are always needed to be added in 3D applications. Knowing how to add the terrain layer, please refer to the introduction of the TerrainLayers.Add() method. The CacheBuilderSCT class is used to perform the building of terrain cache file.

The process of the building SCT cache file includes the following steps:

1. Get the original terrain data, namely the grid dataset.
2. Build the cache file of the terrain data by the CacheBuilderSCT object.

3.9 Conversion Module

SuperMap Objects Java/.NET supports the conversions between multiple data formats, and the conversions of the map projections and coordinates during this processing. The module makes sure the data information is lossless or little loss.
DataExport

The DataExport class. This class provides the methods and properties for exporting the SuperMap format data to the external format data. PNG, TIF, DWG, DXF and other data format are all supported.

DataImport

The DataImport class. This class is used to import data. It can import the external format data (in file) into the formats that supported in SuperMap.
**ExportResult**

The ExportResult class. This class provides the two methods, such as SucceedSettings and FailedSettings, to get the conversion settings of the successful and the failed import tasks.

**ExportSettings**

The ExportSetting class. This class is the set of the ExportSetting objects.

**ExportSetting**

The ExportSetting class. It is the item of the ExportSettings object. The users can get the setting information of the exporter data using the provided methods and properties. And the ExportSettingDWG class, ExportSettingDXF class, ExportSettingPNG class, and ExportSettingTIF class are all inherited from this class.

**ExportSettingDWG**

The ExportSettingDWG class. This class provides the setting of exporting of the AutoDWG data. And it inherits from the ExportSetting class.

**ExportSettingDXF**

The ExportSettingDXF class. This class provides the setting of exporting of the AutoDXF data. And it inherits from the ExportSetting class.

**ExportSettingPNG**

The ExportSettingPNG class. This class provides the parameters of exporting of the PNG (Portable Network Graphic), and it inherits from the ExportSetting class.

**ExportSettingTIF**

The ExportSettingTIF class. This class provides the parameters of exporting of the TIF (Tag Image File) and it inherits from the ExportSetting class.

**ExportSettingBMP**

The ExportSettingBMP class, which provides the setting of exporting of the BMP data. This class inherits from ExportSetting class.
**ExportSettingGIF**

The `ExportSettingGIF` class. This class inherits from the `ExportSetting` class.

**ExportSettingJPG**

The `ExportSettingJPG` class. This class inherits from the `ExportSetting` class.

**ImportDataInfo**

The base class of `ImportDataInfo`. The users can add the corresponding attributes (raster or vector). The `ImportDataInfoDWG` class, `ImportDataInfoDXF` class, `ImportDataInfoGRD` class, `ImportDataInfoIMG` class, `ImportDataInfoMIF` class, `ImportDataInfoPNG` class, `ImportDataInfoSHP` class, `ImportDataInfoTAB`, `ImportDataInfoWOR` class, and `ImportDataInfoTIF` class are all inherited from this class.

**ImportDataInfoBMP**

The `ImportDataInfoBMP` class. This class inherits from the `ImportDataInfo` class.

**ImportDataInfoDWG**

The `ImportDataInfoDWG` class. This class inherits from the `ImportDataInfo` class.

**ImportDataInfoDXF**

The `ImportDataInfoDXF` class. This class inherits from the `ImportDataInfo` class.

**ImportDataInfoGRD**

The `ImportDataInfoGRD` class. This class inherits from the `ImportDataInfo` class.

**ImportDataInfoIMG**

The `ImportDataInfoIMG` class. This class can import the Erdas image data and inherits from the `ImportDataInfo` class.

**ImportDataInfoJPG**

The `ImportDataInfoJPG` class. This class inherits from the `ImportDataInfo` class.

**ImportDataInfoRAW**
The ImportDataInfoRAW class, which provides the information of the imported RAW data.

**ImportDataInfoMIF**

The ImportDataInfoMIF class. This class provides methods and properties of importing the *.mif files of MapInfo and it inherits from the ImportDataInfo class.

**ImportDataInfoPNG**

The ImportDataInfoPNG class. This class provides methods and properties of importing the PNG (Portable Network Graphic) files and it inherits from the ImportDataInfo class.

**ImportDataInfos**

The ImportDataInfos class. This class is a set of ImportDataInfo objects.

**ImportDataInfoSHP**

The ImportDataInfoSHP class. This class provides methods and properties of importing the *.shp files of ArcView and it inherits from the ImportDataInfo class.

**ImportDataInfoTAB**

The ImportDataInfoTAB class. This class provides methods and properties of importing the *.tab files of MapInfo and it inherits from the ImportDataInfo class.

**ImportDataInfoTIF**

The ImportDataInfoTIF class. This class inherits from the ImportDataInfo class.

**ImportDataInfoWOR**

The ImportDataInfoWOR class. This class inherits from the ImportDataInfo class.

**ImportResult**

The ImportResult class. This class provides two methods, SucceedSettings and FailedSettings, to get the conversion settings of successful and the failed import tasks.

**ImportSettings**

The ImportSettings class. This class is a set of the ImportSetting objects.
ImportSetting

The ImportSetting class. This class is the base class of all the import setting classes. It provides the common methods and properties. The ImportSettingDWG class, ImportSettingDXF class, ImportSettingGRD class, ImportSettingIMG class, ImportSettingMIF class, ImportSettingPNG class, ImportSettingSHP class, ImportSettingTAB class, ImportSettingTIF class, and ImportSettingWOR class are all inherited from this class. The ImportSetting object is the item of the ImportSettings class.

ImportSettingBMP

The ImportSettingBMP class, which provides the information of imported setting of the BMP data. This class inherits from ImportSetting class.

ImportSettingGIF

The ImportSettingGIF class, which provides the information of imported setting of the GIF data. This class inherits from ImportSetting class.

ImportSettingJPG

The ImportSettingJPG class, which provides the information of imported setting of the JPG data. This class inherits from ImportSetting class.

ImportSettingRAW

The ImportSettingRAW class, which provides the information of imported setting of the RAW data.

ImportSettingSIT

The ImportSettingSIT class, which provides the settings of import the SIT files. It inherits from the ImportSetting class.

ImportSettingDWG

The ImportSettingDWG class. This class provides the setting information of the imported DXF of AutoCAD data. This class inherits from the ImportSetting class.

ImportSettingDXF
The ImportSettingDXF class. This class provides the parameter setting information of the imported DXF of AutoCAD data. And it inherited from the ImportSetting class.

**ImportSettingGRD**

The ImportSettingGRD class. This class provides the parameter setting information of the imported the GIRD data. And it inherited from the ImportSetting class.

**ImportSettingIMG**

The ImportSettingIMG class. This class provides the parameters setting information of the imported Erdas image data and inherits from the ImportSetting class.

**ImportSettingMIF**

The ImportSettingMIF class. This class provides methods and properties of importing the data in MapInfo data format. It inherits from the ImportSetting class.

**ImportSettingPNG**

The ImportSettingPNG class. This class provides the information of the imported PNG（Portable Network Graphic）data. It inherits from the ImportSetting class.

**ImportSettingSHP**

The ImportSettingSHP class. This class provides the information of the imported shape file of ArcView. It inherits from the ImportSetting class.

**ImportSettingTAB**

The ImportSettingTAB class. This class provides the information of the imported TAB data. It inherits from the ImportSetting class.

**ImportSettingTIF**

The ImportSettingTIF class. This class provides the parameter setting information of Tiff/BigTIFF/GeoTIFF data. It inherits from the ImportSetting class.

**ImportSettingWOR**

The ImportSettingWOR class. This class provides the parameter of the imported workspace of
MapInfo. It inherits from the ImportSetting class.