# iServer cache mechanism

## Overview

Cache mechanism is a very effective method to improve the service accessing efficiency.

When users use GIS functions, they may encounter the following situation:

- The spatial data are complex, containing too many geometric objects, and the symbols to be rendered are complex, causing consuming a very long time to draw map. And because of a large amount of spatial data, reading data takes up large memory every time.
- The map or some areas will be accessed frequently.
- The reading operations of spatial data are more; Writing operations are less; And the map result of client is relative fixity, that is the spatial data changes less.

So SuperMap iServer provides all-around cache mechanism, including HTTP Cache, Request Cache of fully functional service and three types of map cache, to improve the accessing efficiency.

This article will help you understand the cache category, storage formats and versions through SuperMap iServer cache mechanism.

- Cache strategy
- How to open HTTP Cache
- How to open Request Cache
- Map cache
- How to use Distributed Map Tiling service

## Cache strategy

With the development of cache in SuperMap GIS 9D series products, SuperMap server product provides all- around support in cache. It supports HTTP Cache, Request Cache covering all service types, and provides tiles of raster, vector and attributes for map services.

Generally speaking, the map cache refers to map tiles. From the product 7C, besides the map tiles, the map cache also includes vector tiles and attribute tiles.

- HTTP Cache
- Request Cache covering all function services
- Map cache for three types of tiles

## HTTP Cache

SuperMap iServer support to enable HTTP Cache.

HTTP Cache saves the results of different requests to memory to implement caching. You can select whether to enable/disable HTTP Cache according to the demand. After enabling HTTP Cache, the result will be cached automatically. When client sends the same request, the server will return this cached result to improve responding speed.

SuperMap iServer enables HTTP Cache by default. Cache life cycle is 1 minute. When there are no same request in one minute, the cached result will be invalidated automatically. REST map services and REST 3D services supports the HTTP Cache.

## Request Cache covering all function services

Request Cache means the server saving the response result of the request from the client to local. When the server receives the same request later, it only directly return the cached result . Here the response result can be map tile, analysis result, 3D model, etc. Request Cache supports all-around REST services, such as map, data, analysis, and 3D, etc.

Table 1 Services and resources which support Request Cache

| Remote REST service | Resources which support Request Cache |
|---|---|
| REST map service | entireImage, image, overview, symbol, tileImage |
| REST data service | features feature statistic<br>Notes: only support the query operations and the getting operation of features. |

| | | |
|---|---|---|
| REST spatial analysis service | The BufferAnalyst resource | datasetBufferResults, datasetBufferResult, geometryBufferResults, geometryBufferResult |
| | The OverlayAnalyst resource | datasetOverlayResults, datasetOverlayResult, geometryOverlayResults, geometryOverlayResult |
| | The resource for extracting isoline | datasetIsolineResults, datasetIsolineResult, geometryIsolineResult, geometryIsoregionResults |
| | The resource for extracting isoregion | datasetIsoregionResults, datasetIsoregionResult, geometryIsoregionResults, geometryIsoregionResult |
| | The interpolation analysis resource | datasetInterpolation, datasetThiessenResults, datasetThiessenResult, interpolationIDW, interpolationIDWResult, interpolationDensity, interpolationDensityResult, interpolationRBF, interpolationRBFResult, interpolationKriging, interpolationKrigingResult |
| | The Linear Reference resource | datasetLinearReferencing, generateSpatialData, spatialDataResult |
| | The analysis result set of space relations | datasetGeorelationResults, datasetGeorelationResult |
| | The result set resource of dataset proximity analysis | datasetGeorelationResults, datasetGeorelationResult, geometryThiessennResults, geometryThiessennResult |

| | |
|---|---|
| REST transportation analysis service | TSPPaths, serviceAreas, MTSPPaths, location, closestfacilities, paths |
| REST traffic transfer service | transferSolutions, transferPath, transferStops, stopListByKeyword |
| REST 3D service | datas, data, modelIndex, config, dataversion, tiledata |

## Map Cache for three types of tiles

The map cache is mainly created by Distributed Tiling service. This function supports to split the map data into many formats and tiles, such as FastDFS, MongoDB, SMTiles, UGCV5 map tiles, SVTiles vector tiles, and UTFGrid attribute tiles.

### Map Tile

Split and store all layers in map as raster map tiles, supporting FastDFS and MongoDB distributed storage, SMTiles and MBTiles and SuperMap UGC formats.

SuperMap UGC is a general and traditional tile format in SuperMap products. The map tiles with same version can be used. Distributed tile service supports the "UGCV5" tiles, which means the 5.0 original caches.

### Vector Tile

Split and store the vector layers in maps in the format of vector tiles, supporting SVTiles format.

### Attribute Tile

Store the attribute data of vector layers in maps in the format of attribute tiles, supporting UTFGrid format.

Please refer to Map cache format for more information.

Home > Configuration and management > iServer cache mechanism > How to open HTTP Cache

## How to open HTTP Cache

SuperMap iServer supports to open HTTP Cache.

HTTP Cache saves the results of different requests to memory to implement caching. You can select whether to enable/disable HTTP Cache according to demand. After enabling HTTP Cache, the results will be cached automatically. When client sends the same request, the server will return this cached results to improve responding speed.

SuperMap iServer enables HTTP Cache by default. Cache life cycle is 1 minute. When there are no same request in one minute, the cached result will be invalidated. REST map services and REST 3D services support HTTP Cache function.

To enable HTTP Cache, doing the following:

1. Click the HTTP Cache on the **Services** page when logging in to iServer WebManager.
2. The **HTTP Cache** page provides a button to enable or disable HTTP Cache.

## How to open Request Cache

Request Cache supports all-around REST service, such as the map, data, analysis, 3D, which can improve response efficiency of the server side. Request Cache requires the service source or data source of the target services belong to REST services.

The configuration of Request Cache can be implemented by modifying the configuration parameters of service provider. The service providers support to open Request Cache are: REST map service provider, REST data service provider, REST spatial analysis service provider, REST traffic transfer service provider, REST network analyst service provider and REST 3D service provider.

Open the Request Cache by checking the "Disable cache" box. Or you can modify the <cacheDisabled> to false in XML configuration file, please refer to Configuration for Service Provider layer.

Take the REST realspace service provider as an example, the configuration steps are as follows:

1. Add the REST realspace service provider and remote service address: http://iserver.com:8090/iserver/services/realspace-sample/rest. Enable cache:

2. Add the realspace service component restRealspaceProvider-iserver. Use above service provider restRealspaceProvider-iserver and add the rest interface:

3. In the 3D scene list, it browses the scene in the format of realspace. The Request Cache folder restrequestcache will be created in %SuperMap iServer_HOME%\webapps\iserver\output\temp.

When access this service again, the system won't send request to remote REST service. It will call the local cache file to improve the accessing efficiency.

If you don't need the Request Cache, close it directly. But iServer recommends you opening this funtion, and Request Cache is open by default.

# Map cache

Map cache is pre-tiling and pre-storing the map data/image with the specified way when browsing, querying, editing and analyzing the map services, so that the server doesn't need to regenerate the same result again when it receives the same request later, thereby improving the data access efficiency. In traditional GIS map service, map cache provides the cached map tiles based on the stand-alone. However, for massive geographic data, stand-alone map service obviously can not meet online access concurrency requirements. So using cluster system with the precached map tiles in different scales is the basic method to improve map access efficiency. Howerver, the use of map caching technology as a commonly used way to improve the efficiency of online map access, its production and synchronization within the system itself is very time-consuming and error-prone.

So SuperMap iServer proposes a multi-machine parallel tiling technology based on distributed storage, which takes advantage of the cloud platform to improve the tiling efficiency and stability for the massive map data.

Production and usage of map tiles provides how to configure the cached tiles.   And Distributed Map Tiling service can help you further improve the tiling efficiency for the time-consuming tiling tasks.

All the tiles, except UGCV5 format, generated by Distributed Tiling service can be automatically used by map services. You don't need to do additional configurations. For UGCV5 format tiles, you can manually Configure map provider to use it. If you modified the default settings when tiling, such as the default storage path, or other settings, you need to prefer to Configure to use the cached tiles.

Furthermore, you can Publish map tiles directly to map services, and you also can distribute the cached tile data files and tiles set to share them offline.

And if there is a need to update the map tiles because of the change of its data, you can refer to: How to update tiles.

- Tiling principle and internal mechanism
- Tile formats
- Production and usage of Map Tile
- Configuring to use the cached tiles
- Publishing cached tiles
- How to update Map Tile
- Suggestions on using the cached tiles

# Tiling mechanism

In the traditional GIS map services, map cache indicates map tiles precached on standalone computers, and the tiles are provided to the outside world by the standalone computer. This traditional tiling technology make low use of computing resources and it takes months or even years for massive data caching, which greatly affect work efficiency. What is worse, there is no obstacle recovery mechanism, the entire tiling task needs to be redone once an error occurs.

In an era that cloud computing is developed in a fast speed, SuperMap iServer proposes a new tiling technology: Distributed Tiling Technology. Distributed tiling technology supports distributed tiling and storage management of map tiles. Multiple distributed file systems, NAS and massive data storage system (such as FastDFS, MongoDB, EMC Isilon, etc) are supported to significantly enhance the I/O speed and concurrency of map tiles.

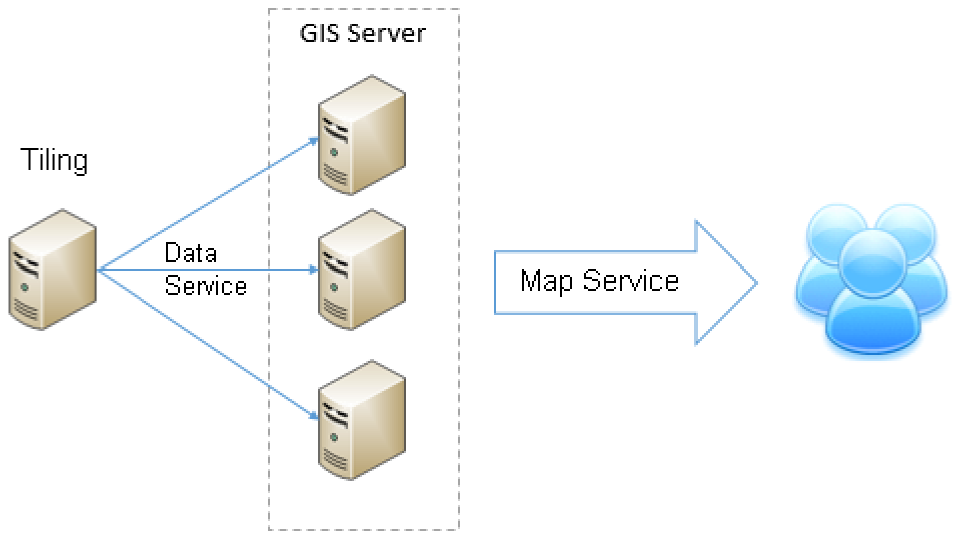
Distributed map tiling service of SuperMap iServer make use of the advantage of cloud platform to enhance tiling efficiency and stability of massive data.

- Development of tiling technology
- Distributed Tiling mechanism

# Development of tiling technology

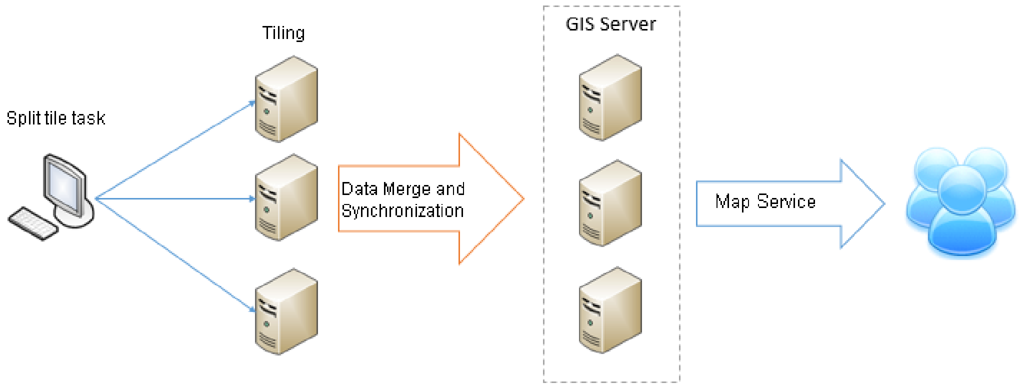
The development of of map service preaching technology goes from manual operation to automatic working, from standalone tiling, manual split multi-machine tiling, multi-machine parallel tiling with sharing directory to Distributed Map Tiling based on distributed storage.

## 1. Standalone machine tiling



Standalone machine tiling: Tiles are created in advance on a standalone machine, then the tiles are synchronized to other GIS servers in the cluster to provide map services to the outside world. This traditional tiling technology make low use of computing resources and it takes months or even years for massive data caching, which greatly affect work efficiency. What is worse, there is no obstacle recovery mechanism, the entire tiling task needs to be redone once an error occurs.

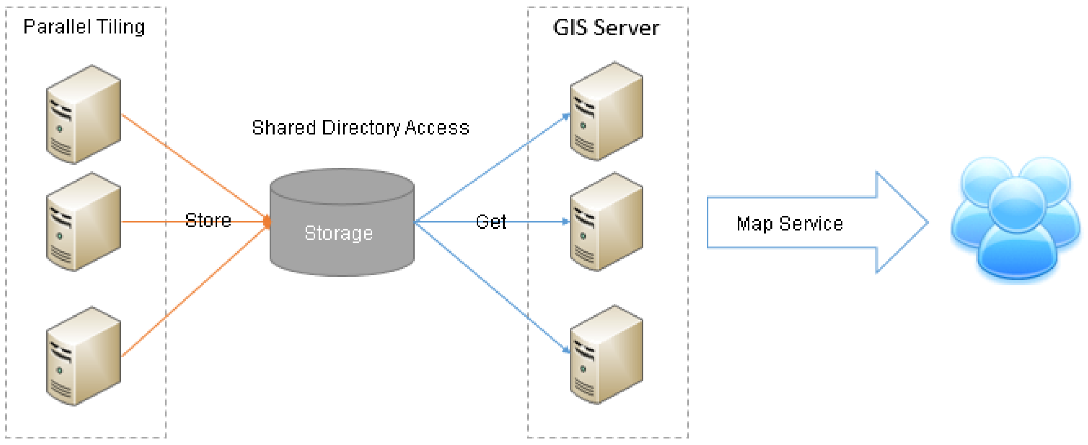
## 2. Manual split multi-machine tiling



With the development of technology, the quantity of map data is growing constantly and standalone tiling cannot satisfy the needs any more. What is more, the cost for hardware resources has reduced relatively. Considering those factors, multi-machine tiling has become the first choice of many GIS applications. In a case that there is no Distributed Map Tiling tool, the multi-machine tiling method commonly used is manually splitting the tiling task into multiple child tasks according to map scales and extent, and then deploy the tasks on different machines for tiling to realize multi-machine tiling.

As to this kind of manual splitting tiling task, following disadvantages exist: manual splitting task has high requirement and it is hard to taking all factors including data accuracy, different levels of different machines, making most use of hardware resources into consideration; each machine needs manual maintenance during tiling and if single points of failure will affect the progress of the entire tiling task; the tiling task result often needs to be synchronized to other GIS servers for providing services outside while data synchronization is a task that is time consuming and prone to cause errors; if update is needed for part of the cached data, the update will be troubling and in most occasions, caching will need to be done from start again. Therefore, an automatic process for multi-machine parallel tiling is now in high demand by all industries and this technology does not rely on manual splitting of tasks and data synchronization should avoid human intervention at the same time.

## 3. Multi-machine parallel tiling with sharing directory



In the high demand of multi-machine tiling technology, a tiling technology based on sharing directory appears to satisfy the demand in applications. The primary features of this technology are: manual splitting is not needed for coordinating multi-machine parallel tiling and human intervention is no longer needed because map stiles are stored based on sharing directory. This kind of tiling technology can satisfy the needs to some extent. However, it cannot satisfy the needs of large scale applications. In the environment with high concurrency requirement, this type of storage mechanism based on sharing directory may encounter disk IO bottleneck problem. Once an error occurs, all work needs to be paused.

All the three methods introduced above cannot meet the challenges of high data density, high computing density, and high concurrency on massive spatial temporal data faced by the GIS map services. The bottleneck of the existing Distributed Map Tiling technology is the storage method of the map tiles. Therefore, to solve the problem, the storage mechanism of the tiling technology needs to be optimized.

## 4. Distributed Map Tiling based on distributed storage

Distributed Map Tiling technology introduced above use central storage server or shared directory to store map tiles. The storage server becomes the bottleneck of the system performance, and it is also the focus of system stability and security. This method cannot satisfy the requirement of large scale storage applications. However, distributed file system supports storing data in multiple distributed machines for uniform management and can solve the problem really well. The distributed file storage system employs extensible system structure and make most use of multiple storage servers to share storage burden, which not only enhances system reliability, usability, IO efficiency, but also is easy for extension.

In a period that computing hardware is upgrading in a fast speed and cloud computing is developing quickly, distributed storage technology has become mature and has been applied in multiple industries. Currently, often used distributed file systems include GFS, HDFS, Lustre, FastDFS, PVFS, GPFS, PFS, Ceph, TFS, etc.

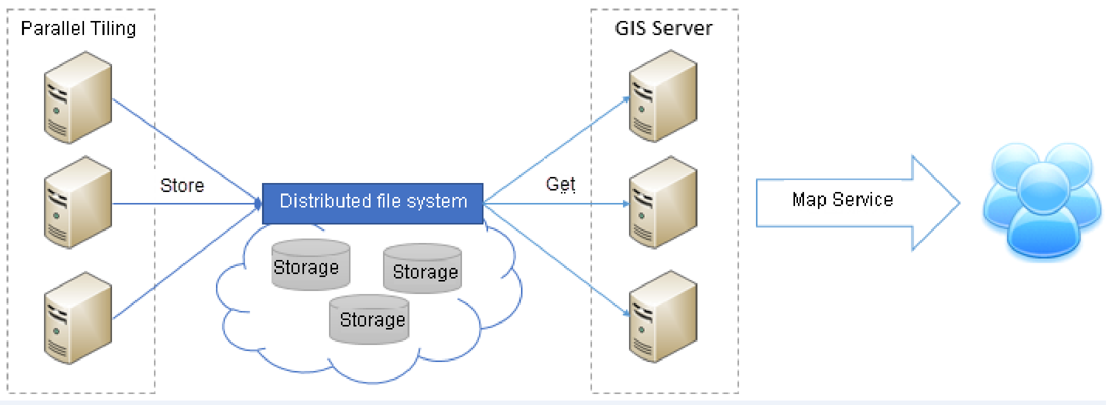
Distributed file systems have good extensibility, fault tolerance, and are transparent to internal users.

1) High usability can be realized based on redundancy recovery mechanism.

2) Hide internal storage logic and shield users and applications from differences of bottom file systems of computer modes. Uniform access interface is provided to users.

3) High extensibility. While storage capability needs to be increased, servers can be added and redesign is not needed for the storage system, which can integrate massive different storage devices in the network.

Features of distributed file system introduced above, especially advantages over shared directory (disk IO bottleneck, low reliability) enable it to become the optimal choice for map tile storage in the Distributed Tiling service.



Distributed Map Tiling technology relies on making most use of the hardware resources in the organization, implementing parallel tiling making use of multiple nodes, employing distributed file system to store tiling results, which help enhance efficiency stability, and reliability of tiling task and online services.

# Distributed Map Tiling mechanism

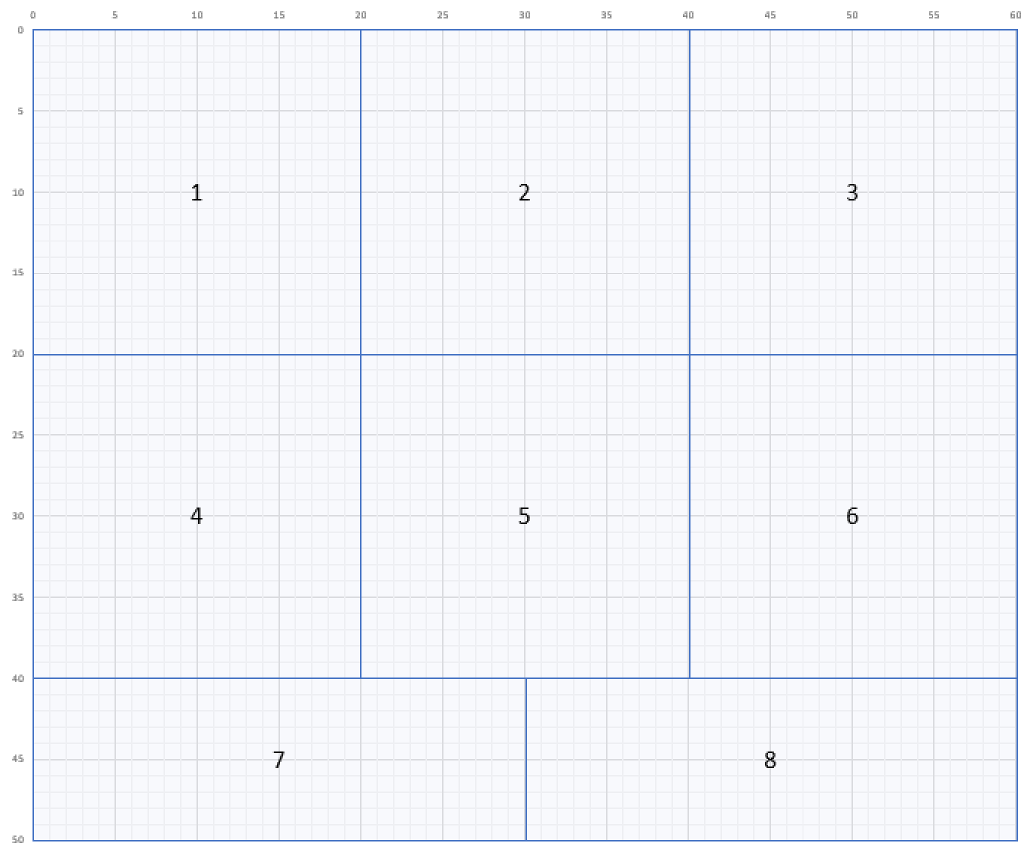
Traditional stand-alone caching technology is usually time-consuming, and the failure during tiling cannot recovery. SuperMap iServer proposes the distributed tiles generating and scheduling technology, and builds a distributed map tiles system with a TileMaster and multiple TileWorkers. Through the real-time communication with TileWorkers, TileMasters realize the dynamic allocation and real-time monitoring of the tiling task. In addition, SuperMap iServer supports distributed storage management of map tiles (presently FastDFS is supported) and a variety of distributed file systems.

## 1. Task splitting mechanism

One of the most important problems for the Distributed Tiling technology is how to completely divide the maps into several tasks, and let multiple computers work together. There are many ways of splitting tasks, e.g., a single tile is regarded as a task unit; every computer gets an average amount of tasks. However, these ways cannot meet with the parallel tiling requirements of high performance, dynamic extending, stability, etc. If a single tile is a tiling unit, because the granularity is too small, it will cause high frequency of communication and read&write among nodes, in this case, the efficiency of tiling will be sharply reducing. If the task unit is too big, it will take too long time to finish it, and it will be difficult to discover the problem occurred. Then the cost of re-tiling is too high, and the low-performance node will be a bottleneck for the tiling. So, the efficiency of tiling will also be affected.

The Distributed Map Tiling service provides an automated task-splitting strategy. According to the scale, geographic extent, tile size, and map complexity, iServer can divide big tiling task into proper grained task units. The principle of task-splitting is that: in a scale, on the basis of tile size, iServer can calculate the total tile number of row and column, then starting from the top-left corner, every N*N tiles is regarded as a task unit, in this case, most of task units can get the same workload. Yet, N is determined by the map complexity and tile size (including pixel size, file size). If the map complexity and tile size are bigger, the tile size will be smaller, and vise versa. By default, the tile format is PNG with 235*256; the reference value of N is 20, which means that there are 400 tiles in a task unit.

It is shown below. Every cell is a tile. If the task unit is 20*20, starting from the top-left corner, every 20*20 is regarded as a unit. If the final parts cannot be divided into 20*20 (e.g., in this sample, there remains 10 rows), in this case, the unit will be divided as 10*n≤400. So, in this sample, the 7th and 8th units are 10*30.



After splitting, the task units are stored in the task unit pool. TileMaster will dispatch task units to the idle TileWorkers. In general, the task splitting mechanism has following features:

1. Task units are separate and independent, so it is easier to dispatch task units to different TileWorkers.

2. The entire task are splitted automatically, even you don't need to set any parameters.

3. Task units are dispatched to TileWorkers automatically. Meanwhile, the TileMaster can monitor the status of TileWorkers.

## 2. Communication mechanism

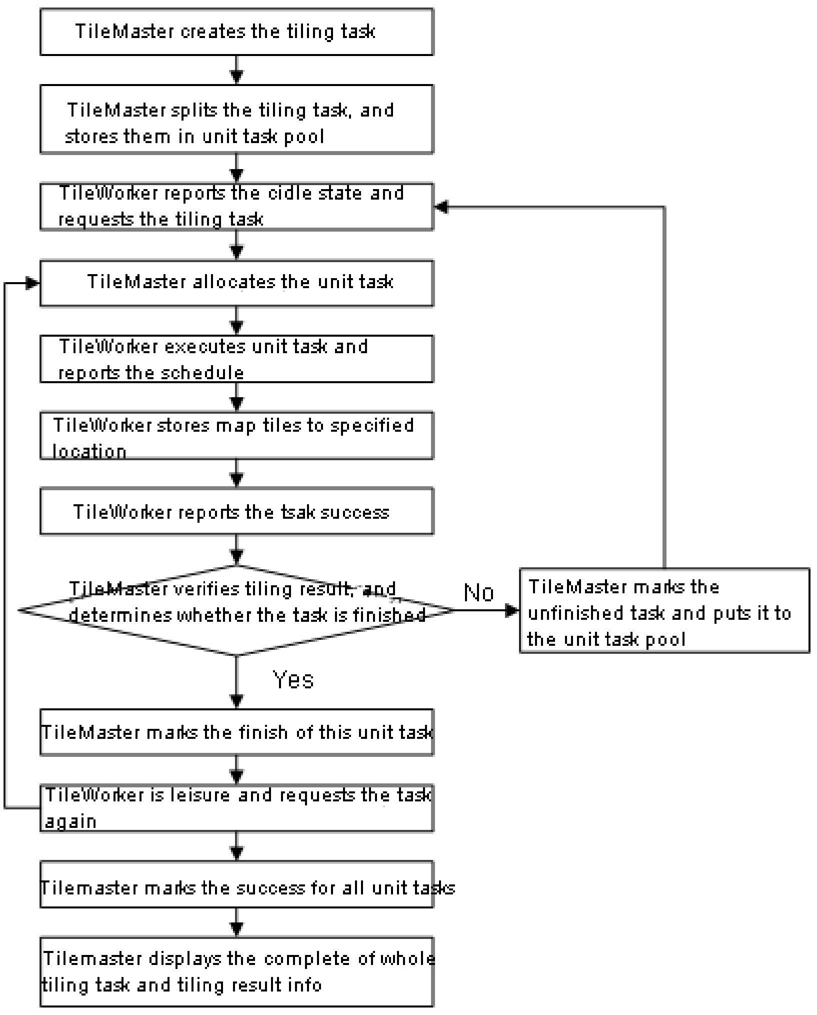
For the Distributed Map Tiling system, there are one TileMaster and multiple TileWorkers. According to the map scale and geographic extent, TileMaster divides the tiling task into multiple fine-grained task units, and dispatches task units to proper TileWorkers. Every TileWorker executes the task units and report its working status to TileMaster.

The module design of TileMaster and TileWorker:

- There are two modules of TileMaster: task management module and task scheduling module. The task management module is responsible for managing the tiling task. You can add, delete, stop the tiling via this module. The task scheduling module is responsible for splitting the tiling task into proper grained task units, and dispatching to TileWorkers for executing. Meanwhile, this module also is used to receive the information of status and workloads of Tileworkers, and balance tasks among TileWorkers.

- TileWorker mainly has the report module and the tiling module. The report module is responsible for reporting its information of status and workloads to TileMaster; if a TileWorker is idle, it will request to TileMaster. The tiling module is responsible for executing the tiling task received from TileMaster, and outputting tiles to the specified path.

The real-time messaging between TilerMaster and TileWorkers makes the tiling task dynamically dispatched and monitored. The workflow of the Distributed Map Tiling system can be described as:


TileMaster creates the tiling task
TileMaster splits the tiling task, and stores them in unit task pool
TileWorker reports the cidle state and requests the tiling task
TileMaster allocates the unit task
TileWorker executes unit task and reports the schedule
TileWorker stores map tiles to specified location
TileWorker reports the tsak success
TileMaster verifies tiling result, and determines whether the task is finished
No
TileMaster marks the unfinished task and puts it to the unit task pool
Yes
TileMaster marks the finish of this unit task
TileWorker is leisure and requests the task again
Tilemaster marks the success for all unit tasks
Tilemaster displays the complete of whole tiling task and tiling result info

1. Users send the request on tiling to TileMaster. The request parameters include map service, map name, geographic extent, scale set, tile size, tile type, etc.

2. TileMaster generate a tiling task according to the request from users.

3. According to the task splitting mechanism, TileMaster splits the task into proper grained task units, and put them to the task pool.

4. TileWorker messages to TileMaster, reporting the status of TileWorker.

5. TileWorker fetches the task unit from the task pool from TileMaster.

6. TileMaster marks that this task unit is received by TileWorker.

7. TileWorker executes the task unit, and reports to TileMaster about the execution status, and place the tiles into a specified path.

8. TileMster receives the status information from TileWorker, and displays the status information to users in real time.

9. After the task unit is finished, TileWorker sends a report to TileMaster.

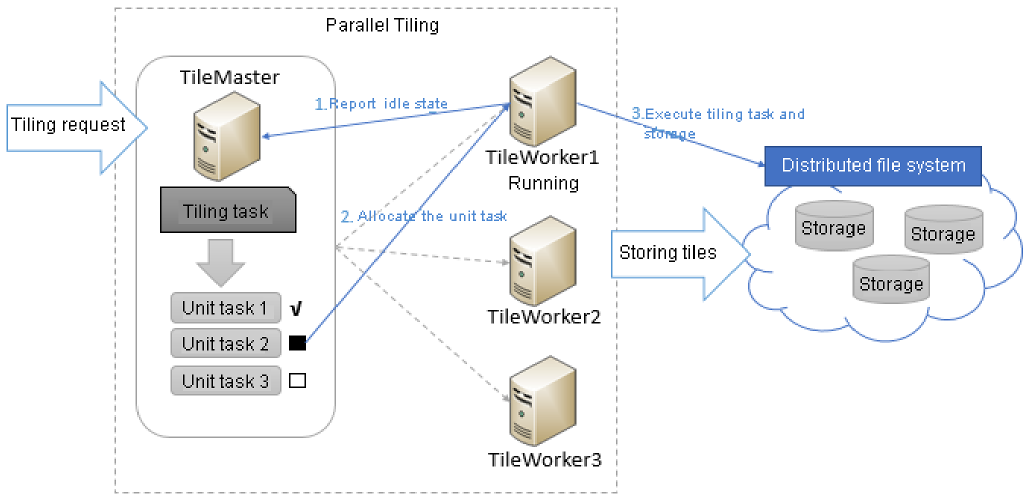
10. TileMaster verifies the tiling results from TileWorkder, and marks it as task finished.

11. After TileWorker is in idle status, it reports its status to TileMaster immediately, and repeat the steps from 4 to 10.

12. If TileMaster detects that a task unit is severely delayed, TileMaster re-marks this task unit as undispatched. Then, this task unit is received by another idle TileWorker, and follow the steps from 4 to 10.

13. After all task units are finished, TileMaster shows the task completed status and tile results information on its interface.

The dispatching and executing task units can be shown as the picture below. First, the TileWorker1 give a report to TileMaster about the idle status; As TileMaster receives the message, TileMaster dispatches the TaskUnit2 to this TileWorker1; TileWorker 1 executes the TaskUnit2, and outputs result tiles to the specified path. When TileWorker1 finishes TaskUnit2, TileWorker1 reports to TileMaster, and TileMaster checks TaskUnit2 and marks it, and dispatches TileWorker1 the next task unit.



During tiling, TileMaster monitors each TileWorkers in real time. Once TileMaster finds a task unit is pending or failed, TileMaster will dispatch this task unit to another TileWorker, which can avoid the influence on the entire tiling work due to one of TileWorkers' failure. In addition, in the Distributed Map Tiling syste, TileWorker node is flexible. That is to say, if a TileWorker is idle, it can apply for itself participating to the tiling task; Meanwhile, TileWorker can also request to cancel its tiling task.

# Tile types

With the development of cache technology, SuperMap GIS 9D products keeps on improving cache generation and cache accuracy.

In general, the map cache refers the map tiles. Starting from 9D, besides map tiles, SuperMap also provides vector tiles and attribute tiles.

- Overview of three types of tile
- Map Tile
- Vector Tile
- Attribute Tile

## Overview of three types of tile

The map caches are mainly created through Distributed Tiling service. This function enables users to tile maps into multiple types, such as FastDFS, MongoDB, SMTiles, MBTiles, UGCV5, SVTiles (vector tile), and UTFGrid (attribute tile).

| UpDownType | Tile Types | Storage Mode | Storage Path | Supported Version | Supported Platform | Distributing Approach |
|---|---|---|---|---|---|---|
| Map Tile | FastDFS | FastDFS distributed file system | FastDFS inner distributed storage. See FastDFS Installation and Configuration, and the result is a tile set in the tile library. | Yes | Linux | Support to distribute by the export of *.smtiles |
|  | MongoDB | MongoDB distributed file system | Data are stored in the specified path. See MongoDB Installation and Configuration, and the result is a tile set in the tile library. | No | Linux, Windows | Support to directly copy and distribute among MongoDB systems |
|  | OTS | OTS distributed file system | Data are stored in Aliyun OTS storage service system. | No | Linux、Windows | -- |

| | | | | | | |
|---|---|---|---|---|---|---|
| | MBTiles | SQLite data base | output path\sqlite\*.mbtiles, such as China__256X256_PNG_T.mbtiles | No | Linux, Windows | Directly copy and distribute tiles |
| | SMTiles | SQLite data base | output path\sqlite\*.mbtiles, such as China_-1085299276_256X256_PNG.smtiles | No | Linux, Windows | Directly copy and distribute tiles |
| | UGCV5 | Stored in the local disk path | output path\cache\, see SuperMap UGC Tile Version. | No | Linux, Windows | Directly copy and distribute tiles |
| | GeoPackage | SQLite data base | output path\sqlite\*.gpkg, such as ChinaProvinces_4326_256X256_PNG.gpkg | No | Linux, Windows | Directly copy and distribute tiles |
| Vector Tile | SVTiles | SQLite data base | output path\sqlite\*.svtiles, such as China_1023937971_256X256.svtiles | No | Linux, Windows | Directly copy and distribute tiles |
| Attribute Tiles | UTFGrid | SQLite data base | output path\sqlite\*.utfgrid, such as China_China_Hyd_R@China400_3857_256X256_8.utfg | No | Linux, Windows | Directly copy and |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | rid | | | distribute tiles |

Note:

- **Supported Platform**: refers to what operating systems can store current tile format.
- **Supported Version**: whether current tile format support tile set versioning or not.

From the table above, we can conclude that:

- Tile formats that support distributed storage are: FastDFS and MongoDB. Due to the distributed advantages of reading and writing, the two formats are more suitable for the distributed tiling.
- FastDFS occupies smaller disk space than MongoDB does.
- MongDB is easy to use and configure. Especially, it can be deployed on Windows.
- The format of SMTiles are convenient to copy and distribute; *.mbtiles can be used for offline status, so it is popular in mobile terminals

## Map Tile

Map tile refers to tiling maps to raster pictures. It supports FastDFS (distributed storage), MongdoDB (distributed storage), SMTiles, MBTiles, and SuperMap UGC.

SuperMap UGC type is the common and traditional type which can be used in all SuperMap products if the SuperMap UGC tile versions are the same. The UGCV5 tile is just refers to V5.0 original cache.

In addition, the picture format of the map tiles include PNG, JPG, GIF. If you choose PNG and the color value is less than 256, pictures will auto stored as PNG8 in SuperMap iServer for saving more storage space.

## Vector Tile

The vector layer can be splitted and stored as the vector tiles. iServer supports SVTiles.

In the real map service field, users not only view maps, but also query, select or highlight elements on maps. So, element service becomes a necessity. Similar to the improvement of map tile access, the rendering of elements on the client side can be improved by pre-caching vector data. Here comes the vector tile technology. The storage space of vector data is smaller than map tiles, so it is more suitable for expressing geospatial elements which requires timeliness, such as POI, routes, etc. The popular online map services, such as Google Maps, Baidu Map, use map tiles as the base map and vector tiles as element service.

## Attribute Tile

The attribute data in the vector layers can be stored as the attribute tiles. iServer supports UTFGrid.

In the real map service field, if there are massive mouse interactive operations, in the traditional way, we will overlay the element layers on the maps, and every element has its own hot spot and event. However, in the environment of massive amount of data and high concurrent requests, the clients, especially the mobile terminal, cannot smoothly render so many geographic elements. In this case, we choose another way, that is, every tile also records the attribute information. In other words, we input attribution information, divided by grids, of elements into the original tiles. This pre-stored element attribute information for each tile is so-called attribute tile. The integration of map tiles and attribute tiles has been widely used in many sectors. The most typical one is MBTiles specification and its subsidiary UTFGrid specification.

# SuperMap UGC format map tiles

SuperMap UGC is the tranditional map tile formats of SuperMap. The map tile in same version can be generic.

Table 1 The version of SuperMap UGC map tiles

| Cache version | Description | Whether to support to publish cache in static | Whether to support to create and publish cache in static | Whether to support the cache management | Default output path |
|---|---|---|---|---|---|

| | | Original | Compact | Original | Compact | Original | Compact | |
|---|---|---|---|---|---|---|---|---|
| 5.0 | The caches used by SuperMap iServer 6R(2012) SP1 or later. | √ | √ | √ | √ | √ | √ | output path\cache\{map name} |
| 4.0 | The caches used by SuperMap iServer 6R and SuperMap iServer 6R(2012). | √ | √ | √ (Default) | √ | √ | √ | Non-background Transparent: output path\cache\{map name}_{TileSize}*{TileSize}; Background Transparent: output path\cache_t\{map name}_{TileSize}*{TileSize}. Example: World_100x100, the 100 is the hex system with size of 256 tiles. |
| 3.1 | iServer 2.0 caches, namely the simple caches. | √ | | √ | | √ | | output path\cache\{map name}_{TileSize}*{TileSize}, such as World_256x256. |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3.0 | iServer 2.0 caches. | | | | | | | |
| 2.1 | The IS.NET cache use new map cache generation scheme. | | | | | | | |
| 2.0 | IS.NET cache. | | | | | | | |
| 1.0 | 3D scene caches, namely, the global subdivision 1.0 caches. | √ | √ | √ | | | | output path\{3D scene name}, such as scene_olympicgreen. |

Note:

- **Publish pre-cached tiles:** SuperMap iServer supports to publish the pre-generated map tiles as services for the client's using.
- **Create and publish caches dynamically:** It means to create and publish map tiles dynamically while browsing map. You can configure the dynamic cache version by using the parameters of UGCMapProvider.
- **Precache management:** Create and manage map tiles online through Distributed Map Tiling module.

- 5.0 version cache can be used for 2D and 3D. The local partitioning cached file can be used in 3D services.
- SuperMap UGC map tiles can be generated by Distributed Tiling service or SuperMap iDesktop, SuperMap iObjects. You can Configure to use SuperMap UGC Map tiles directly.
- In Linux system, when map services use UGCV5 format dynamic caches, or when generating UGCV5 format tiles by Distributed Tiling service, caching operation maybe fails due to the restrictions of inode. So we recommend you to clear those useless files regularly. inode is an area in computer dedicated to storing file meta information. Its size can be specified when formatting the disk. If the inode table is full, though the disk still has space, the file storage operation fails.

# MBTiles and SMTiles format map tiles

SuperMap iServer supports the map tiles that conform with MBTiles standard and an extended format based on MBTiles, SMTiles. MBTiles is a kind of standard made by MapBox that stores map tile data to SQLite database to use, manage and share.

Please refer to http://mapbox.com/mbtiles-spec/ for more information about MBTiles.

- MBTiles
- SMTiles
- Description of MBTiles and SMTiles formats
    - metadata (metadata table)
    - tiles ( tile view)
    - map (tile index table)
    - images (tile data table)

## MBTiles

MBTiles formats require the map coordinate system is: Web Mercator, namely the PCS_WGS_1984_WORLD_MERCATOR, EPSG Code: 3857. MBTiles creates and organizes the map tiles according to TMS of OSGeo, supporting the fixed scale set in Web Mercator. As shown in table 1, the initialization display level 0 in this scale contains a 256*256 tile, and define the origin point is (-20037508.34,-20037508.34), that is, the left bottom of global. MBTiles supports PNG and JPG.

Table 1 Parameter list of MBTiles display levels

| Display Level | Map width and height (pixel) | Tile number | Ground resolution (m/pixel) | map scale (96 dpi) |
|---|---|---|---|---|
| 0 | 256 | 1 | 156543.033928 | 1:591658710.909131 |
| 1 | 512 | 4 | 78271.516964 | 1:591658710.909131 |
| 2 | 1024 | 16 | 39135.758482 | 1:147914677.727283 |
| 3 | 2048 | 64 | 19567.879241 | 1:73957338.863641 |
| 4 | 4096 | 256 | 9783.939621 | 1:36978669.431821 |
| 5 | 8192 | 1024 | 4891.969810 | 1:18489334.715910 |
| 6 | 16384 | 4096 | 2445.984905 | 1:9244667.357955 |
| 7 | 32768 | 16384 | 1222.992453 | 1:4622333.678978 |
| 8 | 65536 | 65536 | 611.496226 | 1:2311166.839489 |
| 9 | 131072 | 262144 | 305.748113 | 1:1155583.419744 |
| 10 | 262144 | 1048576 | 152.874057 | 1:577791.709872 |
| 11 | 524288 | 4194304 | 76.437028 | 1:288895.854936 |
| 12 | 1048576 | 16777216 | 38.218514 | 1:144447.927468 |
| 13 | 2097152 | 67108864 | 19.109257 | 1:72223.963734 |
| 14 | 4194304 | 268435456 | 9.554629 | 1:36111.981867 |

| | | | | |
|---|---|---|---|---|
| 15 | 8388608 | 1073741824 | 4.777314 | 1:18055.990934 |
| 16 | 16777216 | 4294967296 | 2.388657 | 1:9027.995467 |
| 17 | 33554432 | 17179869184 | 1.194329 | 1:4513.997733 |
| 18 | 67108864 | 68719476736 | 0.597164 | 1:2256.998867 |
| 19 | 134217728 | 274877906944 | 0.298582 | 1:1128.499433 |
| 20 | 268435456 | 1099511627776 | 0.149291 | 1:564.249717 |
| 21 | 536870912 | 4398046511104 | 0.074646 | 1:282.124858 |
| 22 | 1073741824 | 17592186044416 | 0.037323 | 1:141.062429 |

The map tile storage format of MBTiles *.mbtiles file name is composed by map name, Hashcode, tile width and height, tile format, T (transparent) and .mbtiles suffix, such as China_69470548_256X256_PNG_T.mbtiles and China_69470548_256X256_PNG.mbtiles.

## SMTiles

In order to meet the application requirements, SuperMap iServer not only supports to create and use the MBTiles tile map data, but also supports the SMTiles whihc is extended on MBTiles. Compared to MBTiles format, SMTiles supports any coordinate system and scale. The start point of tile should be any specified point. The directory of row increase from origin to left bottom. In addition, map tile format supports PNG, JPG, PNG and JPG.

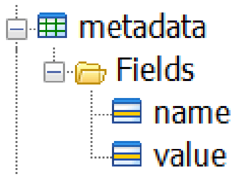
The map tile storage format of SMTiles *.smtiles file name is composed by map name, Hashcode, tile width and height, tile format, T (transparent) and .smtiles suffix, such as China_69470548_256X256_PNG_T.smtiles and China_69470548_256X256_PNG.smtiles.

## Description of MBTiles and SMTiles formats

MBTiles and SMTiles storage mechanism are basically the same. They manage the map tile data through "metadata", tiles, map datasheet, and image datasheet.

### metadata (metadata table)

It adopts the key value to store the settings of map tile data, including "name" and "value" fields, in accordance with MBTiles 1.1 specification, as shown below:



The contents of the metadata table are as shown below:

Table 2 Example of metadata table

| name | value |
|---|---|
| name | World |
| type | baselayer |
| version | 1.1 |
| ext_spec_version | 201310 |
| description | World created on 2012-12-17 16:51:22 by SuperMap iServer |
| format | PNG |
| bounds | -180.0,-90.0,180.0,90.0 |
| ext_spec_version | 201309 |
| axis_origin | -180.0,90.0 |
| axis_positive_direction | RightDown |
| crs_wkid | 4326 |
| crs_wkt | GEOGCS["WGS 84", DATUM["WGS_1984", SPHEROID["WGS 84", 6378137, 298.257223563, AUTHORITY["EPSG", "7030"]], AUTHORITY["EPSG", "6326"]], PRIMEM["Greenwich", 0, AUTHORITY["EPSG", |

| | |
|---|---|
| | "8901"]],<br>UNIT["degree", 0.0174532925199433, AUTHORITY["EPSG", "9122"]],<br>AUTHORITY["EPSG", "4326"]] |
| tile_height | 256 |
| tile_width | 256 |
| resolutions | 78271.516964,529.1666666666670 |
| scales | 5.0E-7 |
| transparent | false |
| mapStatusHashCode | -411043745 |
| map_parameter | {"scale":0.00101610071425,"clipRegion":{"center":null,"id":0,"style... |
| compatible | false |

The keys of the metadata table are as shown below:

Table 3 Description of fields in metadata table

| Field name | Required | Whether iserver extends | Description |
|---|---|---|---|
| name | Yes | No | Map English name. If the map name is Chinese, it will be changed to the Chinese Pinyin. |
| type | Yes | No | Map type. The value is overlay or baselayer. |
| version | Yes | No | The version of map tile data. MBTiles standard version is 1.1. |
| description | Yes | No | Description. |
| format | Yes | No | The format of tile data: png, jpg or jpg_png. The jpg_png is hybrid format supported by |

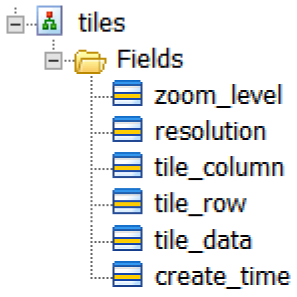
| | | | |
|---|---|---|---|
| | | | the V5cache. |
| bounds | No | No | Map tile range. The unit and map unit should be remain the same. The bounds of iServer are: left, down, right, up, such as: -180,-85,180,85. |
| ext_spec_version | No | Yes | The extended version of SuperMap iServer based on MBTiles standard. Here it is 201310. |
| axis_origin | No | Yes | The position of the tile origin. |
| axis_positive_direction | No | Yes | The enumerated type of positive direction in coordinate system (RightDown, RightUp, LeftDown, LeftUp). For example, x axis is right; y axis is RightDown. |
| crs_wkid | No | Yes | The EPSG Code of coordinate system (-1000 is the custom coordinate system; 0 is the planar coordinate system). |
| crs_wkt | No | Yes | Use wkt to represent the geographic coordinate system. (wkt: a kind of text-markup language made by OGC, please refer to http://docs.geotools.org/stable/javadocs/org/opengis/referencing/doc-files/WKT.html) |
| tile_height | No | Yes | Tile width. It is usually 256. |
| tile_width | No | Yes | Tile height. It is usually 256. |
| resolutions | No | Yes | Resolution set. The resolution corresponding to tiles of all levels. |
| scales | No | Yes | Scale set corresponding to resolution of all levels. |
| transparent | No | Yes | whether the map image are transparent. |
| mapStatusHashCode | No | Yes | The hash code of map name. For example, map World HashCode is: -411043745. |
| map_parameter | No | Yes | The json string of map default parameter. Map default parameters contain all map characteristic, such as map name, mapScale |

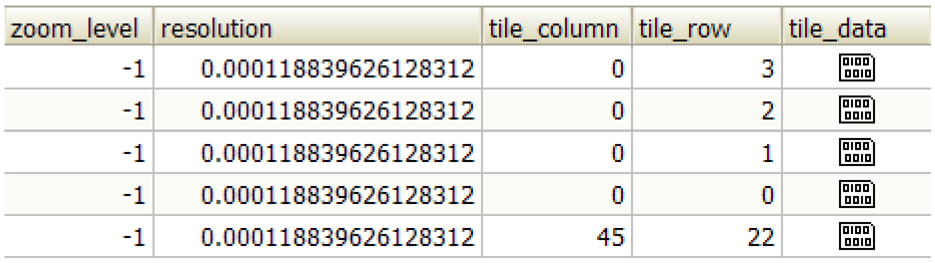
| | | | |
|---|---|---|---|
| | | | and so on. |
| compatible | No | Yes | Whether to be compatible with the MBTiles standards. When the wkid field is 3857, all resolutions in the set belong to MBTiles standard are true, otherwise, false. |

## tiles ( tile view)

Contain all tile data and values that used to locate.

MBTiles format supports four fields: zoom_level, tile_colum, tile_row and tile_data in MBTiles1.1 standard. Based on MBTiles, SMTiles adds resolution field to support scale and resolution. zoom_level are -1, as shown below.





| zoom_level | resolution | tile_column | tile_row | tile_data |
|---|---|---|---|---|
| -1 | 0.000118839626128312 | 0 | 3 | ▦ |
| -1 | 0.000118839626128312 | 0 | 2 | ▦ |
| -1 | 0.000118839626128312 | 0 | 1 | ▦ |
| -1 | 0.000118839626128312 | 0 | 0 | ▦ |
| -1 | 0.000118839626128312 | 45 | 22 | ▦ |

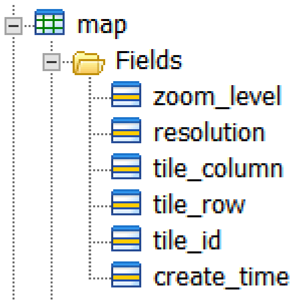
**Notes:** when the ground resolution can not correspond to the resolution in table 3, zoom_level is -1.

Using the tiles view can **reduce the redundant tiles**. In sea or clear land, it may be several pieces with small scale, but in large scale, such as 1:10000, it may be millions of the single color blue tiles. Through splitting tile index and tile storage, MBTiles use view to relate the two. So the index of these tiles point to the same tile, which can reduce the redundancy of true color.

## map (tile index table)

Include the values and tile id which are used to locate. Except the four fields in MBTiles1.1: zoom_level, tile_colum, tile_row and tile_id, it adds the resolution field. Support any scales or resolution. create_time field identifies the time of creating tile.

If the tile is the pure color, tile_id is composed of the hexadecimal values of image RGB. For example, the red image tile_id is "ff0000". Others' tile_id is the cluster of resolution_x_y.

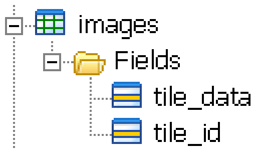


```
map
  Fields
    zoom_level
    resolution
    tile_column
    tile_row
    tile_id
    create_time
```

Figure 4 map structure chart

### images (tile data table)

Include tile id and tile data.



```
images
  Fields
    tile_data
    tile_id
```

According to resolution or corresponding scale, read tile data. For more information about MBTils scale and resolution, please refer to Table 1.

# SVTiles format vector tiles

SVTiles is a type of storage format customized by SuperMap, which stores the vector tile data with the SQLite database. The svtiles file use metadata and tiles data to manage the vector tiles.

The name of the SVTiles file is composed of the map name, Hashcode, width and height of tiles, and the .svtiles suffix like China_-678451788_256X256.svtiles.

From iServer 8C(2017) , Vector Tile adds style information. When caching vector tiles, it will generate the SVTiles file, and the style information file with ".resources" suffix to store the style attribute information of the vector tiles.

The style information file also uses SQLite database. And it is named in the same way as SVTiles.

.svtiles file and .resource file use the following tables and views to store and manage vetor tiles.

- SVTiles
    - metadata (metadata table)
    - tiles (tile data table)
    - geometries (geometric object table)
    - attributes (attribute information table)
    - tilefeatures (tilefeatures view)
    - tilegeometries (tilegeometries view)
- resource
    - style (style table)
    - Symbol (symbol table)

## SVTiles

### metadata (metadata table)

Metadata table adopts key-value pairs for storing settings of vector tiles, including name and value. The structure of the table is shown in the figure:

Below shows an examples of metadata table..

Table 1 Example of metadata table

| name | value |
|---|---|
| name | China |
| version | 201401 |

| | |
|---|---|
| bounds | -20037508.342787,-20037508.342787,20037508.342787,20037508.342787 |
| tile_origin | -20037508.342787,20037508.342787 |
| crs_wkid | 3857 |
| crs_wkt | PROJCS["User Define",<br>GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",<br>SPHEROID["WGS_1984",6378137.0,298.2572235630001,<br>AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],<br>PRIMEM["Greenwich",0.0,AUTHORITY["EPSG","8901"]],<br>UNIT["DEGREE",0.017453292519943295],AUTHORITY["EPSG","4326"]],<br>PROJECTION["Popular Visualisation Pseudo Mercator",<br>AUTHORITY["EPSG","1024"]],<br>PARAMETER["False_Easting",0.0],<br>PARAMETER["False_Northing",0.0],<br>PARAMETER["Central_Meridian",0.0],<br>PARAMETER["Latitude_Of_Origin",0.0],<br>PARAMETER["Scale_Factor",1.0],<br>UNIT["METER",1.0],<br>AUTHORITY["EPSG","3857"]] |
| tile_height | 256 |
| tile_width | 256 |
| scales | 1.690163e-9,3.380327e-9 |

| | |
|---|---|
| resolutions | 156543.033928,78271.516964 |
| geometry_storage_type | SuperMapJson |
| attribute_storage_type | Json |
| layer_infos | [<br>{"Capitals": {<br>"expand_pixels": 2<br>},<br>"Road": {<br>"expand_pixels ": 2<br>},<br>"Provinces": {<br>"expand_pixels ": 2<br>}<br>}<br>] |

Table 2 shows the fields' description of metadata table.

Table 2 Description of fields in metadata table

| Field name | Description |
|---|---|
| name | Name of map. |
| version | The version of the vector tiles, and the current version is 201401. |
| bounds | The tiling extent of the map whose unit is consistent with that of the map. Formats of bounds in iServer are left, down, right and up, like -180, -85, 180 and 85. |
| tile_origin | The position of the tiling origin. |
| crs_wkid | EPSG Code of the coordinate system (-1000 indicates the custom coordinate system and 0 indicates common |

| | |
|---|---|
| | planar coordinate system). |
| crs_wkt | The geographic coordinate system represented by wkt. (wkt: A text markup language developed by OGC. For details, please refer to http://docs.geotools.org/stable/javadocs/org/opengis/referencing/doc-files/WKT.html) |
| tile_height | The tile height, generally 256. |
| tile_width | The tile width, generally 256. |
| scales | Collection of scales corresponding to different levels resolutions. |
| resolutions | Scales set corresponding to each-level resolution. |
| geometry_storage_type | The storage formats of geometric objects. Optional values can be SuperMapJson, WKT, GeoJson, GML. |
| attribute_storage_type | The storage formats of attribute data. Optional values can be Json and Xml. |
| layer_infos | The metadata information used to describe each layer. |

## tiles (tile data table)

All vector tile data and values for locating tiles are included, as illustrated in the figure:

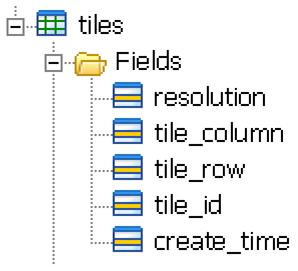


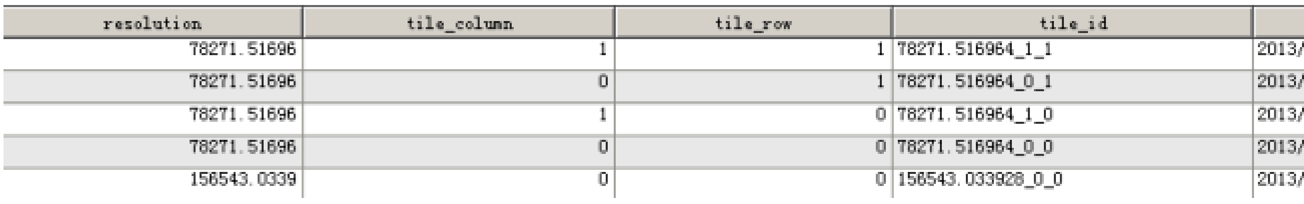
Table 3 shows the fields' description of tile data table.

Table 3 Description of fields in SVTiles tile table

| Field name | Data type | Description |
|---|---|---|
| resolution | varchar | Resolution corresponding to the tile. |

| | | |
|---|---|---|
| tile_colum | integer | The column number of the tile. |
| tile_row | integer | The row number of the tile. |
| tile_id | String | Tile ID. |
| create_time | Text | The creation time. |

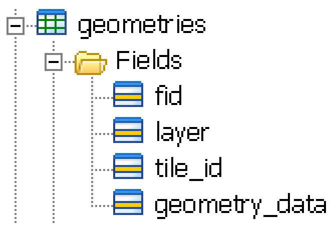
Example of tile dataset content is shown in the figure below.



| resolution | tile_column | tile_row | tile_id | |
|---|---|---|---|---|
| 78271.51696 | 1 | 1 | 78271.516964_1_1 | 2013/ |
| 78271.51696 | 0 | 1 | 78271.516964_0_1 | 2013/ |
| 78271.51696 | 1 | 0 | 78271.516964_1_0 | 2013/ |
| 78271.51696 | 0 | 0 | 78271.516964_0_0 | 2013/ |
| 156543.0339 | 0 | 0 | 156543.033928_0_0 | 2013/ |

## geometries (geometric object table)

Records the geometry information of the geographic features in tiles, as illustrated in the figure below:



- geometries
  - Fields
    - fid
    - layer
    - tile_id
    - geometry_data

Table 4 shows the fields' description of geometric object table.

Table 4 Description of fields in geometric object table

| Field name | Data type | Description |
|---|---|---|
| layer | Text | Layer name. |
| fid | Long | Geometry feature ID. |
| tile_id | Text | Tile ID. |
| geometry_data | Text | The geometry object. |

Each record corresponds to a clipped geometric object. geometry_data stores pixel coordinates. At different levels, pixel coordinates of a same geometric object are different; at the same level, one geometric object can cover multiple tiles, as shown in figure 1. The tile stores geometric object after clipping, as shown in figure 2.

Figure 1 Geometric object covering 2 tiles

Figure 2 Geometric object after clipping

The default storage format is SuperMapJSON. It can also be WKT, GeoJSON, GML. SuperMapJSON format is as follows:

```
{
“type”:”REGION”,
“points”:[0,0,256,0,256,256,0,256,0,0,1,1,3,3,2,2,1,1],
“parts”:[5,4]
}
```

## attributes (attribute information table)

Records the attribute information of the geometric objects, as shown in the figure below.

Table 5 shows the fields' description of attribute information table.

Table 5 Description of fields in attribute information table

| Field name | Data type | Description |
|---|---|---|
| layer | Text | Layer name. |
| fid | Long | Geometry feature ID. |
| attr_data | Text | The attribute data. |
| search_values | Text | Query content. |

The attribute and geometric objects are corresponding to each other. The geometric object of each geographic feature spread in different tiles (2.3), therefore, attributes and geometry has 1:many relationship. The Json format of attr_data is as follows:

```
{
“NAME”:”Beijing”,
“PostCode”:100000,
“POP”:11510000,
“Country”:”China”
}
```

search_values are strings separated by comma and it is used for fast query. For instance, search_values for Beijing is “Beijing,100000”. Query statement according to search_values is as follows:

```
where search_values like ‘%Beijing%’
```

## tilefeatures (tilefeatures view)

The tilesfeatures view generated based on tiles, attributes, geometies.

Table 6 shows the fields' description of tilefeatures view.

Table 6 Description of values of tilefeatures view

| Field name | Data type | Description |
|---|---|---|
| resolution | Double | The resolution. |
| tile_column | Long | The column number of the tile. |
| tile_row | Long | The row number of the tile. |
| tile_id | String | Tile ID. |
| create_time | Text | The creation time. |
| layer | Text | Layer name. |
| fid | Integer | Geometry feature ID. |
| geometry_data | Text | The geometry object. |
| search_values | Text | Query content. |
| attr_data | Text | The attribute data. |

## tilegeometries (tilegeometries view)

The tilesgeometries view generated based on tiles, attributes, geometries is used to completely represent the geometric information of tile and geographic features in tile. If attribute information is not required, the query based on tilegeometries will be faster, as shown in the figure below:

Table 6 shows the fields' description of tilegeometries view.

Table 7 Description of fields in tilegeometries view

| Field name | Data type | Description |
|---|---|---|
| resolution | Double | The resolution. |
| tile_column | Long | The column number of the tile. |
| tile_row | Long | The row number of the tile. |
| tile_id | String | Tile ID. |
| create_time | Text | The creation time. |
| layer | Text | Layer name. |
| fid | Integer | Geometry feature ID. |
| geometry_data | Text | The geometry object. |

- SQL manifest.

metadata

```
CREATE TABLE metadata (name text, value text);
CREATE UNIQUE INDEX metadata_idx ON metadata (name);
```

tiles

```
CREATE TABLE tiles (resolution double, tile_column integer, tile_row
integer,tile_id text, create_time
text);
CREATE UNIQUE INDEX tiles_index on
tiles(resolution,tile_column,tile_row);
Create index tiles_id_index ON tiles(tile_id);
```

geometries

```
CREATE TABLE geometries(layer text,fid long,tile_id text,geometry_data
text );
CREATE UNIQUE INDEXgeometries_index ON
geometries(layer ,fid,tile_id);
```

attributes

```
CREATE TABLE attributes( layer text, fid long, attr_datatext,search_values
text);
CREATE UNIQUE INDEX attrbutes_index ON attributes (layer, fid);
```

tilefeatures

```
CREATE VIEW tilefeatures as
SELECT A.*,B.layer,B.fid,B.geometry_data,C.search_values,C.attr_data
FROM tiles as A,geometries as B,attributes as C
WHERE A.tile_id=B.tile_id and B.layer=C.layer and B.fid=C.fid
```

tilegeometries

```
CREATEVIEW tilegeometries as
SELECT A.*,B.layer,B.fid,B.geometry_data
FROM tiles as A,geometries as B
WHERE A.tile_id=B.tile_id
```

- Comparison between SVTiles V201401 SVtiles V201310

Functionality comparison is shown in table 8:

Table 8 SVTile functionality comparison between SVTiles V201401 SVtiles V201310

| **Function** | **V201310** | **V201** |
|---|---|---|
| Geometric object storage | SuperMapJson | Extended regulation supports multiple formats |
| Feature attribute storage | Not supported | XML, Json |

Data table comparison is shown in table 9:

Table 9 Table comparison between SVTiles V201401 SVtiles V201310

| Version | Table description |
|---|---|
| V201310 | metadata metadata table. |
| | tiles tile table. |
| | tiles_data field stores all geometric object collection. |
| V201401 | metadata metadata table. |
| | tiles tile table, which only stores tile index. |
| | geometries geometry object table. |
| | attributes attribute information table. |
| | tilefeatures tile and feature view. |
| | tilegeometries tile and geometric object view. |

- SuperMapJson

SuperMapJson is a type of Json format defined by SuperMap iServer. This format is composed of 3 parts:

λ type: Geometric shape type.

Optional values are POINT, LINE, REGION, which corresponds to MultiPoint, MultLineString, MultiPolygon in WKT separately.

λ points: Represents a group of pixel coordinate sequence, each neighboring items representing a pair of coordinates.

parts: Represents the segmentation information of the current geometric object. Each segment represents a child object and the value of segment represents the number of child object points.

Example:

```
{
“type”:”REGION”,
“points”:[0,0,256,0,256,256,0,256,0,0,1,1,3,3,2,2,1,1],
“parts”:[5,4]
}
```

In the example above, “parts”:[5,4] represents that the object has two segments, and the coordinates of the two segments are [0,0 ,256,0,256,256,0,256,0,0], [1,1,3,3,2,2,1,1] respectively.

Note: For geometric object with type being POINT, json can contain no parts items. Each coordinate in points of this type of geometric object represents a child POINT.

## resource

### style (style table)

The style table is used to record each layer's style type and its style description. The fields in the table are shown below:

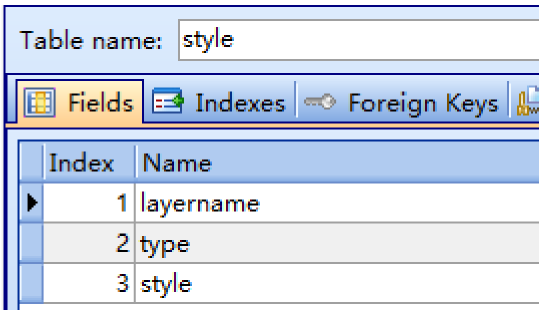


Table name: style

Fields | Indexes | Foreign Keys

| Index | Name |
|---|---|
| 1 | layername |
| 2 | type |
| 3 | style |

The meanings of the above fields show below.

Table 10 The fields of style table

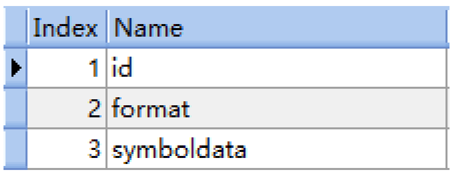
| Field name | Data type | Description |
|---|---|---|
| layer name | Text | The name of map layer. |
| type | Text | The style type. Currently, it supports CartoCSS. |
| style | Text | The style information of layer. |

Example:

```
#District@Changchun[zoom<=9.80459446371E-5][zoom>=1.05263157895E-5]{text-placement-type:simple;text-placements:"E,NE,SE,W,NW,SW";line-pattern-file:url(SYMBOLLINE__District@Changchun__64__64__true__-1483079698.png);polygon-pattern-file:url(SYMBOLFILL__District@Changchun__
64__64__true__-1483079698.png);point-file:url(SYMBOLMARKER__District@Changchun__64__64__true__-1483079698.png);polygon-opacity:1;polygon-pattern-opacity:1;}
```

## Symbol (symbol table)

The symbol table records the symbol image in tile.



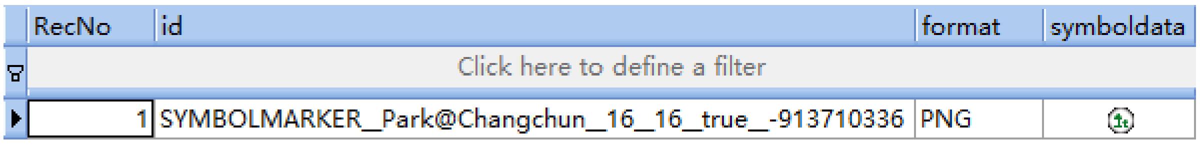
| Index | Name |
|---|---|
| 1 | id |
| 2 | format |
| 3 | symboldata |

The fields in the table are shown below:

Table 12 The fields of symbol table

| Field name | Data type | Description |
|---|---|---|
| id | Text | The identifier of the symbol. id consists of symbol type, the layer where the symbol locates, image width and height, transparent or not, and hashcode. |
| format | Text | Image format, such as PNG. |
| symboldata | blob | The image of symbol. |

The symbol contents in database are shown below.



| RecNo | id | format | symboldata |
|---|---|---|---|
| | Click here to define a filter | | |
| 1 | SYMBOLMARKER_Park@Changchun__16_16_true_-913710336 | PNG | |

# UTFGrid format attribute

# tiles

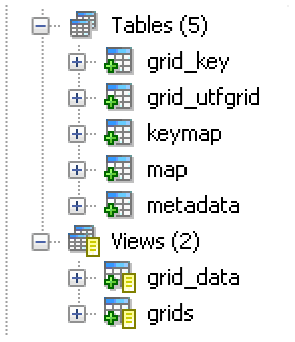
UTFGrid is a specification developed by MapBox for storing attribute map data to SQLite database and quickly using, managing and sharing the data. The specification is published by MapBox, please refer to http://www.mapbox.com/demo/visiblemap for details.

UTFGrid employs Unicode characters to encode geographic features into data tiles that save space. The tile is called Grid and stored in SQLite database.

SuperMap iServer extend UTFGrid based on original specification to satisfy the needs. After the extension, the .utfgrid file keeps the metadata table "metadata" and any coordinate system and resolution are supported.

Name of UTFGrid file is composed of map name, layer name, EPSG Code, tile width and height, pixCell, and the .utfgrid extension (for example, World_Countries@World_4326_256X256_8.utfgrid). Please refer to CoordSysType for EPSG Code, and pixCell of table 2 for pixCell.

The structure of the database table is as follows:



- metadata

The metadata table employs key-value pair to store settings of map properties data, including two fields of the text type "name" and "value", which are identical to utfgrid1.0.0. The structure of the table is illustrated in table 1:

Table 1 metadata table example

| name | value |
|---|---|
| name | World |
| layerName | Countries@World |
| version | 1.0.0 |

| | |
|---|---|
| description | World created on 2013-07-23 17:27:18 by SuperMap iServer |
| format | json |
| bounds | -180.000,-90.000,180.000,90.000 |
| ext_spec_version | 201310 |
| axis_origin | -180.0,90.0 |
| axis_positive_direction | RightDown |
| crs_wkid | 4326 |
| crs_wkt | GEOGCS["GCS_WGS_1984", DATUM["D_WGS_1984", SPHEROID["WGS_1984",6378137.0,298.2572235630001,AUTHORITY["EPSG","7030"]], AUTHORITY["EPSG","6326"]], PRIMEM["Greenwich",0.0,AUTHORITY["EPSG","8901"]], UNIT["DEGREE",0.017453292519943295], AUTHORITY["EPSG","4326"]] |
| tile_height | 256 |
| tile_width | 256 |
| resolutions | 0.34886137129 |
| scales | 1.4677821811243212E8 |
| pixCell | 8 |

Table 2 shows the fields' description of metadata table.

Table 2: Description of fields in metadata table

| Field name | Requ | iServ | Description |
|---|---|---|---|

| | ired or not | er extension or not | |
|---|---|---|---|
| name | Yes | No | Name of map. |
| layerName | Yes | No | Layer name. |
| version | Yes | No | UTFGrid version specification. Version of UTFGrid specification is 1.0.0. |
| description | Yes | No | Descriptive information. |
| format | Yes | No | UTFGrid format: json, html, jsonp, rjson. |
| bounds | No | No | Map extent used to acquire attribute data, with the unit being the map units. The format of bounds of iServer is as follows: -180, -85, 180, 85. |
| ext_spec_version | No | Yes | Version extended based on MBTiles specification by SuperMap iServer, 201310 here for instance. |
| axis_origin | No | Yes | The position of the tiling origin. |
| axis_positive_direction | No | Yes | Enumeration of positive directions of the axes of the coordinate system (RightDown, RightUp, LeftDown, LeftUp). |
| crs_wkid | No | Yes | EPSG Code of the coordinate system (-1000 indicates the custom coordinate system and 0 indicates common planar coordinate system). |
| crs_wkt | No | Yes | The geographic coordinate system represented by wkt. (wkt: A text markup language developed by OGC. For details, please refer to http://docs.geotools.org/stable/javadocs/org/opengis/referencing/doc-files/WKT.html) |
| tile_height | No | Yes | The grid height, generally 256. |
| tile_width | No | Yes | The grid width, generally 256. |

| | | | |
|---|---|---|---|
| resolutions | No | Yes | Resolution set, resolutions corresponding to different levels of tiles. |
| scales | No | Yes | Collection of scales corresponding to different levels resolutions. |
| pixCell | No | Yes | Pixel width of each cell in the grid. |

- grid_utfgrid

grid_utfgrid table is used to record grid_id and utfgrid in utfgrid. The structure of the table is as follows:

Table 3 grid_utfgrid table filed description

| Field name | Data type | Description |
|---|---|---|
| grid_id | TEXT | Grid ID |
| grid_utfgrid | BOLB | UTFGrid data |

Example of table contents is as follows:

- map

The map table is used to record row number, column number, resolution, id corresponding to the grid, scale, tile level, time for tile creation, etc. of each grid. The structure of the table is as follows:

Table 4 grid_utfgrid table filed description

| Field name | Data type | Description |
|---|---|---|
| zoom_level | integer | Tile level, with -1 indicating custom scale set. |
| resolution | varchar | Resolution corresponding to the tile. |
| tile_colum | integer | The column number of the tile. |

| | | |
|---|---|---|
| tile_row | integer | The row number of the tile. |
| grid_id | TEXT | Grid ID. |
| create_time | TEXT | The time when the tile is created. |

Example of table contents is as follows:

- keymap

The keymap table is used to record geographic feature information and the key_name value. Each geographic feature corresponds to a key_name value, that is, the SMID value of the geographic feature. The structure of the table is as follows:

Table 5 keymap Table Field Description

| Field name | Data type | Description |
|---|---|---|
| key_name | TEXT | UTFGrid ID. |
| key_json | BOLB | The json format data of the geographic feature information. |

Example of table contents is as follows:

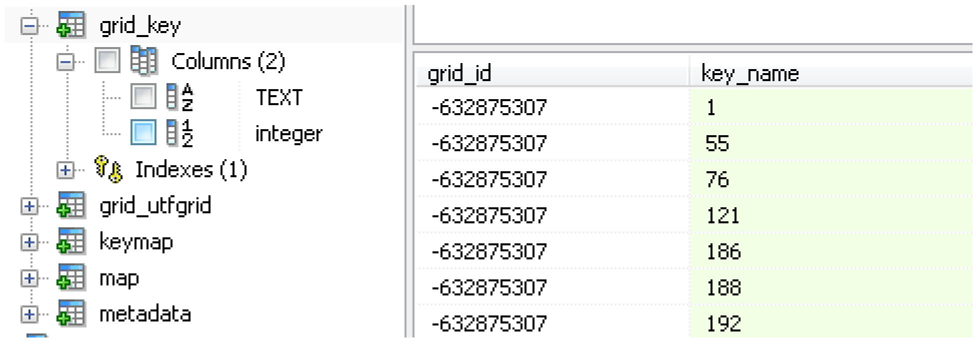
- grid_key

The grid_key table is used to record key_name values corresponding to each grid cell. Each grid_id corresponds to multiple key_name values, that is, a grid cell has multiple geographic features. The structure of the table is as follows:

Table 6 grid_key table filed description

| Field name | Data type | Description |
|---|---|---|
| grid_id | TEXT | Grid ID. |
| key_name | integer | key name. |

Example of table contents is as follows:



| grid_id | key_name |
|---|---|
| -632875307 | 1 |
| -632875307 | 55 |
| -632875307 | 76 |
| -632875307 | 121 |
| -632875307 | 186 |
| -632875307 | 188 |
| -632875307 | 192 |

# GeoPackage standard format

GeoPackage is the open data format that is set by OGC for storing geographical information. Storage format is platform-independent SQLite database file. GeoPackage can store vector element data as well as raster tile data, such as remote sensing pyramid and map tile matrix set.

Now the latest version of GeoPackage standard format is 1.0, for detail please refer to OGC official website (www.opengeospatial.org), or: http://www.geopackage.org.

For GeoPackage standard Chinese version, please refer to SuperMap open source project: http://github.com/SuperMap/geopackage_cn.

## File format

GeoPackage's storage form is the platform-independent SQLite database file, and the filename extension is “.gpkg”. SQLite has advantages such as self-contained, single file, cross-platform and server-unrelated. Therefore storage based on SQLite simplifies GeoPackage file's producing, distribution and using, meanwhile ensures the integrity of GeopPackage file data.

By default SuperMap will store GeoPackage tile in [SuperMap iServer installation directory]\webapps\iserver\output\sqlite\*.gpkg, and filename consists of mapname,EPSG Code, tile width and height, and .gpkg suffix, for example:ChinaProvinces_4326_256X256_PNG.gpkg.

## File content

GeoPackage stores data by a series of tables, including tables and views such as coordinate system, content description, feature data, tile data, metadata. Among them, the first two is required, and in GeoPackage there should have at least one feature data table or tile data table.

Now SuperMap only supports tile in GeoPackage, here is a brief introduction to the tile storage.

### Basic content

Coordinate system's coordinate reference system is referenced by content description table and geometrical features list, thereby connecting vectors and tile data of use list with real location on earth.

Content description table provides identifiable and descriptive information, which defines tile and feature table's name, data format and content description.

### Tile

GeoPackage's tile is organized, stored and indexed according to tile pyramid and exact tile zoom level.

## Tile pyramid

In GeoPackage, you can store multiple raster and tile pyramid datasets in data table or view. Tile pyramid means the pyramid structure consisted of tiles with different resolution which represents different space range in different zoom level, i.e, tile data. GeoPackage's tile pyramid dataset records data such as the zoom level, row and column number of every tile.

## Tile matrix set

Tile matrix set is the definition of the tile pyramid's hierarchy. If GeoPackage has tile pyramid data table, then a table or view of tile matrix set is required to define minimum border and spatial reference system.

## Tile matrix

Tile matrix is the rows and columns of tiles in a specific zoom level. Every tile pyramid data table can have multiple tile matrix. Tile matrix table or view records the structure of tile matrix in every zoom level, including tile matrix's row and column number, width and height of tile and so on. Here assign (0,0) to the upper-left tile coordinate of tile matrix that is in a random zoom level.

## Zoom level

In GeoPacakage, tile matrix layer's zoom level is an integer with the range from 0 to n, and it increase or decrease by 1 between adjacent zoom level. The 0 level scale can display entire current map's smallest scale in a tile, and another level's scale will change at a fixed rate or different rate based on this. As the zoom level increases, the actual space that every tile represents becomes smaller, and the spatial resolution will be higher. In GeoPackage, the tile with coordinate of (0,0) means upper left tile of tile matrix that is in a random zoom level.

iServer implements GeoPackage standard tile by using 0 to 20 level of scale. The 0 level scale can display entire current map's biggest scale in a tile (default pixel is 256*256), and the adjacent layers' scale zoom two times.

Take Web Mercator's (EPSG Code: 3857) coordinate system as example, the default scale level is:

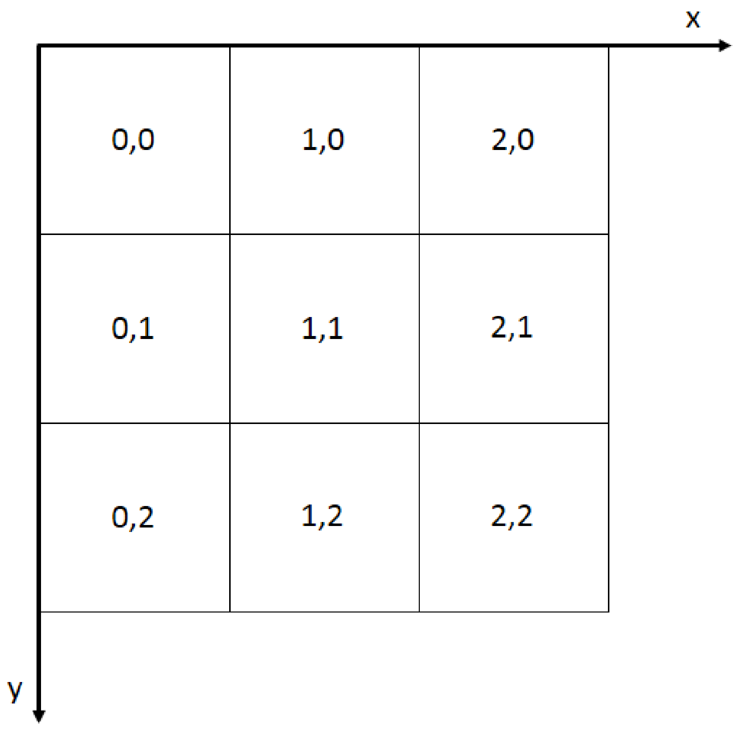
| Level | Map scale |
|---|---|

| | |
|---|---|
| 0 | 1/591658710.9091313 |
| 1 | 1/295829355.45456564 |
| 2 | 1/147914677.72728282 |
| 3 | 1/73957338.86364141 |
| 4 | 1/36978669.431820706 |
| 5 | 1/18489334.71591035 |
| 6 | 1/9244667.357955176 |
| 7 | 1/4622333.678977588 |
| 8 | 1/2311166.839488794 |
| 9 | 1/1155583.419744397 |
| 10 | 1/577791.7098721985 |
| 11 | 1/288895.85493609926 |
| 12 | 1/144447.92746804963 |
| 13 | 1/72223.96373402482 |
| 14 | 1/36111.98186701241 |
| 15 | 1/18055.990933506204 |
| 16 | 1/9027.995466753102 |
| 17 | 1/4513.997733376551 |
| 18 | 1/2256.9988666882755 |
| 19 | 1/1128.4994333441377 |

| | |
|---|---|
| 20 | 1/564.2497166720689 |

# ZXY Standard Tiles

SuperMap iServer and iExpress support to read map tiles of the ZXY standard, in order to use OpenStreetMap and other Internet tile map services. The tiling rules for ZXY tiles are as follows: divide the map in full extent from the top left corner, from top to bottom and from right to left, into tiles of 256 * 256 pixels each. The upper left corner is numbered as line 0, column 0, and the number increases as it goes to the down and right directions. As shown below:



For ZXY map tiles, currently only the set of fixed map scales in the Web Mercator coordinate system is supported (PCS_WGS_1984_WORLD_MERCATOR, EPSG Code: 3857). As shown in Table 1, the zoom level of 0 includes one map tile of 256 * 256 covering the entire globe, and the origin coordinates are defined as (-20037508.34,20037508.34), that is the global upper left corner.

Table 1: Parameter List for Each Zoom Level of zxyTileImage Resource

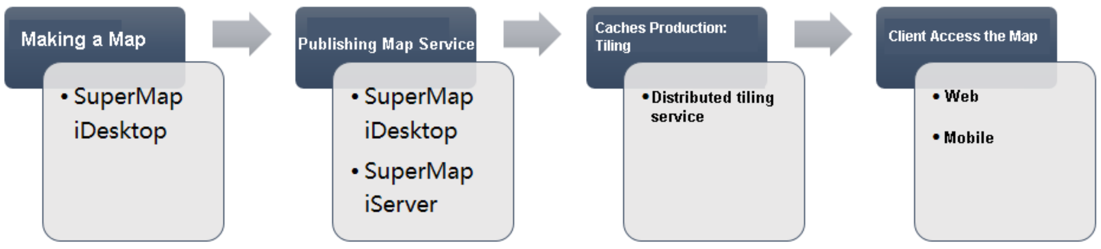
| Zoom level | Map width and height (pixels) | Number of tiles | Ground resolution (m/pixel) | Map scale (96 dpi) |
|---|---|---|---|---|
| 0 | 256 | 1 | 156543.033928 | 1:591658710.909131 |
| 1 | 512 | 4 | 78271.516964 | 1:295829355.454566 |
| 2 | 1024 | 16 | 39135.758482 | 1:147914677.727283 |
| 3 | 2048 | 64 | 19567.879241 | 1:73957338.863641 |
| 4 | 4096 | 256 | 9783.939621 | 1:36978669.431821 |
| 5 | 8192 | 1024 | 4891.969810 | 1:18489334.715910 |
| 6 | 16384 | 4096 | 2445.984905 | 1:9244667.357955 |
| 7 | 32768 | 16384 | 1222.992453 | 1:4622333.678978 |
| 8 | 65536 | 65536 | 611.496226 | 1:591658710.909131 |
| 9 | 131072 | 262144 | 305.748113 | 1:1155583.419744 |
| 10 | 262144 | 1048576 | 152.874057 | 1:577791.709872 |
| 11 | 524288 | 4194304 | 76.437028 | 1:288895.854936 |
| 12 | 1048576 | 16777216 | 38.218514 | 1:144447.927468 |
| 13 | 2097152 | 67108864 | 19.109257 | 1:72223.963734 |
| 14 | 4194304 | 268435456 | 9.554629 | 1:36111.981867 |
| 15 | 8388608 | 1073741824 | 4.777314 | 1:18055.990934 |
| 16 | 16777216 | 4294967296 | 2.388657 | 1:9027.995467 |
| 17 | 33554432 | 17179869184 | 1.194329 | 1:4513.997733 |

| | | | | |
|---|---|---|---|---|
| 18 | 67108864 | 68719476736 | 0.597164 | 1:2256.998867 |
| 19 | 134217728 | 274877906944 | 0.298582 | 1:1128.499433 |
| 20 | 268435456 | 1099511627776 | 0.149291 | 1:564.249717 |
| 21 | 536870912 | 4398046511104 | 0.074646 | 1:282.124858 |
| 22 | 1073741824 | 17592186044416 | 0.037323 | 1:141.062429 |
| 23 | 2147483648 | 70368744177664 | 0.0186615 | 1:70.5312145 |

Home > Configuration and management > iServer cache mechanism > Map cache > Producing and using map tiles

# Producing and using map tiles

SuperMap provides multiple tools to create map tiles, which is used to improve the access efficiency of online map services. This article will introduce the map tile production and use flow from cartography, tiles to use map services.



- Map producing and publishing
    - Produce maps
    - Publish map services
- Producing map tiles: tiling

- Using map services
    - Online access
    - Offline access

## Map producing and publishing

SuperMap iDesktop is the desktop GIS tool of SuperMap. It is the professional GIS data analysis, processing and cartography platform. This article recommends to use SuperMap iDesktop to create data and map. This tool supports to quickly publish map data as SuperMap iServer map services.

### Produce maps

As a traditional cartography tool, SuperMap iDesktop has the 2D and 3D cartography capacity. And it provides the rich and custom 2D and 3D symbols.

Please refer to SuperMap iDesktop Help for more information.

### Publish map services

SuperMap iDesktop supports to quickly publish the local and remote data, map in workspace as SuperMap iServer services. The service types can be REST service, OGC standard service and so on.

Please refer to SuperMap iDesktop Help for more information about the details.

You can also Publish services through SuperMap iServer.

## Producing map tiles: tiling

The distributed tile tasks provided by SuperMap iServer can add the multiple tile nodes in different computers, implement that multiple computers tile together. The distributed tile service supports to tile for all published map services. The service data source can be SuperMap workspace, remote WMS service, remote WMTS service, remote REST Map service, SuperMap cloud service, Bing Maps service, Tianditu service, MBTiles files, SMtiles files and so on.

You can split the map tiles according to the specified logic, and store them in the FastDFS and MongoDB distributed file system. You can store map tiles to .smtils based on MBTiles standard. You can also save the map tiles in local disk according to the SuperMap V5.0 cache strategy. Using distributed tile service can perform tile task for the one or more vector layers and attributes information in map service, which gets SVTiles vector tiles and UTFGrid attributes. Please refer to Map tile types .

If you use distributed tile task, the map tiles, vector tiles and attribute tiles all can be used by map services, and there is no need to configure again. If you modified the default storage path or other custom settings, you can Configure map service to use the cached tiles.

So we recommend to use distributed tile service to create the map tiles. Please refer to Distributed Tiling service.

In addition, you can use other tools to create SuperMap UGC format tiles for the workspace data, such as SuperMap iDesktop and SuperMap iObjects.

## Using map services

The published maps in client and the pre-tiled map tiles is to improve the efficiency of displaying map. So it plays a role in client-side. You can access these tiles online, and also put these tiles in mobile side.

### Online access

The map services of SuperMap iServer can be used by multiple terminals, such as SuperMap iDesktop, iObjects, iClient, and iMobile etc. But the common used scene is the online map service on Web side and mobile side.

For a map service and existing tiles, the output images request methods of client online access:

- Browse the map through .flash, .flash3d, .javascript, .vectortile and .silverlight of map resource. At this time, it needs to make sure that the map fixed scale and the existing scale stay the same. The image format of ouput request is the same with the existing tiles.
- Send the request to the server through the tileImage resource. It can request the specified tiles according to the scale, tile row, tile column, tile format and so on. It needs to confirm that the request parameters and tile size stay consistency;
- Access map service through Web client or mobile. You can develop a map browse script, sending request to the map service.

When accessing the SuperMap iServer map services through Web or mobile, If the server uses the existing map tiles, it can call the tiles directly, which improves the online access efficiency. Now it supports the Web tool: iClient for Flash, iClient for JavaScript, iClient for Silverlight, iClient for 3D. Mobile tool: iClient for Android, iClient for iOS, iClient for Windows 8, iMobile for iOS and iMobile for Android.

Except for the map tiles, SuperMap iServer also provides the vector tiles and attribute tiles. Take the grid tiles as the base map; overlay the vector POI data render;

So this ensures the maps output efficiency, and ensures the timeliness demanding POI, route data update quickly, also supports users frequently mouse interaction.

### Offline access

You can access the map service of SuperMap iServer through iClient or iMobile on terminals. But limited by the network, the downloading tiles on mobile side is still can not meet the requirement. So the server provides the application of map tile offline. That is, create the map tiles as a offline map package (*.smtiles or *.mbtiles). Download it or copy it to the mobile terminal.

# Configuring to use the cached tiles

Tiles generated through Distributed Map Tiling service, including Map Tile, Vector Tile, and Attribute Tile can be used automatically by the map, with no need for additional configuration. For a map service and existing tile, mapping request methods from the client includes:

- Browse the map through flash, flash3d, javascript, vectortile, silverlight representation of the map resource. You need to ensure that the fixed scales of the map be identical to scales of existing tiles and parameters for requesting map like picture format be identical to those of the existing tile.
- Send request to server through the tileImage resource, which allows you to request specified tile according to parameters such as scale, tile row, tile column, tile formats, etc. Request parameters need to be identical to those of the requested tile.
- Develop a map browsing script through the iClient and send request to map service. Request parameters need to be identical to those of the requested tile.

Of course, if you modified the default storage path or other custom settings, you can use these tiles through manual configuration. Among which, UGCV5 format tiles can be used through configuring Map service provider. FastDFS、MongoDB、SMTiles、MBTiles、UTFGrid、SVTiles tiles can be used through configuraitng Map service component.

- Configure to use SuperMap UCG Map Tile

- Configure map component for using other tiles
    - Universal configuration
    - Distributed tile library configuration

## Configure to use SuperMap UCG Map Tile

SuperMap UGC Map Tile can be generated through Distributed Tiling service or SuperMap iDesktop, SuperMap iObjects. Versions include5.0, 4.1, etc. Please refer to SuperMap UGC Map Tile versions for details.

For generated SuperMap UGC tiles, you first need to make sure that the corresponding map has been published as map service, then you can use the Map Tile through configuring the map service provider, SuperMap UGC5.0 tiles for instance. During operation, you can configure service provider through WebManager, or directly Modify XML service configuration. Take local map service provider as an example, open WebManager and enter the configuration page for the map providers, (http://<server>:<host>/iserver/manager/providers/<Map Service Provider Name>), set cache version in advanced settings (that is, version of SuperMap UGC Map Tile), click Save to take effect.

SuperMap UGC Map Tile used by SuperMap iServer are stored in picture output path, that is, [SuperMap iServer installation directory]\webapps\iserver\output\cache\, by default. Please refer to SuperMap UGC Map Tile versions for specific paths of different SuperMap UGC map tile versions.

While using SuperMap UGC tiles, if the storage path of the Map Tile are not paths described above, you need to copy the Map Tile to the corresponding paths. Also, you can modify the cache output path of the corresponding map service provider (<outputPath>../webapps/iserver/output</outputPath>) to make it identical to storage path, therefore ensuring that the map service can properly use the SuperMap UGC tiles.

## Configure map component for using other tiles

We can use FastDFS, MongoDB, SMTiles Map Tile, SVTiles Vector Tile, UTFGrid Attribute Tile through configuring map service components.

Modifying the configuration of map service components can be realized through WebManager or Modifying service configuration file.

### Universal configuration

Open WebManager and enter the page for map service component configuration (http://<server>:<host>/iserver/manager/components/<Map Service Component Name>), check "Enable Cache" and set following parameters:

- Enable Cache: Whether to enable caching or not.
- Cache Read-only or not: Determines whether the cache files are editable or not. If checked, generated cache files will be read-only. If not checked, existing tile files will be able to be dynamically updated.
- Cache Survival Time: The survival time of the cache is calculated from the creation of the cache.The unit is minutes. 0 indicates that the cache will survive forever. This configuration is only valid for SMTiles cache, attribute tile cache, and vector tile cache.
- Use Map Tile: Support SMTiles, FastDFS, MongoDB storage formats. If FastDFS or MongoDB format is selected, existing storage location needs to be imported or a new storage location needs to be added. If the newly added storage location is used by following tile service, the generated tiles will be automatically used by the current map service.
- User Attribute Tile: Currently support UTFGrid Attribute Tile, and the default storage location is [SuperMap iServer installation directory]\webapps\iserver\output\sqlite\.
- User Vector Tile: Currently support SVTTiles tiles, and the default storage location is [SuperMap iServer installation directory]\webapps\iserver\output\sqlite\.

During configuration for SMTiles format, existing SMTiles Map Tile (*.smtiles) will be employed for mapping of map services by default. However, if the current map service and mapping request are compliant to MBTiles rules and there are MBTiles Map Tile (*.mbtiles) in the cache directory, the map service will employ existing MBTiles.

## Distributed tile library configuration

For tiles in distributed tile libraries, you can click Configure to Map Service button corresponding to distributed tile library on the distributed tile library, select service component corresponding to map in the tile library to realize fast configuration.

While selecting service component:

- If there are no tiles for the current map component in the tile library, tiles will be able to be stored to tile library dynamically by default after configuring to map service component.
- If there are already tiles for the current map component in the tile library, the tile library will be readonly by default.

After configuring distributed tile library to map service, the corresponding map service will employ the tileset of the newest version in the distributed tile library for mapping by default.

If you need to configure tilset of old version, please refer to Configure to use the historical-version tiles .

# Publishing map tiles directly

SuperMap iServer supports directly publishing map tiles into map services.

There are two methods for tile publishing: directly publishing map tiles into map services; publishing tiles into map services through customized method.

## Quickly publishing tiles

You can click Quickly publish services in WebManager to directly publish tiles into map services. Pleaser refer to following links for specific steps:

- Publish the tiles stored in FastDFS
- Publish the tiles stored in MongoDB
- Publish SMTiles tile package
- Publish SVTiles tile package
- Publish UGCV5 tile
- Publish MBTiles tile package
- Publish GeoPackage tile
- Publish TPK tile package

## Publishing tiles through customized method

Publishing map services can be realized through creating map service provider and map service component.

You can refer to Configure service provider through iServer WebManager to create a map service provider, or Configure map service provider through modifying Service configuration file. Examples for latter method, you can refer to the following links:

- Configure FastDFS map service provider
- Configure MongoDB map service provider
- Configure MBTiles map service provider
- Configure SMTiles map service provider
- Configure SVTiles map service provider
- Configure GDP map service provider
- Configure GeoPackage map service provider
- Configure TPK map service provider

# How to update tiles

With Distributed Tiling service, SuperMap iServer can generate tiles either with multiple scales or within a specified boundary. Those tiles can be used in server-side or directly in client side as it is offline status. In general, you should think about the updating issue of the tiles when the maps on the server changed, e.g., the style of a layer changed or some elements in a layer are added, deleted or modified. In this case, to update the tiles could keep the data in time.

You don't need to worry about the tiles whether they are the latest or out of date when there are some changes on the maps. SuperMap iServer will automatically check each tiles. If a tile checked is out of date, a new tile will replace the old one dynamically. Besides the automatic update, you can also set the tiles as read-only, and manually update tiles by Distributed Map Tiling service.

Due to the massive quantity of data, the efficiency of the Distributed Map Tiling service is better than the automatic tile updating. So, SuperMap recommends you update tiles manually or use the tiling tool to update tiles.

- Update and append tiles manually
- Tile updating tool
    - Methods
    - Directional update
    - Note
- Remove tiles with specified bounds

## Update and append tiles manually

The Distributed Map Tiling service of SuperMap iServer supports updating and appending new tiles into the existing tiles. For instance, to update or append new tiles with certain scales or certain bounds into the existing tiles. SuperMap supports the following formats of tiles to be updated or appended: FastDFS, MongoDB, SMTiles, MBTiles, UGCV5, SVTiles (vector based), UTFGrid (attribute based).

The steps of the "update and append tiles manually" are identical with How to use Distributed Map Tiling service. You need to select a path to place the tiles (such as output path\sqlite\) or to the tile library (determined by storage ID). Meanwhile, you should specify the scale and the bound that you want to append. If a new tile is identical with the existing tile, the old one will be replaced.

As for the distributed tiles, there will be prompted with "The tiles exist. Do you want to create a new tiling version？“ before starting to tile. In this case, you should select No, and select an exist version for updating or appending. Then, click OK.

## Tile updating tool

SuperMap iServer provides a tile updating tool for importing the tiles of FastDFS, MongoDB, SMTiles, and UGCV5.

Through this tool, you could:

- Update tiles by other tiles
- Combine tiles which are stored in a distributed way
- According to the specified scale and the bound, export the distributing tiles into the file format (such as SMTiles) for conveniently distributing.

### Methods

Log in iServer WebManager, click **Services**>**Advanced**>**Tiles Update**. The detailed steps are:

1. Click **Create tile update task**;
2. Set **Input tile set**: the source of the tiles
3. Set **Target tile set**: the tiles to be updated
4. iServer automatically shows scales in the list of **Input tile set scale**. Select the scales to updte and click **Add**.
5. Set **Update bounds**. By default, it is the maximum bound of the tiles
6. Click **Start update**.

### Directional update

You could update the tiles which are being used by the map service component or the map service provider. Click **Update tile** on the configure page, so that it goes to the tile updating tool:

- For the map service component which starts the tiles, you can choose a tile type on the configure page, and click the **Update tile** link under the **Storage Location**.
- For the map service provider of the tile package (e.g., SMTiles), you can click **Update tile** link under the **File Path** on the configuration page.

### Note

To make sure the data correctness, please consider the following aspects when updating tiles:

- Make sure that, in the tile source and the target data, tiles with the same map name should have the same tile styles, such as tile size, transparency, etc.
- Besides the requirements above, if the tiles are non-distributed storage tiles such as SMTiles, UGCV5, the map of **Input tile set** should be identical with the map of **Target tile set**.

## Remove tiles with specified bounds

If a certain scope of the data needs to update, you can use clearCache. That is to say, delete the tiles in this bound first, and then create new tiles.

For example, if you want to delete tiles with the bounds Lon 120°~150° & Lat 30°~50°, please input the URL:

http://supermapiserver:8090/iserver/services/map-world/rest/maps/WorldMap/clearcache.rjson?bounds={"rightTop":{"y":50,"x":150},"leftBottom":{"y":30,"x":120}}。

It will return true when it is removed successfully.

Note: the clearCache will clear all tiles in this bound and tiles partially in this bound if you specify a geographic scope.

# Suggestions on using the cached tiles

In a scale, the more tiles of the map we get, the higher hit rate they will be. However, the greater hit rate could also result in the huge amount of tiles to be transferred, which will, in return, affect the map response time. Therefore, we need to find a balance between the server concurrency and the network traffic. In other words, the client side should display maps in an optimal way.

For preparing tiles, you can take account of the following factors.

- Tile types
- Tile size
- Scale
- Geographic extent

## Tile types

SuperMap iServer supports map tile, vector tile, and attribute tile. Tiles can be used for online sharing and offline viewing. You can choose proper types of tiles according to your practical situation.

- Map Tile

Map tile refers to tiling maps to raster pictures. It supports FastDFS (distributed storage), MongdoDB (distributed storage), SMTiles, MBTiles, and SuperMap UGC.

SuperMap UGC type is the common and traditional type which can be used in all SuperMap products if the SuperMap UGC tile versions are the same. The UGCV5 tile is just refers to V5.0 original cache.

In addition, the picture format of the map tiles include PNG, JPG, GIF. If you choose PNG and the color value is less than 256, pictures will auto stored as PNG8 in SuperMap iServer for saving more storage space.

- Vector Tile

The vector layer can be splitted and stored as the vector tiles. iServer supports SVTiles.

In the real map service field, users not only view maps, but also query, select or highlight elements on maps. So, element service becomes a necessity. Similar to the improvement of map tile access, the rendering of elements on the client side can be improved by pre-caching vector data. Here comes the vector tile technology. The storage space of vector data is smaller than map tiles, so it is more suitable for expressing geospatial elements which requires timeliness, such as POI, routes, etc. The popular online map services, such as Google Maps, Baidu Map, use map tiles as the base map and vector tiles as element service.

- Attribute Tile

The attribute data in the vector layers can be stored as the attribute tiles. iServer supports UTFGrid.

In the real map service field, if there are massive mouse interactive operations, in the traditional way, we will overlay the element layers on the maps, and every element has its own hot spot and event. However, in the environment of massive amount of data and high concurrent requests, the clients, especially the mobile terminal, cannot smoothly render so many geographic elements. In this case, we choose another way, that is, every tile also records the attribute information. In other words, we input attribution information, divided by grids, of elements into the original tiles. This pre-stored element attribute information for each tile is so-called attribute tile. The integration of map tiles and attribute tiles has been widely used in many sectors. The most typical one is MBTiles specification and its subsidiary UTFGrid specification.

## Tile size

There are two sizes of tiles in the precaching services of SuperMap iServer: 512*512 and 256*256.

When visiting the map service of SuperMap iServer with the tiles layer of the iClient, by default you use the 256 * 256 image. For example, iClient for Ajax can use SuperMap.Web.Mapping.TiledDynamicRESTLayer (default is 256 * 256); iClient for

Silverlight can use SuperMap.Web.iServerJava6R.TiledDynamicRESTLayer (default is 256 * 256). Of course , you can also change the TileSize into 512 * 512.

You need to set the size of the cache image during caching so as to take advantage of precaching.

## Scale

To take full advantage of the precaching service of SuperMap iServer and realize multi-scale zoom, you can precache a map into multiple scales of caches. Setting the scale array on SDK of SuperMap iServer such as for JavaScript, for Flash and for Silverlight to realize the multi-scale zoom.

Take iClient for JavaScript for example, sets the scale array of the map into the same scale of precaches. All the layers of the map during scaling will only display the scale of the precaching images.

## Geographic extent

Users can precache the frequently accessed map area, and then generate dynamic caches for the less accessed map area according to their needs.

For example, when accessing a world map, we usually don't pay attention on the sparsely populated areas which don't need generating caches, and hence we can specify the map into areas as shown in the figure below: