1. void clear(); //清空数据，释放内存

清空上一次的计算结果。每次在执行计算之前，都要先调用一次该方法，以确保本次计算结果不受影响。

1. int AddNode(*wstring*& name, double p=0, double r=0, double l=1.0, int type=1);//添加一个节点，并返回该对象在算法组件内部的编号

接口参数说明：

name:节点的名字（需保证唯一）

p:容量

//WDM设备 容量=波长粒度\*1000\*波长数（以M为单位，如果已经为M，则不需要\*1000）

//SDH设备 容量=带宽

r: 交叉粒度

//WDM设备 交叉粒度=波长粒度\*1000（以M为单位，如果已经为M，则不需要\*1000）

//SDH设备 交叉粒度=交叉粒度

l: 波长转换(是否支持波长转换,是：1，否：0)

type:节点类型（无需处理）

接口返回值

节点在算法组件内部的编号，需记录下来，方便后期处理

1. int AddLine(long b, long e, *wstring*& name, int w, double c, double l =0,double p=0, double r=1.0,bool lineType =false); //添加一条物理链路，并返回该对象在算法组件内部的编号

接口参数说明：

b:开始节点编号（由AddNode方法返回）

e:结束节点编号（由AddNode方法返回）

name: 链路的名字（需保证唯一）

//w: 容量=带宽 ( SDH和WDM设备连接时，容量参数以SDH参数为准，取最小容量节点)

//c: 长度=长度(以KM为单位)

//l: 业务数=业务数（按默认值处理）

//p: 可靠度=可用度（按默认值处理）

//r: 交叉粒度 （SDH取节点的交叉粒度，WDM取波长粒度）

w:波长总数，计算公式：（node1带宽+node2带宽）/（node1交叉粒度+node2交叉粒度）?

c: 容量=带宽 ( SDH和WDM设备连接时，容量参数以SDH参数为准，取最小容量节点)

l: 长度=长度(以KM为单位)

p: 业务数=业务数（按默认值处理）

r: 可靠度=可用度（按默认值处理）

lineType: 无需处理，保持默认值即可

接口返回值

该条物理链路在算法组件内部的编号，需记录下来，方便后期处理

1. int AddLogic(long b, long e, std::wstring& name, double p = 0); //添加逻辑链路

接口参数说明：

b:开始节点 id

e:结束节点id

name:逻辑链路name

p: 带宽（100，155等等，如果一条逻辑链路下有多条业务，则此参数等于多条业务的业务颗粒度之和）

1. int AddService(long s, long t, *wstring*& name, *wstring*& name2, double p=1.0, double r=0, int d=0); //添加一条业务（又称通道，位于资源树逻辑链路下以OCH开头的节点），并返回该条业务在算法组件内部的编号

接口参数说明：

s: 开始节点编号（由AddNode方法返回）

t: 结束节点编号（由AddNode方法返回）

name:业务的名字（位于资源树逻辑链路下以OCH开头的节点）

name2:该业务父级的名字（位于资源树逻辑链路下以LLK开头的节点）

p: 业务颗粒，对应业务颗粒度

r: 保护方式=保护方式（1+0保护:0,1+1保护:1）

d: 保护标记（主业务是0，备用业务是1，目前只存在主业务，暂时不考虑备用业务的情况）

接口返回值

业务在算法组件内部的编号，需记录下来，方便后期处理

1. int AddService(int l, std::wstring& name, std::wstring& name2, double p = 1.0, double r = 0, int d = 0); //添加业务

//p：颗粒, r：保护方式， d：保护标记(0=工作，1=保护)

接口参数说明：

l: AddLogic方法返回的id

name:主业务名称

name2:备用业务名称

p:业务颗粒度

r: 保护方式=（1+0保护:0,1+1保护:1）

d: 保护标记（主业务是0，备用业务是1，目前只存在主业务，暂时不考虑备用业务的情况）

注意：此方法在执行RWA算法前调用，详细用法见demo

1. bool InitGraph1() //无向图初始化。初始化算法组件内部的拓扑图，只有返回值为true时，才能执行下面的算法函数，否则终止[[1]](#footnote-1)
2. *vector*<*vector*<int>> GetServicePath(*wstring*& name, int prtd=0,int diff=0, int weight=0); //返回给定节点对的主备用节点路径(计算业务路由，按单条业务计算)1

//int prtd=0,int diff=0, int weight=0

接口参数说明：

name:业务名称

prtd:保护方式

diff:分离方式

weight:路由策略

接口返回值

二维数组，以下是返回结果示例，实际使用时只需关注第0行

[0] 1 3 5 7 0 此行为物理链路编号（AddNode返回的编号）

[1] 2 9 6 1 8 此行为波长时隙/号（暂时不处理，后期显示波长/时隙图时使用）

如何使用：

1. 将上面的二维数组中的0维所对应的线路编号，在拓扑图上显示出来。
2. 在【导出计算结果】界面，将计算结果导出；示例excel详见《实例计算结果.xls》。

Excel文档说明：

* 1. 本次导出只涉及前四个sheet,即统计信息、节点信息、链路信息和通道信息
  2. 统计信息sheet只导前三行，即节点总数、链路总数和通道总数，其余的列显示但保持默认值0
  3. 节点信息sheet，波长数=带宽/交叉粒度（m）
  4. 链路信息sheet中的带宽利用率字段，公式为：已用带宽/带宽，精确到小数点后三位
  5. 通道信息sheet中的路由分配字段，需要把计算结果返回的线路编号转换为节点编号并显示出来

注意：此方法会调用多次，有多少条业务，就会调用多少次

1. void ClearCal(void);//只清除临时计算结果；注意此方法和clear()的区别，clear不光要清除结果，同时还会将nodes,lines,sevices数据一并清除
2. Generic::Dictionary<String^, Generic::List<Generic::List<int>^>^>^ RWA(int% r, int weigh, int diff, int prtd, int m); //返回 RWA 计算节点路径和波长数组，功能与GetServicePath相同1

// weight=3, int diff=2, int prtd=1, int m=5

接口参数说明：

r: 输入=系统波长数，输出=实际用到的波长数，r为跟踪引用变量

weight: （0=跳数，1=距离， 2=可靠度[可靠+拥塞],3=容量）目前该参数不起作用，默认=3

diff: 分离方式

prtd: 保护方式

m: 单条业务路径最大数

注意：此方法调用一次即可；在方法内部会遍历所有的业务

1. bool InitGraph2();//有向图初始化1
2. bool InitGraph3();//双向图初始化1

1. 注意：此接口只有在授权机上才可以正常运行 [↑](#footnote-ref-1)