GBASE®

GBase 8s GCI 接口使用指南

www.gbase.cn

GBase 8s GCI接口使用指南，南大通用数据技术股份有限公司

版权声明

免责声明

通讯方式

南大通用数据技术股份有限公司

天津市高新区开华道22号普天创新产业园东塔20-23层

电话： 400-013-9696 邮箱： info@gbase.cn

商标声明

目录

1 GCI 综述

本手册为GBase 8s产品兼容Oracle的数据访问接口提供的GBase Call Interface（简称GCI）的使用说明手册。

1.1 头文件

GCI接口库头文件有四个：

- Gci.h：gci 接口定义文件
- Dci.h：dci 接口兼容文件
- Oci.h：oci 接口兼容文件
- Gcierror.h：错误码定义文件

1.2 编译

GCI接口库的名称：libclntsh.so 或 Libclntsh_gbase.so

使用GCI接口库进行应用程序编译时，需要设定LD_LIBARY_PATH路径包含有：

- $GCICLIENTDIR
- $GBASEDBTDIR/lib/esql
- $GBASEDBTDIR/lib

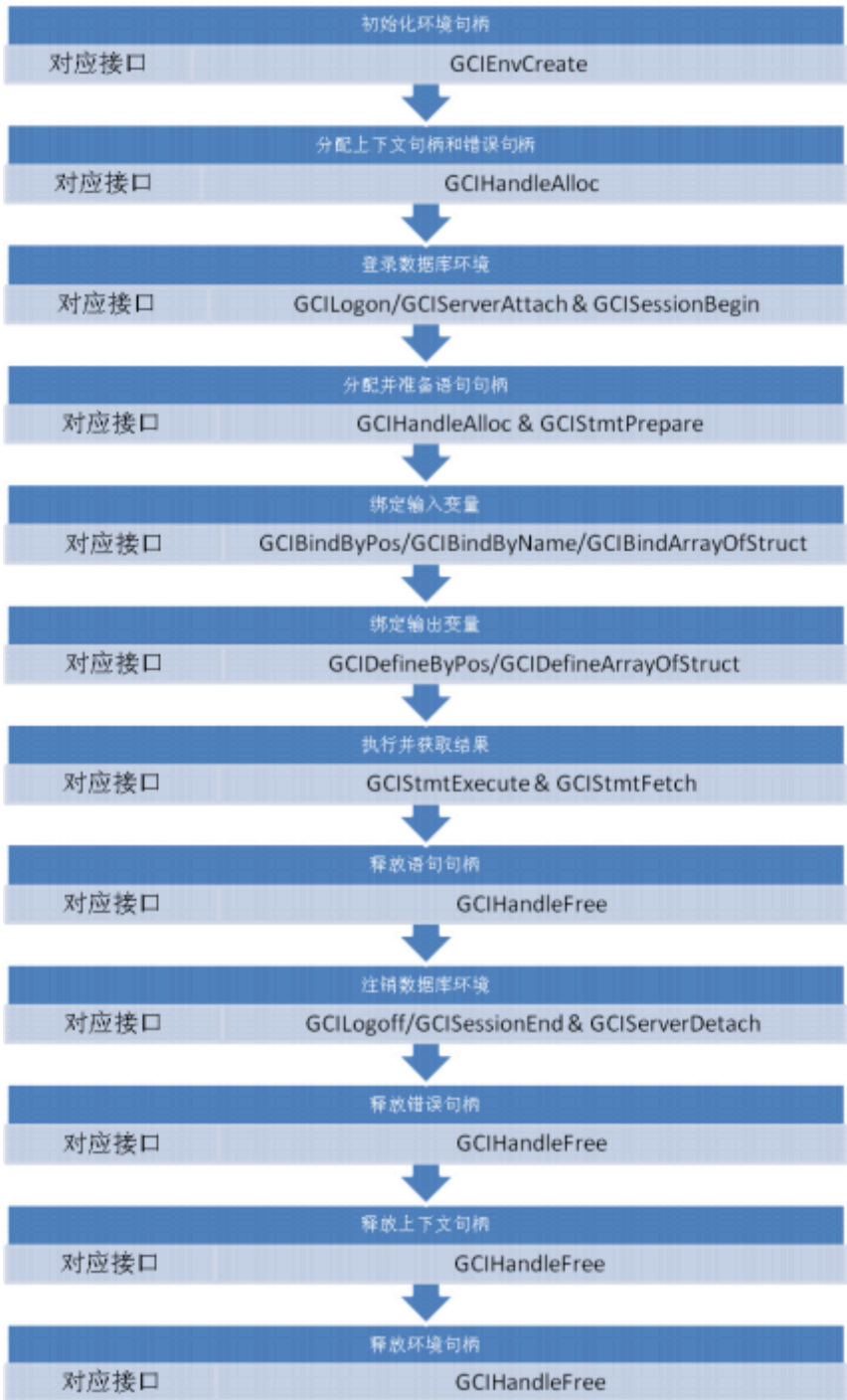
其中GBASEDBTDIR是${GCICLIENTDIR}/relyon。

1.3 兼容平台

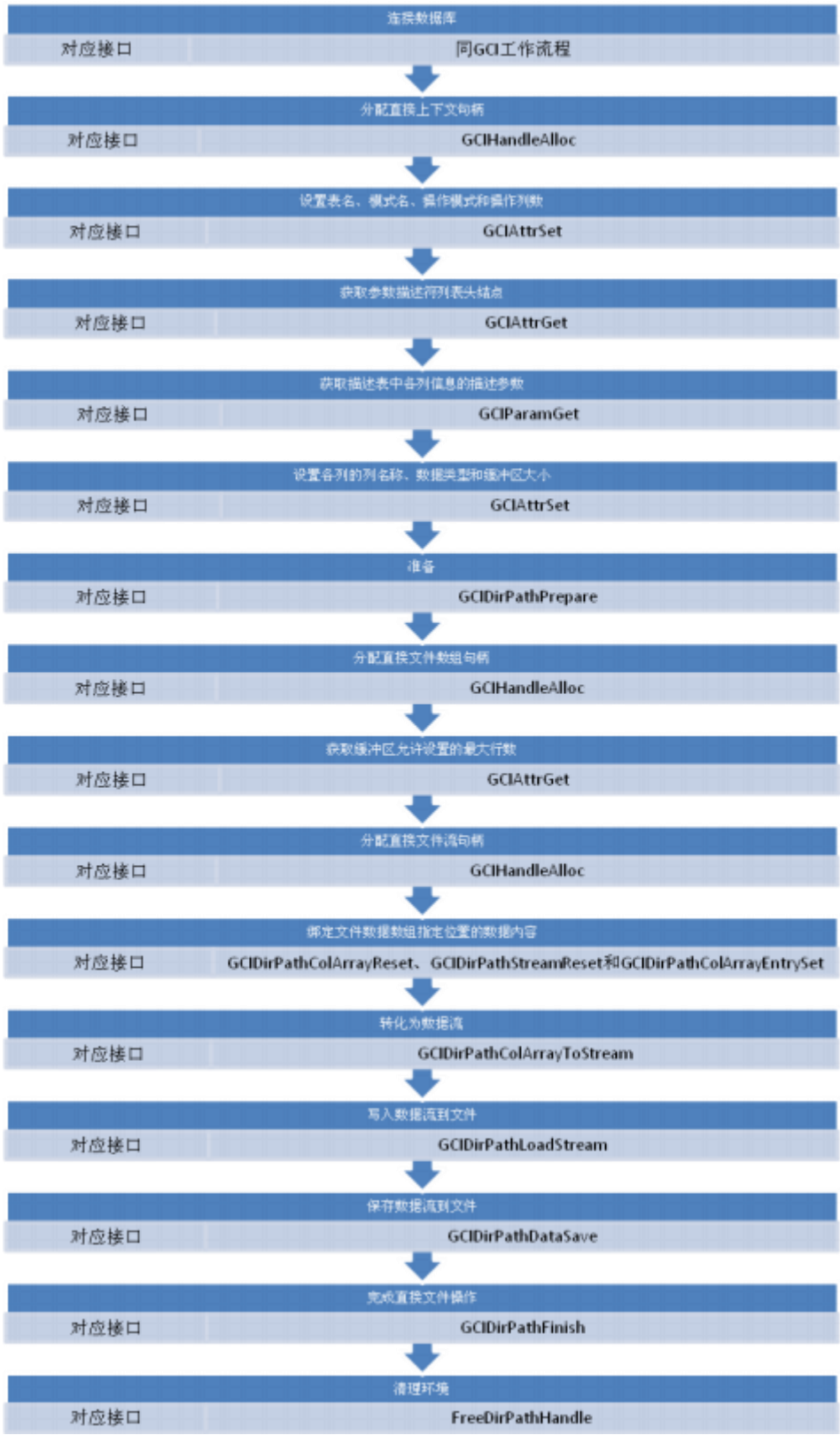
操作系统： 64位凝思磐石V4.2，64位中标麒麟V5，麒麟V10，RHEL6

CPU：x86_64

1.4 GCI SQL语句句柄工作流程



1.5 Direct Path Loading工作流程



1.6 GCI多线程开发

GCI接口是线程安全的， 所以能够很好的支持多线程并发。在多线程开发过程中， 需要分别为每个线程分配单独的环境句柄、上下文句柄和语句句柄，以此保证线程的安全性和程序的正确性。

注： GCI接口打开log日志时，不是线程安全的。

2 2 配置文件与参数设置

2.1 概述

在使用GCI接口之前， 需要对系统的配置文件和参数进行设置，涉及到的配置文件有一个，为： sqlhosts.std。

2.2 sqlhosts.std

位置： GBASEDBTSQLHOSTS环境变量指定

参数： 对于每一个数据库实例， 添加一行配置信息，格式为：

```
Server-name protocol IP-address Port-name
```

其中：

Server-name：数据服务的实例名

protocol：连接使用的协议，取值为onsoctcp

IP-address：服务器IP地址

Port-name ：/etc/services中Server-name对应的端口号

2.3 其他环境变量

位置： profile文件

参数：

```
export GBASEDBTSERVER=数据库实例名

export GBASEDBTDIR= ${GCICLIENTDIR}/relyon
export GBASEDBTDBNAME=连接的数据库名(如果程序连接时已经指定数据库名，可以不用设置)
export GCICLIENTDIR=gci接口路径

export
LD_LIBRARY_PATH=$(GBASEDBTDIR)/lib:$(GBASEDBTDIR)/lib/esql:$(GCICLIENTDIR)/lib:$LD_LIBRARY_PATH
```

3 类型

3.1 概述

表1.1中列举了GCI支持的数据类型：

数据类型	类型说明
SQLT_CHR	变长字符串类型
SQLT_STR	变长字符串类型
SQLT_AFC	定长字符串
SQLT_VCS	变长字符串
SQLT_AVC	变长字符串
SQLT_INT	整数类型
SQLT_FLT	浮点数类型
SQLT_BIN	二进制类型
SQLT_CLOB	大对象数据，文本数据
SQLT_BLOB	大对象数据，二进制数据
SQLT_ODT	时间类型（datetime ）
SQLT_DAT	时间类型（datetime ）
SQLT_BFLOAT	单精度浮点
SQLT_BDOUBLE	双精度浮点
SQLT_IBFLOAT	二进制单精度浮点（与SQLT_BFLOAT同）
SQLT_IBDOUBLE	二进制双精度浮点（与SQLT_BDOUBLE同）
SQLT_DATE	日期数据
SQLT_TIME	时间数据类型
SQLT_TIMESTAMP	时间戳类型
SQLT_INTERVAL_YM	Interval年月数据类型
SQLT_INTERVAL_DS	Interval天秒数据类型
SQLT_NUM	数字型类型

3.2 字符串类型

GBase 8s主要支持以下字符串类型及对应SQLT类型：

C语言类型(64位系统)	缺省字节数	数据库类型名称	SQLT类型
Char[n]	1	CHAR(n) /NCHARn<=32767	SQLT_CHR/ SQLT_STR/ SQLT_AVC/ SQLT_AFC/ SQLT_VCS
CHAR[N]	无	Varcar(n)/NVARCHAR,n <=255	
CHAR[N]	2048	LVARCHAR(N) n<32739	

在GCI接口中，访问GBase 8s系统中的字符串类型数据，类型指定为SQLT_CHR，SQLT_STR，
SQLT_AVC，SQLT_VCS时效果是相同的，并没有区别。

但在不同厂家配置中，GCI接口返回字符串类型会有所不同：

factory	SQLT	数据库类型
1	SQLT_STR	CHAR
1	SQLT_STR	VARCHAR
1	SQLT_STR	LVARCHAR
2	SQLT_AFC	CHAR
2	SQLT_CHR	VARCHAR
2	SQLT_CHR	LVARCHAR
3	SQLT_AFC	CHAR
3	SQLT_CHR	VARCHAR
3	SQLT_CHR	LVARCHAR

其中指定为SQLT_AFC类型访问数据库时，在数据绑定时，其缓存可以不包含字符串结束 符的位置。比如向char(10)字段插入一个10字节的字符串，其绑定的缓存大小可以申请为只 有10字节大小。此时由于并不包含字符串结束符， 使用strlen获取长度是大于或等于10的。

其他的字符串类型， 其缓存大小需加上结束符位置，而绑定时长度则是strlen()的长度。

两者差异在进行char(1)类型字段访问是最为明显。如果使用SQLT_AFC类型获取该字段数 据，缓存大小可以设置为1。但是其他类型获取时， 会报截断错误。插入过程也是同样， SQLT_AFC类型时， 可以在缓存为1字节情况下正常插入， 但是其他类型一般(有可能会有 随机性)会报截断错误。

例：

```
Char ch =  ‘i’;

rc=GCIBindByPos(stmt,&Bindp[0],errhp,1,(dvoid*)&ch,(sb4)1,SQLT_AFC,NULL,NULL,NULL,0,NULL,GCI_DEFAULT);
```

上例中，类型指定SQLT_AFC可正常插入，但如果是其他类型SQLT_CHR则可能报截断错误，因此此时(&ch)字符串没有结束符。

3.3 数值类型

3.3.1 整型

GBase 8s支持的整型与GCI类型对应：

C语言类型(64位系统)	字节数	数据库类型名称	SQLT类型
short	2	smallint	SQLT_INT
int	4	int	SQLT_INT
long	8	Bigint/int8	SQLT_INT
char	1	无	SQLT_INT

SQLT_INT在GCI接口中是所有整型类型的统称，执行绑定操作时必须给出数据的大小。否则缺省按照4字节整型处理。

单字节整型在GBase 8s并没有对应类型， 但是在GCI接口操作整型数据的过程中， 仍可以使用单字节的char类型或者unsigned char类型进行操作。在此情况下， 查询整型数据有可能会出现溢出错误。

针对单字节整型，GBase 8s没有对应的单独类型支持，可以使用其他的整型类型与之对应。在GCI接口中可以正常使用单字节整型变量，其范围是- 127至127，不在这个范围的会报错。其绑定变量长度为1字节。

目前GCI接口库中对于无符号整型的处理并没有完整支持。

3.3.2 浮点

GBase 8s支持的浮点类型数据有：

C语言类型(64位系统)	精度	数据库类型名称	SQLT类型
float	7	Real/SMALLFLOAT/DEC(N) N<7	SQLT_FLT/SQLT_BFLOAT/SQLT_IBFLOAT
short	15	FLOAT/double precision/DEC(N) n>7	SQLT_FLT/SQLT_BDOUBLE/SQLT_IBDOUBLE
16位精度以上用字符串	32	Dec(n)	SQLT_CHR

GCI接口在使用浮点数类型做数据库访问操作时用以下类型进行指定：

- SQLT_FLT：表示使用的是单或双精度浮点数类型。使用这个类型的时候，需要在绑定时指定绑定变量的大小。如 C 程序使用 float 类型变量时，指定大小是 sizeof(float)，即 4.2 是 double 类型，其大小是 8。

- SQLT_BFLOAT/SQLT_IBFLOAT：在 Oracle 系统中，这两个类型分别对应单精度浮点和二进制单精度浮点类型。 GBase8s 系统中两者统一为单精度浮点。使用该类型进行数据访问时， 不需指定变量的缓存大小， 其大小默认为 4，不必在绑定接口中明确指定， 即忽略该参数。
- SQLT_BDOUBLE/SQLT_IBDOUBLE：GBbase8s 系统中两者统一为双精度浮点类型， 即 C 程序变量类型为 double，其缓存大小默认为 8，不必在绑定接口中明确指定， 即忽略该参数。
- 在 C/C++程序中，还没有比双精度浮点更高精度的类型。因此对于精度大于 16 的数值， 在 GCI 中，目前是可以通过字符串类型进行访问。

3.3.3 数字类型

数字类型主要是指使用SQLT_NUM类型的处理。

首先，在获取类型时，GCI接口根据不同的参数配置，对于数字类型，返回的类型有所不同。具体见表：

Factory / 数字类型	1（缺省值）	2	3	备注
SQL_INTEGER	SQLT_INT	SQLT_NUM	SQLT_NUM	四字节整型
SQL_DECIMAL	SQLT_CHR	SQLT_NUM	SQLT_NUM	dec
SQL_SMALLINT	SQLT_INT	SQLT_NUM	SQLT_NUM	两字节整型
SQL_INFX_BIGINT	SQLT_INT	SQLT_NUM	SQLT_NUM	bigint
SQL_BIGINT	SQLT_INT	SQLT_NUM	SQLT_NUM	Int8
SQL_FLOAT	SQLT_FLT	SQLT_FLT	SQLT_FLT	float
SQL_REAL	SQLT_FLT	SQLT_FLT	SQLT_FLT	smallfloat

其次，在插入数据过程中，使用SQLT_NUM类型作为绑定变量的类型时，应用程序须使用 字符串类型缓存进行绑定。字符串长度可以根据实际字段的精度(数字类型)设定，一般 设置长度为精度加1。

当操作数据库字段的数字精度大于8字节所表示的范围，或者浮点数精度超过15时， GCI 接 口需要使用字符串类型，进行绑定操作。如果使用的是SQLT_NUM类型，也会转为字符串
类型进行操作。

例：

```
Char ch[32] =  ‘1.12345678909876543 21’;

rc=GCIBindByPos(stmt,&Bindp[0],errhp,1,(dvoid*)ch,(sb4)32,SQLT_NUM,NULL,NULL,NULL,0,NULL,GCI_DEFAULT);
```

上例中，类型指定SQLT_NUM ，实际缓存的内容为字符串。

3.4 时间类型

3.4.1 概述

GBase 8s的日期时间类型有：

C类型	GBase 8s 类型	sql数据长度	SQLT类型
GCIDateTime 或 GCIDate	DATE	16字节	SQLT_DAT SQLT_ODT SQLT_TIMESTAMP SQLT_DATE SQLT_TIME
	TIME	16字节	
	DATETIME	16字节	

在不同的应用场景中使用的SQLT的类型是不同的，不同的类型使用的数据结构也不同：

- SQLT_DAT:其数据结构是一个 7 字节的数组。
- SQLT_ODT：其使用的数据结构是 GCIDate 数据结构，详见 SQLT_ODT 章节
- 。 SQLT_TIMESTAMP：是一个对象句柄， 详见 Datetime 对象类型章节

由于GBase 8s与Oracle在时间类型的区别也导致GCI接口根据厂家设定不同的SQLT的类型 使用， 见表：

Factory / Sql类型	1（缺省值）	2	3
DATE	SQLT_DAT	SQLT_DAT	SQLT_TIMESTAMP
TIME	SQLT_DAT	SQLT_DAT	SQLT_TIMESTAMP
DATETIME	SQLT_DAT	SQLT_DAT	SQLT_TIMESTAMP

上表可见SQLT_TIMESTAMP与SQLT_DAT是不同时出现的，这主要是应用程序在获取字 段的sql类型时的情况（主要原因是Oracle时间类型是两种对应到8s只有一种Datetime）。实 际两种类型都可以获取用来绑定数据。

3.4.2 Datetime 对象类型

GBase 8s支持的时间类型数据有时间类型，日期类型，时间戳类型。

在C语言中， 使用GCIDateTime对象表示时， 使用子类型区分。

C语言类型(64位系统)	子类型	数据库类型名称	SQLT类型
GCIDateTime时间	GCI_DTYPE_TIME	datetime hour to second	SQLT_TIME
GCIDateTime日期	GCI_DTYPE_DATE	Datetime year to day	SQLT_DATE

GCIDateTime时间戳	GCI_DTYPE_TIMESTAMP	除去以上类型的其他时间类型， 例如datetime year to fraction(5)，datetime hour to fraction(4).	SQLT_TIMESTAMP

GCI接口在使用时间类型做数据库访问操作时，有如下规则：

- GCIDateTime 时间， 包含小时(范围 0~23)，分钟(范围 0~59)，秒(范围 0~59)，不包含纳秒。
- GCIDateTime 日期，包含年(范围-4712~9999)，月(范围 0~11)，日(范围 1~31)。
- GCIDateTime 时间戳，包含年(范围-4712~9999)，月(范围 0~11)， 日(范围 1~31)， 小时(范围 0~23)，分钟(范围 0~59)， 秒(范围 0~59)，纳秒(范 围 0~999999999)

关于纳秒， GBase8s中datetime的fraction类型的位数范围是1~5，最大只保留5位有效数字， 所以纳秒的低4位会被舍弃。例如，对于datetime year to fraction(5)的数据， 输入值为9999， 存入数据库的值为0。

Datetime对象类型使用举例：

```
/* 分配用于存储数据的变量*/
GCIDateTime *tstmpltz = (GCIDateTime *)NULL;
/* Col1 是时间戳 */
GCIText *sqlstmt = (GCIText *)"SELECT col1 FROM foo";
/* 分配Datetime对象句柄 */
status = GCIDescriptorAlloc(envhp,(void **)&tstmpltz, GCI_DTYPE_TIMESTAMP,
0, (void **)0);
....
status = GCIStmtPrepare (stmthp, errhp, sqlstmt, (ub4)strlen ((char *)sqlstmt),
(ub4)GCI_NTV_SYNTAX, (ub4)GCI_DEFAULT);
/* 绑定对象句柄，注意是指针的地址*/
status = GCIDefineByPos(stmthp, &defnp, errhp, 1, &tstmpltz, sizeof(tstmpltz),
SQLT_TIMESTAMP_LTZ, 0, 0, 0, GCI_DEFAULT);
/*执行和获取*/
GCIStmtExecute(svchp, stmthp, errhp, 1, 0,(GCISnapshot *) NULL,
(GCISnapshot *)NULL, GCI_DEFAULT)
```

3.4.3 SQLT_DAT 类型

SQLT_DAT类型是oracle数据库中特有的一种精度到秒的日期类型封装格式：

- 使用 7 字节缓存表示日期
- 其中年占两个字节， 即第一，二字节。分别表示世纪和年代， 且做加 100 运算。表示范围公元前 4712 年 1 月 1 日至公元 9999 年 12 月 31 日。
- 第三字节表示月，取值范围是 1- 12
- 第四字节表示日，取值范围是 1-31

- 第五字节表示时，取值范围是 1-24
- 第六字节表示分，取值范围是 1-60
- 第七字节表示秒，取值范围是 1-60

其中年的表示8s与oracle是不同的，8s是到0-9999年

Oracle例：

```
Example (for 30-NOV-1992, 3:17 PM)
-------------------------------------------------------
Byte    1       2       3       4       5       6       7
-------------------------------------------------------
Meaning       Century Year   Month   Day     Hour   Minute Second
Values        119       192    11       30      16      18       1
-------------------------------------------------------
```

GCI接口兼容了这个数据类型，如果使用这个类型进行数据绑定，获取到的时间精度只能到秒。

注：如果sql类型是时间类型，目前不支持其插入操作。

3.4.4 SQLT_ODT 类型

SQLT_ODT类型是使用unix时间描述进行数据访问的类型， 一般用来数据插入过程。该类型进行时间类型插入对于应用程序来说是简单高效的，即在获取机器时间直接进行赋值，无需应用程序转换， 方便快捷。其数据结构定义如下：

SQLT_ODT时间类型的数据结构为：

```
struct GCITime
{
        ub2 GCITimeHH;                          /* hours; range is 0 <= hours <=23 */
        ub2 GCITimeMI;                         /* minutes; range is 0 <= minutes <= 59 */
        ub2 GCITimeSS;                         /* seconds; range is 0 <= seconds <= 59 */
};
typedef struct GCITime GCITime;
struct GCIDate
{
   sb2 GCIDateYYYY;                    /* gregorian year; range is -4712 <= year <= 9999 */
   ub2 GCIDateMM;                    /* month; range is 1 <= month < 12 */
   ub2 GCIDateDD;                     /* day; range is 1 <= day <= 31 */
   GCITime GCIDateTime;               /* time */
   sb4 fraction;
};
```

3.5 Interval类型

GBase 8s支持的Interval类型数据两种，Year-Month 和Day-Second。

在C语言中， 都用GCIInterval表示，使用子类型区分。

C语言类型(64位系统)	子类型	数据库类型名称	SQLT类型
GCIInterval Year-Month	GCI_DTYPE_INTERVAL_YM	Interval year to month	SQLT_INTERVAL_YM
GCIInterval Day-Second	GCI_DTYPE_INTERVAL_DS	Interval day to second / interval day to fraction(5)	SQLT_INTERVAL_DS

GCI接口在使用Interval类型做数据库访问操作时， 有如下规则：

- GCIInterval Year-Month，包含年(范围 0~9999），月(范围 0~11)。
- GCIInterval Day-Second，包含日期(范围 0~999999999），小时(范围 0~23)，分钟(范围0~59），秒(范围 0~59)， 纳秒(范围 0~999999）。
  关于纳秒， GBase8s 中 interval 的 fraction 类型的位数范围是 1~5，最大只保留 5 位有效数字，所以纳秒的低 1 位会被舍弃。例如， 对于 interval day to fraction(5) 的数据，输入值为 9，存入数据库的值为 0。
- 关于 Interval 类型的内部转换。

在GCI中使用的Interval类型，不包含GBase 8s支持的Interval的其他类型，例如interval-hour to second ，interval minute to fraction(4)等等。

如果数据库中的interval类型，与GCI使用的类型不一致，会导致数据丢失。为解决这一问题， 在GCI内部进行了数据转换，规则如下：

1. 取得数据库中interval的类型信息，当Interval类型不一致时，进行转化。
2. 取得数据库中interval的largest qualifier。
3. 将GCI传入的数值，转换成largest qualifier的对应类型。
4. 将转换后的值， 传递到数据库。 举例说明：

GCI输入值类型为 Interval day to fraction(3) ，值为‘1 2:3:4.005’。

数据库中类型为Interval hour to second。

由于interval类型不同， 需要转换。

数据库中interval类型的largest qualifier是hour ，所以需要将interval day to fraction(3)转换为 interval hour to second。

1 day等于24 hours，所以转换后为‘26:3:4.005’。由于数据库中的interval类型不包含fraction部分，所以‘005’会被舍弃，最终传递给数据库的值是‘26:3:4’。

3.6 大对象类型

GBase 8s支持的两种智能大对象类型数据可以与OCI大对象对应兼容，即BLOB和CLOB， 分别是二进制大对象和文本大对象。

在C语言中， 都用GCILobLocator表示，子类型也相同。

C语言类型(64位系统)	子类型	数据库类型名称	SQLT类型
GCILobLocator	GCI_DTYPE_LOB	CLOB	SQLT_CLOB
GCILobLocator	GCI_DTYPE_LOB	BLOB	SQLT_BLOB

4 SQL语句的兼容替换

由于GBase 8s与Oracle的很多类型都很接近，但是其SQL语句的定义却不同，导致两者建表 语句不能通用。此时，GCI接口对其进行了SQL语句的替换操作，提高了两者的兼容性。

4.1 建表语句

建表语句，主要是包含createtable 和 alter table 两类SQL语句，两者替换规则相同。类型替换如下表：

数据类型	Oracle类型	GBase 8s替换后
字符串	Varchar2(n) n<=255	Varchar(n)
	Varchar2	lvarchar
	Varchar2(n) n >255	Lvarchar(n)
数字类型	number	dec
	Number(n)	Dec(n,0)
	Number(m,n)	Dec(m,n)
时间类型	Timestamp(0)	timestamp
浮点类型	Float(n) 1<= n <=24	real
	Float(n) 24< n <=126	float
interval	Interval day[n] to second(0)	Interval day[n] to second

替换过程不区分大小写。

4.2 大对象更新以及插入空对象

oracle大对象主要对应GBase 8s的智能大对象类型，BLOB和CLOB两种，其余在GBase 8s中并不支持。

在OCI接口中，实现插入和写入大对象数据过程主要有以下过程：

- 向表中插入数据，其中需要插入的大对象字段，通过使用 empty_blob/empty_clob 函数插入一个空的大对象数据。
- 将刚刚插入的大对象数据查询出来。
- 调用大对象写接口，完成大对象内容写入。

在上述过程的第一步中，使用到了empty_blob/empty_clob两个函数，在GBase 8s系统中是没有支持的，GCI接口库对此进行了适配， 主要将这两个函数转换为：

OCI	GCI
empty_blob	FILETOBLOB
empty_clob	FILETOCLOB

转换过程不区分大小写。

更新大对象为空对象时也做上述替换。

例：

```
OCI:Update lob set c1 = empty_blob() where id = :1

GCI:update lob set c1= filetoblob(‘/dev/null’,’client’) where id = ?
```

当前， 对于空大对象函数的替换操作，暂不能在windows系统中支持。

4.3 存储过程执行

ORACLE系统执行存储过程，可以通过以下sql语句执行

带参数：begin testproc1(1); end;

不带参数： begin testproc1; end;

这种sql语句在GBase 8s中并不能支持，GCI接口将其转换为以下方式:

	oracle	GBase 8s
带参数	Begin testproc1(1); end;	Execute procedure (x);
不带参数	begin testproc1; end;	Execute procedure();

5 参数绑定模式

5.1 概述

GCI接口参数和结果集绑定基本与OCI兼容，但在绑定内存布局的设定上有所限制， 并不能完全按照OCI的方式实现任意的内存布局绑定。

GCI接口参数和结果集绑定主要分两种模式， 行和列模式绑定：

- 行模式情况下，内存申请为一整块，所有数据按照行集模式保存。空值指示和长度指示可以灵活设定。
- 列模式情况， 内存设定较为简单，每列都单独申请内存并执行绑定，包括空值指示和长度指示缓存均为单独设定

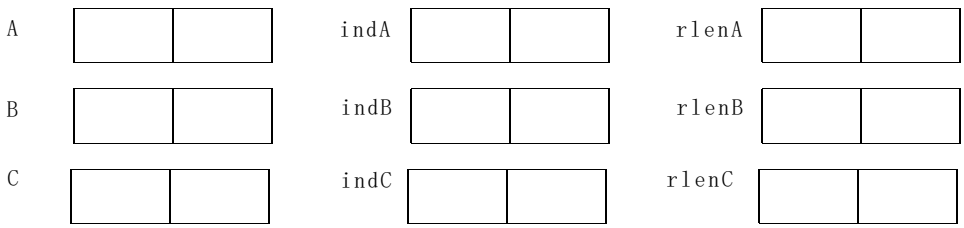
注： 1.目前参数绑定过程，没有支持空值指示和长度指示参数

2.长度指示和空值指示可以不设定， 但设定时其大小需与批量处理的条目数匹配。

5.2 列绑定

GCI接口对于按列式绑定的模式支持相对简单，即每列对应数组，包括空值指示和长度指示。

内存块申请策略如图：



上图中：

有三个字段(A/B/C)访问，分别各自申请长度为2的数组。

与其对应的空值指示也为各自申请长度为2的类型为sb2的数组，长度指示为各自申请长度为2的类型为ub2的数组。

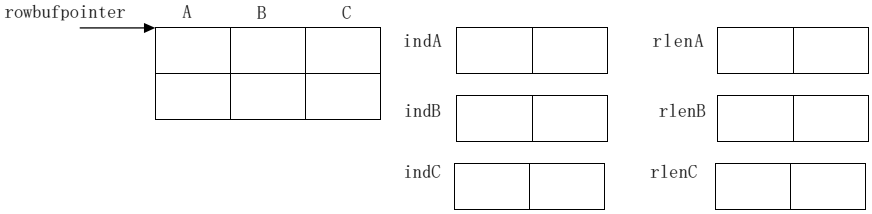
列绑定方式时，应用程序不需要使用GCIBindArrayOfStruct或GCIDefineArrayOfStruct接口指定偏移量(如调用各个偏移量参数可设定为0)

5.3 行绑定

GCI接口对于行绑定的模式支持相对灵活，绑定时根据内存块申请策略不同，设定不同的偏移量。

典型的有：

1. 行集只包含数据，空值指示、长度指示缓存各自独立申请数组。



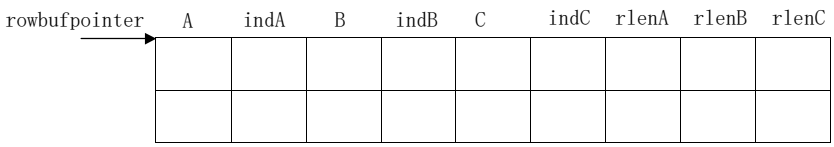
此种情况下，数据缓存按行申请一个内存快， 长度指示与空值指示各自申请对应的缓存数组，空值指示和长度指示与列式绑定申请同样的内存。

此时GCIBindArrayOfStruct或GCIDefineArrayOfStruct调用参数为：

GCIDefineArrayOfStruct(define,pos,rowsize,2,2,0);或

GCIDefineArrayOfStruct(define,pos,rowsize,0,0,0);

2. 行集包含空值指示和长度指示。



此时，内存申请是数据和空值+长度指示的行集数组，只有一个内存块，偏移量设定统一为行长(包含空值和长度指示）。

设定偏移量：GCIDefineArrayOfStruct(define,pos,rowsize,rowsize,rowsize,0);

3. 行集， 空值指示，长度指示各自申请缓存块。

此时， 内存申请总计为三块，分别是行集，空值指示缓存以及长度指示缓存块。

设定偏移量：GCIDefineArrayOfStruct(define,pos,rowsize,6,6,0);

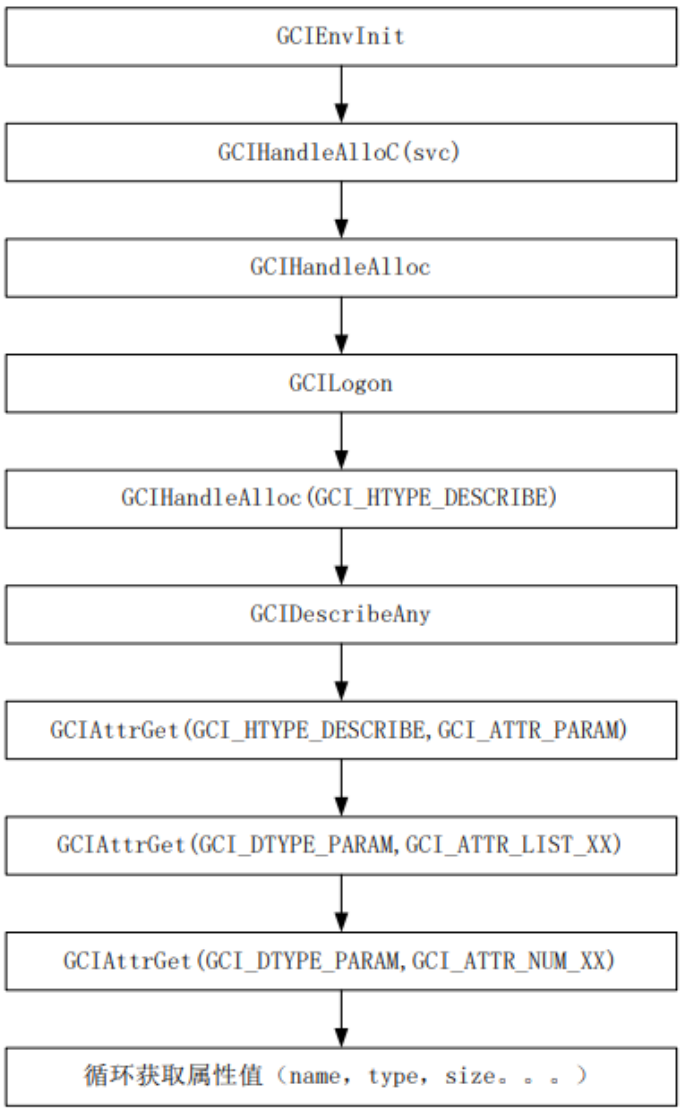
注： GCI接口在行绑定模式中， 数据集的行集是必须分配一整块内存的。只有空值和长度指示内存块可以灵活的进行设定。

6 数据库对象访问

Oracle数据库对象访问类似于ODBC的目录函数，主要用于获取数据库对象的描述信息。GCI接口目前支持四种类型的访问， 分别是：

- GCI_PTYPE_PKG：包对象访问
- GCI_PTYPE_PROC：存储过程或函数对象的访问
- GCI_PTYPE_TABLE：表对象
- GCI_PTYPE_VIEW：视图

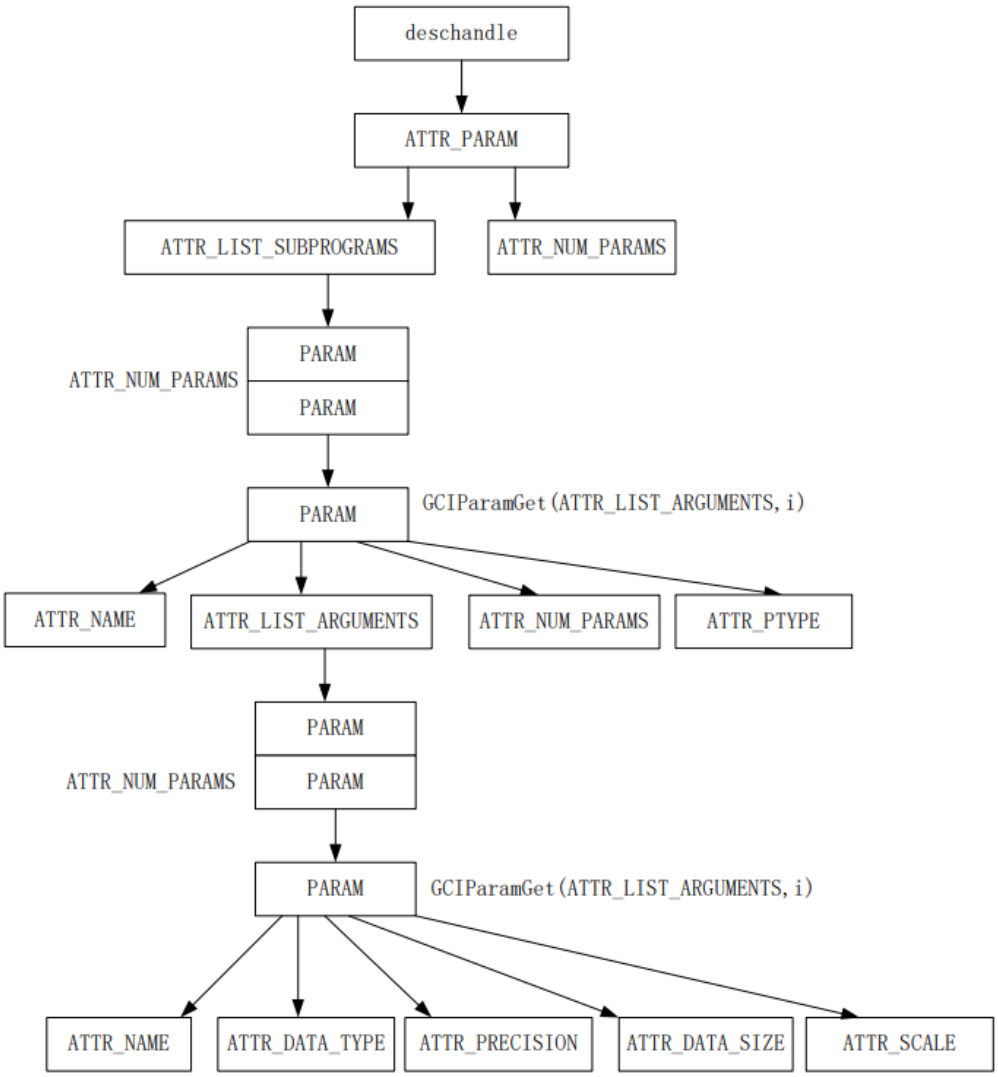
6.1 访问过程



6.2 包对象访问

GBase 8s数据库中，并没有包的概念，GCI接口仅是用作适配。其本身意义是一个存储过程和函数定义的集合。包对象访问过程一般是包含两层，但每层的访问过程是一致的(用到的属性不同）：一是包本身
二是包内某个存储过程或函数

具体访问对象的取值过程如下图：

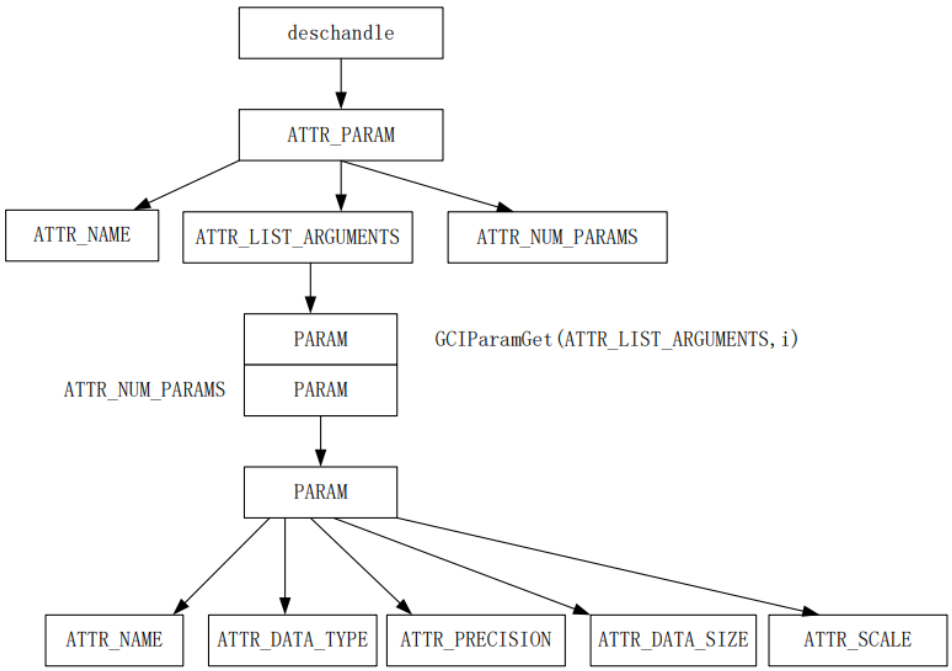


去除list对象节点外，其他均为DTYPE_PARAM属性，使用GCIAttrGet接口获取。

List对象的每个元组则用GCIParamGet接口获取。

6.3 存储过程对象访问

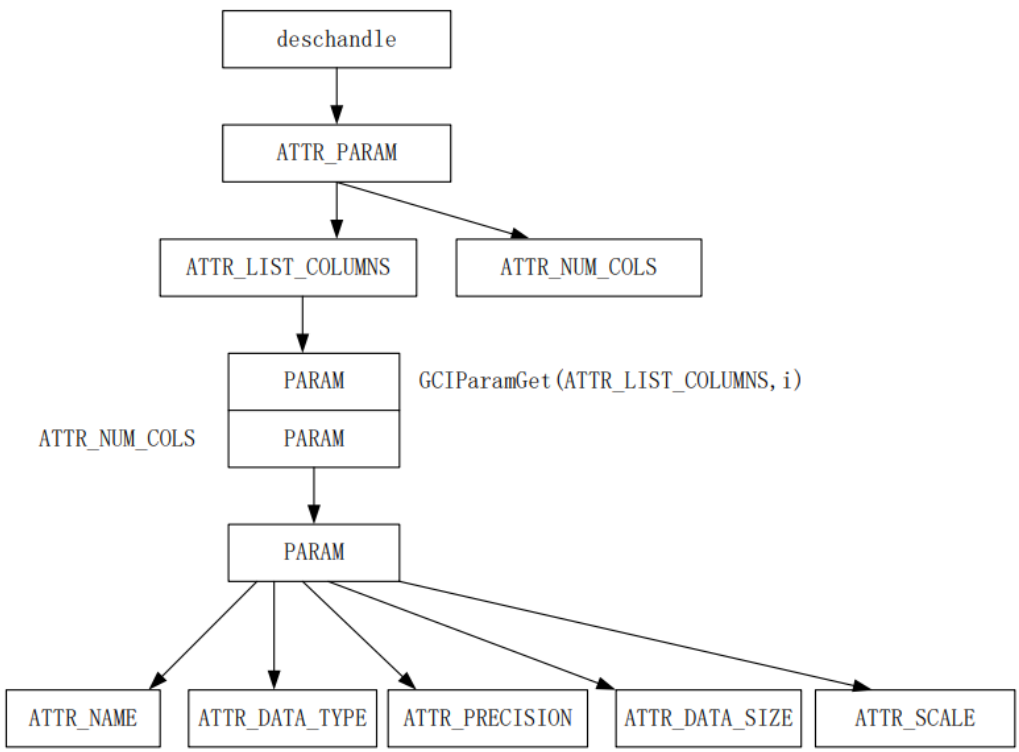
存储过程对象信息访问就是在包对象访问中有所体现。



6.4 表和视图访问

表和视图的访问目前GCI接口只支持其列属性的描述信息获取功能。

访问过程与存储过程访问基本一致，只是用到的属性略有不同。



7 GCI函数说明

7.1 通用接口

7.1.1 GCIInitialize

函数原型：

```
sword   GCIInitialize(
                ub4 mode,
                dvoid *ctxp,
                dvoid *(*malocfp)(dvoid *ctxp, size_t size),
                dvoid *(*ralocfp)(dvoid *ctxp, dvoid *memptr, size_t newsize),
                void (*mfreefp)(dvoid *ctxp, dvoid *memptr)
                   );
```

功能描述：

初始化GCI全局环境， GCI会在这个函数中初始化内部的全局变量和加载一些配置信息，这 是使用GCI与数据库建立连接的第一步。

参数说明：

参数	输入/输出	说明
mode	输入	初始化模式。取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式
ctxp	输入/输出	保留参数， 目前不使用(仅用于和Oracle保持兼容)
malocfp	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
ralocfp	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
mfreefp	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

该函数在每个应用中只需调用一次。

7.1.2 GCIEnvInit

函数原型：

```
sword   GCIEnvInit(
                GCIEnv **envp，
                ub4 mode，
                size_t xtramem_sz,
                dvoid **usrmempp
                )；
```

功能描述：

分配并初始化GCI环境句柄，此函数必须在GCIInitialize之后调用。

参数说明：

参数	输入/输出	说明
envp	输出	输出一个生成的环境句柄指针， 应用可以在环境句柄上分配其他类型的句柄
mode	输入	初始化模式，取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式
xtramem_sz	输入	保留参数，目前不使用（仅用于和Oracle保持兼容）
usrmempp	输出	保留参数，目前不使用（仅用于和Oracle保持兼容）

返回值：

如果执行成功， 则 返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

当调用该函数初始化一个环境句柄以后， 必需调用GCIHandlFree来释放这个句柄。需要注 意的是，在环境句柄分配以后， 只允许在上面分配一个上下文句柄，并只允许建立一个连 接， 但是在连接建立后， 允许分配多个语句句柄。

7.1.3 GCIEnvCreate

函数原型：

```
sword   GCIEnvCreate(
                    GCIEnv **envhpp,
                    ub4 mode,
                    CONST dvoid *ctxp,
                    CONST dvoid *(*malocfp)(
                                dvoid *ctxp,
                                size_t size
                                ),
                    CONST dvoid *(*ralocfp)(
                                dvoid *ctxp,
                                dvoid *memptr,
                                size_t newsize
                                ),
                    CONST void (*mfreefp)(
                                dvoid *ctxp,
                                dvoid *memptr
                                ),
                    size_t xtramemsz,
                    dvoid **usrmempp
                   );
```

功能描述：

创建GCI全局环境，创建并初始化环境句柄，同时在该函数中GCI将加载相关全局变量和配 置信息，以完成后续的连接步骤。

参数说明：

参数	输入/输出	说明
envhpp	输出	输出一个生成的环境句柄指针，应用程序可以在环境句柄上分配其他的句柄
mode	输入	初始化模式，取值如下： GCI_DEFAULT:缺省模式 GCI_THREADED:多线程模式
ctxp	输入/输出	保留参数， 目前不使用(仅用于和 Oracle 保持兼容)
malocfp	输入	保留参数， 目前不使用(仅用于和 Oracle 保持兼容)
ralocfp	输入	保留参数， 目前不使用(仅用于和 Oracle 保持兼容)
mfreefp	输入	保留参数， 目前不使用(仅用于和 Oracle 保持兼容)
xtramemsz	输入	保留参数， 目前不使用(仅用于和 Oracle 保持兼容)
usrmempp	输出	保留参数， 目前不使用(仅用于和 Oracle 保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

当调用该函数初始化一个环境句柄以后， 必须在程序的最后调用GCIHandleFree来释放这个 句柄。同时，当环境句柄分配以后， 目前只允许在上面分配一个上下文句柄，并只允许建 立一个连接，但连接建立之后允许分配多个语句句柄。

7.1.4 GCIHandleAlloc

函数原型：

```
sword   GCIHandleAlloc(
                    dvoid *parenth,
                    dvoid **hndlpp,
                    CONST ub4 type,
                    CONST size_t xtramem_sz,
                    dvoid **usrmempp
                  );
```

功能描述：

分配并初始化各类句柄。

参数说明：

参数	输入/输出	说明
parenth	输入	一个环境句柄的指针，从该句柄上可以分配其他类型的句柄
hndlpp	输出	返回新分配的句柄指针
type	输入	指定要从环境句柄上分配的句柄类型，有： ● GCI_HTYPE_SERVER：分配一个GCIServer结构的句柄，该句柄上存放建立的连接句柄 ● GCI_HTYPE_SESSION：分配一个GCISession结构的句柄，该句柄上存放建立连接所需的登录信息，如用户名和密码等 ● GCI_HTYPE_SVCCTX：分配一个GCISvcCtx结构的句柄(上下文句柄)，该句柄上存放的是其他各类句柄的关联信息 ● GCI_HTYPE_STMT：分配一个GCIStmt结构的句柄(语句句柄)，分配该句柄时，环境句柄上必须已经建立了上下文句柄，并已连接到服务器 ● GCI_HTYPE_DIRPATH_CTX：分配一个GCIDirPathCtx结构句柄(直接文件操作环境上下文) ● GCI_HTYPE_DIRPATH_COLUMN_ARRAY：分配一个GCIDirPathColArray结构句柄(数据数组描述符) ● GCI_HTYPE_DIRPATH_STREAM：分配一个GCIDirPathStream结构句柄(数据流描述符) ● GCI_HTYPE_ERROR：分配一个GCIError结构的句柄(错误句柄)，该句柄存放操作失败后的错误码和相关错误信息 ● GCI_HTYPE_DESCRIBE：分配一个GCIDescribe结构的句柄(描述符句柄)，该句柄存放数据库对象描述符操作信息
xtramem_sz	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
usrmempp	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

调用该函数分配句柄成功以后，需要在程序的最后调用GCIHandleFree来释放相应的句柄。

7.1.5 GCIHandleFree

函数原型：

```
sword   GCIHandleFree(
                    dvoid *hndlp,
                    CONST ub4 type
                );
```

功能描述：

释放已分配的各类句柄，用于结束当前会话、断开数据库连接后的句柄清理。

参数说明：

参数	输入/输出	说明
hndlp	输入	需要释放的句柄
type	输入	释放句柄的句柄类型。取值参见GCIHandleAlloc函数中type参数的取值，环境句柄如下： ● GCI_HTYPE_ENV：释放环境句柄 ● GCI_HTYPE_SVCCTX：释放上下文句柄 ● GCI_HTYPE_SERVER：释放服务句柄 ● GCI_HTYPE_SESSION：释放会话句柄 ● GCI_HTYPE_ERROR：释放错误句柄 ● GCI_HTYPE_STMT：释放语句句柄 ● GCI_HTYPE_DIRPATH_CTX：释放dirpath上下文句柄 ● GCI_HTYPE_DIRPATH_COLUMN_ARRAY：释放dirpath类数组句柄 ● GCI_HTYPE_DIRPATH_STREAM：释放dirpath流句柄 ● GCI_HTYPE_DESCRIBE：释放描述符句柄

返回值：

如果释放成功， 则返回GCI_SUCCESS，释放失败，则返回GCI_ERROR。

注释：

该函书需要配合GCIHandleAlloc或GCIEnvCreate使用。

7.1.6 GCIServerAttach

函数原型：

```
sword    GCIServerAttach(
                        GCIServer *srvhp,
                        GCIError *errhp,
                        CONST GCIText *dblink,
                        sb4 dblink_len,
                        ub4 mode
                            );
```

功能描述：

将一个数据库服务挂载到一个指定的连接句柄上。

参数说明：

参数	输入/输出	说明
srvhp	输入	连接句柄指针，该句柄会跟输入的服务名相关联
errhp	输入	错误信息句柄，该接口调用失败时， 相关错误码及错误信息会保存到该句柄上
dblink	输入	要关联的数据库服务名
dblink_len	输入	数据库服务名的长度

mode	输入	附加模式，取值如下： GCI_DEFAULT:缺省模式 GCI_THREADED:多线程模式

返回值：

如果执行成功，则返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

调用该接口之后，可在适当的时候调用GCIServerDetach接口来解除连接与服务名之间的关 联。

7.1.7 GCIServerDetach

函数原型：

```
sword    GCIServerDetach(
                    GCIServer *srvhp,
                    GCIError *errhp,
                    ub4 mode
                        );
```

功能描述：

解除连接句柄与数据库服务名之间的关联。

参数说明：

参数	输入/输出	说明
srvhp	输入	连接句柄， 该句柄将会被解除与数据库服务名的关联
errhp	输入	错误信息句柄， 接口调用失败时，错误码和错误信息会被写入该句柄中
mode	输入	附加模式， 取值如下： GCI_DEFAULT:缺省模式 GCI_THREADED:多线程模式

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.8 GCISessionBegin

函数原型：

```
sword    GCISessionBegin(
                    GCISvcCtx *svchp,
                    GCIError *errhp,
                    GCISession *usrhp,
                    ub4 credt,
                    ub4 mode
                        );
```

功能描述：

使用登录信息在指定连接句柄上打开与数据库服务的连接。

参数说明：

参数	输入/输出	说明
svchp	输入	指定打开连接的上下文， 在此之前， 上下文必须已经被关联到了连接句柄
errhp	输入	错误信息句柄， 接口调用失败时，错误码和错误信息会被写入该句柄中
usrhp	输入	登录信息句柄， 执行登录之前， 该句柄上必须已经设置了登录的用户名和口令
credt	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
mode	输入	连接模式， 取值如下： GCI_CRED_RDBMS：用户名/密码模式 GCI_CRED_EXT:外部整数模式(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

调用该接口进行连接后， 须在适当的位置调用GCISessionEnd接口来结束该连接。

7.1.9 GCISessionEnd

函数原型：

```
sword   GCISessionEnd(
                    GCISvcCtx *svchp,
                    GCIError *errhp,
                    GCISession *usrhp,
                    ub4 mode
                        );
```

功能描述：

结束GCISessionBegin中连接类句柄与数据库服务之间的连接。

参数说明：

参数	输入/输出	说明
svchp	输入	指定断开连接的上下文
errhp	输入	错误信息句柄，该接口调用失败时将错误码及错误信息写入该句柄
usrhp	输入	登录信息句柄
mode	输入	连接模式， 取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED ： 多线程模式

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

在解除关联之前，如果连接上存在未提交的事务， 则解除连接后将全部回滚。

7.1.10 GCILogon

函数原型：

```
sword   GCILogon(
                GCIEnv *envhp,
                GCIError *errhp,
                GCISvcCtx **svchp,
                CONST GCIText *username,
                ub4 uname_len,
                CONST GCIText *password,
                ub4 passwd_len,
                CONST GCIText *dbname,
                ub4 dbname_len
        );
```

功能描述：

根据数据库服务名、用户名和密码，登录到一个指定的数据库服务上，并初始化相关上下文句柄。

参数说明：

参数	输入/输出	说明
envhp	输入	操作所在的环境句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码及错误信息写入该句柄
svchp	输入/输出	该接口内会自动生成一个当前环境句柄下的上下文句柄，并将地址保存到该指针
username	输入	登录的用户名
uname_len	输入	登录的用户名的长度
password	输入	登录的口令
passwd_len	输入	登录的口令的长度
dbname	输入	当GCI参数connect_mode为1时表示数据库服务名，为2时表示为数据源名称
dbname_len	输入	dbname的长度

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

建立连接之后需要通过调用GCILogoff来断开连接。当dbname为空时使用默认数据库gci_demodb。

7.1.11 GCILogon2

函数原型：

```
sword   GCILogon2(
                GCIEnv  *envhp,
                GCIError *errhp,
                GCISvcCtx **svchp,
                CONST GCIText *username,
                ub4 uname_len,
                CONST GCIText *password,
                ub4 passwd_len,
                CONST GCIText *dbname,
                ub4 dbname_len,
                ub4 mode
                        );
```

功能描述：

根据数据库服务名、用户名和密码，登录到一个指定的数据库服务上，并初始化相关上下文句柄。可以使用现有连接池中的连接。

参数说明：

参数	输入/输出	说明
envhp	输入	GCI 环境句柄。对于连接池和会话池，这必须是在其中创建相应池的池
errhp	输入/输出	错误信息句柄，该接口调用失败时将错误码及错误信息写入该句柄
svchp	输入/输出	该接口内会自动生成一个当前环境句柄下的上下文句柄，并将地址保存到该指针
username	输入	登录的用户名
uname_len	输入	登录的用户名的长度
password	输入	登录的口令
passwd_len	输入	登录的口令的长度
dbname	输入	数据库服务名
dbname_len	输入	数据库服务名的长度
mode	输入	连接模式，取值如下： ● GCI_DEFAULT:缺省模式 ● GCI_LOGON2_CPOOL:连接池模式

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.12 GCILogoff

函数原型：

```
sword   GCILogoff(
                        GCISvcCtx *svchp,
                        GCIError *errhp
                ) ;
```

功能描述：

断开通过GCILogon与服务器建立的连接。

参数说明：

参数	输入/输出	说明
svchp	输入	要断开连接的上下文句柄
errhp	输入	错误信息句柄，调用接口失败时将错误码与错误信息写入该句柄

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

解除连接时，若存在未提交的事务， 则解除连接后将全部回滚。

7.1.13 GCIStmtPrepare

函数原型：

```
sword    GCIStmtPrepare(
                        GCIStmt *stmtp,
                        GCIError *errhp,
                        GCIText *stmt,
                        ub4 stmt_len,
                        ub4 language,
                        ub4 mode
                      );
```

功能描述：

准备一条SQL语句，以便随后调用GCIStmtExecute来执行。

参数说明：

参数	输入/输出	说明
stmtp	输入	用来准备执行的语句句柄
errhp	输入	错误信息句柄，该接口调用失败时，将错误码和错误信息写入该句柄
stmt	输入	准备执行的SQL语句
stmt_len	输入	准备执行的SQL语句的长度
language	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
mode	输入	准备模式，取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功，则返回GCI_SUCCESS，否则返回GCI_ERROR。

注释：

在调用该函数成功准备一条SQL以后， 可以多次调用GCIStmtExecute执行这条SQL语句。如 果在调用该函数执行以后，对SQL上的参数或结果集进行了绑定操作， 那么，只有在下一 次调用该函数以后这些绑定信息才会解除。SQL语句的格式既可以是单条的SQL语句也可以 是多条SQL组合成的一条语句。

7.1.14 GCIStmtPrepareWithSvc

函数原型：

```
sword    GCIStmtPrepareWithSvc(
                        GCISvcCtx *svchp,
                        GCIStmt *stmtp,
                        GCIError *errhp,
                        GCIText *stmt,
                        ub4 stmt_len,
                        ub4 language,
                        ub4 mode
                    );
```

功能描述：

准备一条SQL语句，以便随后调用GCIStmtExecute来执行。

参数说明：

参数	输入/输出	说明
svchp	输入	执行所用的上下文句柄
stmtp	输入	用来准备执行的语句句柄
errhp	输入	错误信息句柄，该接口调用失败时，将错误码和错误信息写入该句柄
stmt	输入	准备执行的SQL语句
stmt_len	输入	准备执行的SQL语句的长度
language	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
mode	输入	准备模式，取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功，则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.15 GCIStmtExecute

函数原型：

```
sword    GCIStmtExecute(
                        GCISvcCtx *svchp,
                        GCIStmt *stmtp,
                        GCIError *errhp,
                        ub4 iters,
                        ub4 rowoff,
                        CONST GCISnapshot *snap_in,
                        GCISnapshot *snap_out,
                        ub4 mode
                    );
```

功能描述：

执行通过GCIStmtPrepare准备后的语句。

参数说明：

参数	输入/输出	说明
svchp	输入	执行所用的上下文句柄
stmtp	输入	用来执行的语句句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄
iters	输入	执行操作的行数。 ● 执行select语句时，如果大于等于1，GCIStmtExecute会执行记录预取功能， 此时未进行结果集绑定， 将返回失败；如果是0，那么不会预取，结果集记录获取需要在fetch过程完成。 ● 执行其他语句时，设定为0或1时，表示只执行一行；设定大于1，表示批量执行，即一次执行多行。
rowoff	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
snap_in	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
Snap_out	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
mode	输入	● GCI_COMMIT_ON_SUCCESS – 自动提交模式，当SQL执行以后， 会自动提交执行的SQL ● GCI_DEFAULT – 默认模式，当SQL执行以后，不自动提交执行的SQL

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.16 GCIStmtFetch

函数原型：

```
sword   GCIStmtFetch(
                    GCIStmt *stmtp,
                    GCIError *errhp,
                    ub4 nrows,
                    ub2 orientation,
                    ub4 mode
                );
```

功能描述：

获取SQL生成的结果集中的行集。当执行一条查询以后， 可以多次调用该函数来提取结果 集中的数据行， 直到该函数返回GCI_NO_DATA ，说明结果集中的数据行已全部获得。

参数说明：

参数	输入/输出	说明
stmtp	输入	用来提取结果集的语句句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄
nrows	输入	一次操作需要获取的行数
orientation	输入	行集提取的方式，可以有以下几种方式： GCI_FETCH_NEXT：从当前游标位置向下进行提取操作 GCI_FETCH_FIRST：（仅用于和Oracle保持兼容） GCI_FETCH_LAST： （仅用于和Oracle保持兼容） GCI_FETCH_PRIOR： （仅用于和Oracle保持兼容）
mode	输入	提取模式， 取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功， 但在提取结果集时出现警告性错误(如字符串截断等），则返回GCI_SUCCESS_WITH_INFO，如果执行成功， 但结果集返回的行数小于iters参数指定的行 数，则返回GCI_NO_DATA，执行正常返回GCI_SUCCESS，执行出错返回GCI_ERROR。

7.1.17 GCIStmtFetchWithSvc

函数原型：

```
sword   GCIStmtFetchWithSvc(
                    GCISvcCtx *svchp,
                    GCIStmt *stmtp,
                    GCIError *errhp,
                    ub4 nrows,
                    ub2 orientation,
                    ub4 mode
                );
```

功能描述：

获取SQL生成的结果集中的行集。当执行一条查询以后， 可以多次调用该函数来提取结果 集中的数据行， 直到该函数返回GCI_NO_DATA ，说明结果集中的数据行已全部获得。

参数说明：

参数	输入/输出	说明
svchp	输入	执行所用的上下文句柄
stmtp	输入	用来提取结果集的语句句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄
nrows	输入	一次操作需要获取的行数
orientation	输入	行集提取的方式，可以有以下几种方式： GCI_FETCH_NEXT：从当前游标位置向下进行提取操作 GCI_FETCH_FIRST：（仅用于和Oracle保持兼容） GCI_FETCH_LAST： （仅用于和Oracle保持兼容） GCI_FETCH_PRIOR： （仅用于和Oracle保持兼容）
mode	输入	提取模式， 取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功， 但在提取结果集时出现警告性错误(如字符串截断等） ，则返回GCI_SUCCESS_WITH_INFO，如果执行成功， 但结果集返回的行数小于iters参数指定的行 数，则返回GCI_NO_DATA，执行正常返回GCI_SUCCESS，执行出错返回GCI_ERROR。

7.1.18 GCITransStart

函数原型：

```
sword    GCITransStart(
                    GCISvcCtx *svchp,
                    GCIError *errhp,
                    uword timeout,
                    ub4 flags
                );
```

功能描述：

开启事务处理模式。

参数说明：

参数	输入/输出	说明
svchp	输入	要开启事务模式的上下文句柄
errhp	输入	错误信息句柄，调用该接口失败时将错误码和错误信息写入该句柄。可设置为空
timeout	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
flags	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.19 GCITransCommit

函数原型：

```
sword   GCITransCommit(
                    GCISvcCtx *svchp,
                    GCIError *errhp,
                    ub4 flags
                  );
```

功能描述：

提交SQL的执行动作。

参数说明：

参数	输入/输出	说明
svchp	输入	要提交的上下文句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄，可设置为空
flags	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.20 GCITransRollback

函数原型：

```
sword   GCITransRollback(
                    GCISvcCtx *svchp,
                    GCIError *errhp,
                    ub4 flags
                    );
```

功能描述：

回滚SQL的执行动作。

参数说明：

参数	输入/输出	说明
svchp	输入	要回滚的上下文句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码与错误信息写入该句柄，可设置为空
flags	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR

7.1.21 GCIBindByName

函数原型：

```
sword   GCIBindByName(
                    GCIStmt *stmtp,
                    GCIBind **bindp,
                    GCIError *errhp,
                    CONST GCIText *placeholder,
                    sb4 place_len,
                    dvoid *valuep,
                    sb4 value_sz,
                    ub2 dty,
                    dvoid *indp,
                    ub2 *alenp,
                    ub2 *rcodep,
                    ub4 maxarr_len,
                    ub4 *curelep,
                    ub4 mode
                  );
```

功能描述：

按参数名称绑定SQL语句中的参数。

参数说明：

参数	输入/输出	说明
stmtp	输入	绑定影响的语句句柄
bindp	输出	绑定信息句柄
errhp	输入	错误信息句柄， 该接口调用失败时将错误码和错误信息写入该句柄
placeholder	输入	绑定的参数名称
placeh_len	输入	参数名称的长度
valuep	输入	参数值缓冲区指针
value_sz	输入	参数类型单个值的大小
dty	输入	参数的数据类型，取值请参考GCI数据类型介绍
indp	输入	控制指示位缓存(仅用于和Oracle保持兼容)
alenp	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
rcodep	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
maxarr_len	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
curelep	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
mode	输入	绑定模式， 取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.22 GCIBindByPos

函数原型：

```
sword    GCIBindByPos(
                    GCIStmt *stmtp,
                    GCIBind **Bindp,
                    GCIError *errhp,
                    ub4 position,
                    dvoid *valuep,
                    sb4 value_sz,
                    ub2 dty,
                    dvoid *indp,
                    ub2 *alenp,
                    ub2 *rcodep,
                    ub4 maxarr_len,
                    ub4 *curelep,
                    ub4 mode
                    );
```

功能描述：

按参数在SQL语句中出现的位置进行绑定。

参数说明：

参数	输入/输出	说明
stmtp	输入	绑定影响的语句句柄
bindp	输出	输出的绑定信息句柄
errhp	输入	错误信息句柄。该接口调用失败时将错误码和错误信息写入该句柄
position	输入	参数在SQL语句中出现的位置， 从1开始计数
valuep	输入	参数值缓冲区指针
value_sz	输入	参数类型单个值的大小
dty	输入	参数的数据类型，取值参见GCI数据类型介绍
indp	输入	空值指示位缓存(仅用于和Oracle保持兼容)
alenp	输入	数据长度指示缓存(仅用于和Oracle保持兼容)
rcodep	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
maxarr_len	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
curelep	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
mode	输入	绑定模式， 取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.23 GCIBindArrayOfStruct

函数原型：

```
sword    GCIBindArrayOfStruct(
                                GCIBind *bindp,
                                GCIError *errhp,
                                ub4 pvskip,
                                ub4 indskip,
                                ub4 alskip,
                                ub4 rcskip
                              );
```

功能描述：

指定参数绑定时，每个参数项中值得间隔大小，以字节计算。

参数说明：

参数	输入/输出	说明
bindp	输入	参数绑定结构指针， 来自GCIBindByName或GCIBindByPos的参数输出
errhp	输入	错误信息句柄， 该接口调用失败时将错误码和错误信息写入该句柄
pvskip	输入	参数值下一个值得间隔长度，在行绑定时，可以通过制定该参数来找到下一行值所在的位置（仅用于行绑定)， 列绑定为0即可。
indskip	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
alskip	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
rcskip	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.24 GCIDefineByPos

函数原型：

```
sword    GCIDefineByPos(
                        GCIStmt *stmtp,
                        GCIDefine **defnp,
                        GCIError *errhp,
                        ub4 position,
                        dvoid *valuep,
                        sb4 value_sz,
                        ub2 dty,
                        dvoid *indp,
                        ub2 *rlenp,
                        ub2 *rcodep,
                        ub4 mode
                      );
```

功能描述：

按位置来指定查询返回结果集中每一列的存储空间。

参数说明：

参数	输入/输出	说明
stmtp	输入	要绑定的语句句柄
defnp	输出	绑定结构输出指针
errhp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄
position	输入	绑定结果集中的列所在的位置， 从1开始计数
valuep	输入	存放获取值的缓冲区指针
value_sz	输入	单个该类型值的大小
dty	输入	绑定的数据类型
indp	输入	空值指示位缓存
rlenp	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
rcodep	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
mode	输入	绑定模式， 取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.25 GCIDefineArrayOfStruct

函数原型：

```
sword   GCIDefineArrayOfStruct(
                    GCIDefine *defnp,
                    GCIError *errhp,
                    ub4 pvskip,
                    ub4 indskip,
                    ub4 rlskip,
                    ub4 rcskip
                    );
```

功能描述：

用来指定行集中每一列中每行值存储位置间隔的大小，以字节计算。

参数说明：

参数	输入/输出	说明
defnp	输入	绑定结构指针， 该参数通过调用GCIDefineByPos后输出
errp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄
pvskip	输入	行绑定时， 每个值间隔大小
indskip	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
rlskip	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
rcskip	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.26 GCIAttrSet

函数原型：

```
sword    GCIAttrSet(
                        dvoid *trgthndlp,
                        ub4 trghndltyp,
                        dvoid *attributep,
                        ub4 size,
                        ub4 attrtype,
                        GCIError *errhp
                       );
```

功能描述：

设置句柄上的属性值。

参数说明：

参数	输入/输出	说明
trgthndlp	输入	需要设置属性的句柄指针
trghndltyp	输入	需要设置的句柄的类型， 可取以下值： ● GCI_HTYPE_SVCCTX上下文句柄 ● GCI_HTYPE_SESSION连接信息句柄 ● GCI_HTYPE_STMT语句句柄 ● GCI_HTYPE_DIRPATH_CTX直接文件操作上下文 ● GCI_DTYPE_PARAM参数信息句柄
attributep	输入	要设置的属性值指针，取值情况如下所示： ● GCI_HTYPE_SVCCTX GCI_ATTR_SERVER - GCIServer结构句柄 GCI_ATTR_SESSION - GCISession结构句柄 ● GCI_HTYPE_SESSION GCI_ATTR_USERNAME - 字符串指针 GCI_ATTR_PASSWORD - 字符串指针 ● GCI_HTYPE_STMT GCI_ATTR_PREFETCH_ROWS - 无符号整形指针 ● GCI_HTYPE_DIRPATH_CTX GCI_ATTR_BUF_SIZE – 无符号整形指针 GCI_ATTR_NAME – 字符串指针 GCI_ATTR_NUM_COLS – 无符号短整形指针 GCI_ATTR_SCHEMA_NAME – 字符串指针 GCI_ATTR_DIRPATH_INPUT – 无符号单字节整形指针 ● GCI_DTYPE_PARAM GCI_ATTR_NAME – 字符串指针 GCI_ATTR_DATA_TYPE – 无符号短整形指针 GCI_ATTR_DATA_SIZE – 无符号整形指针
size	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

参数	输入/输出	说明
attrtype	输入	要设置的句柄属性， 不同类型的句柄有不同的属性： ● GCI_HTYPE_SVCCTX GCI_ATTR_SERVER - 在上下文句柄中附加一个连接句柄 GCI_ATTR_SESSION - 在上下文句柄中附加一个连接信息句柄 ● GCI_HTYPE_SESSION GCI_ATTR_USERNAME - 在连接句柄中设置登录的用户名 GCI_ATTR_PASSWORD - 在连接句柄中设置登录的口令 ● GCI_HTYPE_STMT GCI_ATTR_PREFETCH_ROWS - 在语句句柄上设置预取的结果集行数 ● GCI_HTYPE_DIRPATH_CTX GCI_ATTR_BUF_SIZE – 直接文件操作时缓冲区的大小 GCI_ATTR_NAME – 直接文件操作时的表名 GCI_ATTR_NUM_COLS – 直接文件操作时表的列数 GCI_ATTR_SCHEMA_NAME – 保留参数，用于兼容Oracle，无实际功能 GCI_ATTR_DIRPATH_INPUT – 保留参数，用于兼容Oracle，无实际功能 ● GCI_DTYPE_PARAM GCI_ATTR_NAME – 直接文件操作时设置列的名称 GCI_ATTR_DATA_TYPE – 直接文件操作时设置列的数据类型 GCI_ATTR_DATA_SIZE – 直接文件操作时设置列的数据类型大小，以字节计
errhp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.27 GCIAttrGet

函数原型：

```
sword   GCIAttrGet(
                    CONST dvoid *trgthndlp,
                    ub4 trghndltyp,
                    dvoid *attributep,
                    ub4 *sizep,
                    ub4 attrtype,
                    GCIError *errhp
            );
```

功能描述：

获取句柄上的属性值。

参数说明：

参数	输入/输出	说明
trgthndlp	输入	要获取属性的句柄
trghndltyp	输入	目标句柄的类型，取值如下： ● GCI_HTYPE_STMT - 语句句柄 ● GCI_DTYPE_PARAM – 参数句柄 ● GCI_HTYPE_DIRPATH_CTX – 直接文件操作句柄 ● GCI_HTYPE_DIRPATH_COLUMN_ARRAY – 直接文件数据数组句柄
attributep	输出	存放输出属性的缓冲区
sizep	输出	存放输出属性大小的缓冲区
attrtype	输入	要获取的属性， 取值如下： ● GCI_HTYPE_STMT GCI_ATTR_PREFETCH_ROWS – 当前一次性返回结果集记录的行数 GCI_ATTR_NUM_COLS - 结果集中列的个数 GCI_ATTR_ROW_COUNT - 已经返回记录的总行数 GCI_ATTR_PARAM_COUNT – 获取绑定参数的个数 ● GCI_HTYPE_DESCRIBE GCI_ATTR_PARAM-当前描述符参数对象 ● GCI_DTYPE_PARAM GCI_HTYPE_RESULT_COL_ATTR GCI_ATTR_DATA_SIZE – 列的数据大小 GCI_ATTR_DATA_TYPE – 列的数据类型 GCI_ATTR_NAME – 列的名称 GCI_ATTR_PRECISION – 列的精度 GCI_ATTR_SCALE – 列的刻度 GCI_ATTR_TYPE_NAME - 数据类型名称 GCI_ATTR_LIST_COLUMNS 同 GCI_HTYPE_RESULT_COL_ATTR 属性 GCI_ATTR_LIST_COLUMNS GCI_ATTR_LIST_ARGUMENTS GCI_ATTR_NAME – 直接文件操作时设置列的名称 GCI_ATTR_DATA_TYPE – 直接文件操作时设置列的数据类 型 GCI_ATTR_DATA_SIZE – 直接文件操作时设置列的数据类 型大小 GCI_ATTR_PRECISION – 直接文件操作时设置列的精度 GCI_ATTR_SCALE – 直接文件操作时设置列的刻度 GCI_ATTR_TYPE_NAME- 数据类型名称 GCI_PTYPE_PKG GCI_ATTR_LIST_SUBPROGRAMS-数据库程序对象列表 GCI_ATTR_NUM_PARAMS-对象参数的个数 GCI_ATTR_PTYPE-数据库对象类型 GCI_PTYPE_PROC GCI_ATTR_LIST_ARGUMENTS-数据库对象参数列表 GCI_ATTR_NUM_PARAMS-对象参数的个数 GCI_PTYPE_TABLE GCI_PTYPE_VIEW GCI_ATTR_LIST_COLUMNS-表或视图列属性列表 GCI_ATTR_NUM_COLUMNS-列数 ● GCI_HTYPE_DIRPATH_CTX GCI_ATTR_LIST_COLUMNS – 返回描述表中各列信息的参数句柄 ● GCI_HTYPE_DIRPATH_COLUMN_ARRAY GCI_ATTR_NUM_ROWS – 当前缓冲区允许一次性最大设置的记录行数
errhp	输入	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.28 GCIErrorGet

函数原型：

```
sword   GCIErrorGet(
                    dvoid *hndlp,
                    ub4 recordno,
                    GCIText *sqlstate,
                    sb4 *errcodep,
                    GCIText *bufp,
                    ub4 bufsiz,
                    ub4 type
                );
```

功能描述：

获取GCI操作失败的错误信息。

参数说明：

参数	输入/输出	说明
hndlp	输入	错误信息句柄的指针
recordno	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
sqlstate	输出	错误的状态码
errcodep	输出	错误码
bufp	输出	错误的描述信息
bufsiz	输入	bufp所指缓冲区的大小
type	输入	支持GCI_HTYPE_ERROR和GCI_HTYPE_ENV

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.29 GCIServerVersion

函数原型：

```
sword   GCIServerVersion(
                    void    *hndlp,
                    CIError *errhp,
                    sb1 *bufp,
                    ub4 bufsz,
                    ub1 hndltype
                    );
```

功能描述：

得到服务器的版本号。

参数说明：

参数	输入/输出	说明
hndlp	输入	当前连接服务的上下文句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码及错误信息写入该句柄
bufp	输入	返回结果集的缓冲区地址
bufsz	输入	返回结果集的缓冲区的大小
hndltype	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.30 GCIServerSession

函数原型：

```
sword    GCIServerSession(
                        void    *hndlp,
                        GCIError *errhp,
                        sb1 *bufp,
                        ub4 bufsz,
                        ub1 hndltype
                        );
```

功能描述：

得到服务器当前的连接数。

参数说明：

参数	输入/输出	说明
hndlp	输入	当前连接的上下文句柄
errhp	输入	错误信息句柄，该接口调用失败时将错误码及错误信息写入该句柄
bufp	输入	返回结果集的缓冲区地址
bufsz	输入	返回结果集的缓冲区的大小
hndltype	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 则返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.31 GCIParamGet

函数原型：

```
sword    GCIParamGet(
                        CONST dvoid *hndlp,
                        ub4 htype,
                        GCIError *errhp,
                        dvoid **parmdpp,
                        ub4 pos
                    );
```

功能描述：

返回描述句柄或语句句柄中的根据位置指定的参数描述符。

参数说明：

参数	输入/输出	说明
hndlp	输入	一个存放描述信息的句柄
htype	输入	hndlp参数句柄的类型，支持以下两种类型： ● GCI_HTYPE_STMT – 语句句柄 ● GCI_DTYPE_PARAM – 参数句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
parmdpp	输出	输出的描述符句柄， 通过该句柄调用GCIAttrGet函数就可以得到指定位置上的描述信息
pos	输入	要获取描述符上指定位置的描述信息

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.32 CIDescriptorFree

函数原型：

```
sword    GCIDescriptorFree(
                    dvoid *descp,
                    CONST ub4 type
                    );
```

功能描述：

释放描述符。

参数说明：

参数	输入/输出	说明
descp	输入	要释放的描述符指针
type	输入	描述符类型，目前只支持GCI_DTYPE_LOB

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.33 GCIDescriptorAlloc

函数原型：

```
sword    GCIDescriptorAlloc(
                    CONST dvoid *parenth,
                    dvoid **descpp,
                    CONST ub4 type,
                    CONST size_t xtramem_sz,
                    dvoid **usrmempp
                    );
```

功能描述：

获取数据库对象描述符句柄。

参数说明：

参数	输入/输出	说明
parenth	输入	上下文句柄指针
descpp	输出	描述符句柄
type	输入	获取数据库对象类型，支持类型有： ● GCI_DTYPE_LOB大对象类型 ● GCI_DTYPE_DATE日期对象类型 ● GCI_DTYPE_TIMESTAMP时间戳对象类型 ● GCI_DTYPE_TIME 时间对象类型 ● GCI_DTYPE_INTERVAL_DS Day to second间隔对象类型 ● GCI_DTYPE_INTERVAL_YM year to month间隔对象类型
xtramem_sz	输入	保留参数，objptr参数中的字符串长度
usrmempp	输出	保留参数，目前不适用(仅用于和Oracle调用保持兼容)

返回值：

执行成功返回GCI_SUCCESS。否则返回GCI_ERROR。

7.1.34 GCIParamSet

函数原型：

```
sword    GCIParamSet(
                        dvoid *hdlp,
                        ub4 htyp,
                        GCIError *errhp,
                        CONST dvoid *dscp,
                        ub4 dtyp,
                        ub4 pos
                );
```

功能描述：

预留

7.1.35 GCIDescribeAny

函数原型：

```
sword    GCIDescribeAny(
                        GCISvcCtx *svchp,
                        GCIError *errhp,
                        dvoid *objptr,
                        ub4 objnm_len,
                        ub1 objptr_typ,
                        ub1 info_level,
                        ub1 objtyp,
                        GCIDescribe *dschp
                      );
```

功能描述：

描述和某个数据库对象， 得到其内部构造的详细信息的指针。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄指针
errhp	输入/输出	错误信息句柄， 该接口调用失败时， 错误信息会保存在错误信息句柄中
objptr	输入	被描述的对象的指针，目前只支持字符串类型指针
objnm_len	输入	objptr参数中的字符串长度
objptr_typ	输入	objptr指针类型， 目前仅支持GCI_OTYPE_NAME这一类型的对象名称指针
info_level	输入	保留参数， 目前不适用(仅用于和Oracle调用保持兼容)
objtyp	输入	objptr参数所致的对象类型，可以为下面几种对象： ● GCI_PTYPE_PROC 存储过程 ● GCI_PTYPE_PKG 对象包 ● GCI_PTYPE_TABLE 表对象 ● GCI_PTYPE_VIEW 视图对象

返回值：

执行成功返回GCI_SUCCESS。否则返回GCI_ERROR。

注：调用该函数对某个对象进行描述以后，需要结合调用GCIParamGet和GCIAttrGet来获取描述符的详细信息。

7.1.36 GCITerminate

函数原型：

```
sword GCITerminate ( ub4 mode);
```

功能描述：

从共享内存子系统中分离(仅用于和Oracle保持兼容)。

参数说明：

参数	输入/输出	说明
mode	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.37 GCIBreak

函数原型：

```
sword GCIBreak (
                    void *hndlp,
                    GCIError *errhp
                    );
```

功能描述：

中断当前的执行。

参数说明：

参数	输入/输出	说明
hndlp	输入	语句或者上下文句柄
errhp	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.38 GCIPasswordChange

函数原型：

```
sword GCIPasswordChange (
                        GCISvcCtx *svchp,
                        GCIError *errhp,
                        const GCIText *user_name,
                        ub4 usernm_len,
                        const GCIText *opasswd,
                        ub4 opasswd_len,
                        const GCIText *npasswd,
                        sb4 npasswd_len,
                        ub4 mode
                            );
```

功能描述：

修改用户名密码。

参数说明：

参数	输入/输出	说明
svchp	输入	指定打开连接的上下文，在此之前，上下文必须已经被关联到了连接句柄
errhp	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
user_name	输入	登录的用户名
usernm_len	输入	登录的用户名的长度
opasswd	输入	登录的旧口令
opasswd_len	输入	登录的旧口令的长度
npasswd	输入	登录的新口令
npasswd_len	输入	登录的新口令的长度
mode	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

注：此处修改的用户指的是非系统用户。

7.1.39 GCIEnvNlsCreate

函数原型：

```
sword GCIEnvNlsCreate (
                        GCIEnv          **envp,
                        ub4              mode,
                        void             *ctxp,
                        void             *(*malocfp)
                          (void        *ctxp,
                          size_t        size),
                        void             *(*ralocfp)
                          (void        *ctxp,
                          void         *memptr,
                          size_t       newsize),
                        void              (*mfreefp)
                          (void      *ctxp,
                            void     *memptr),
                        size_t            xtramem_sz,
                        void              **usrmempp,
                        ub2               charset,
                        ub2               ncharset
                            );
```

功能描述：

创建GCI全局环境，创建并初始化环境句柄，同时在该函数中GCI将加载相关全局变量和配置信息，以完成后续的连接步骤。

参数说明：

参数	输入/输出	说明
envp	输出	指向其编码设置由模式指定的环境句柄的指针。该设置由派生自 envhpp 的语句句柄继承。
mode	输入	指定初始化模式，取值如下： ● GCI_DEFAULT:缺省模式 ● GCI_THREADED:多线程模式
ctxp	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
malocfp	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
ralocfp	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
mfreefp	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
xtramem_sz	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
usrmempp	输出	保留参数，目前不使用(仅用于和Oracle保持兼容)
charset	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
ncharset	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.40 GCIStmtFetch2

函数原型：

```
sword   GCIStmtFetch2(
                    GCIStmt         *stmtp,
                    GCIError        *errhp,
                    ub4             nrows,
                    ub2             orientation,
                    sb4             scrollOffset,
                    ub4             mode
                );
```

功能描述：

从（可滚动的）结果集中获取一行。

参数说明：

参数	输入/输出	说明
stmtp	输入/输出	用来提取结果集的语句句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时将错误码和错误信息写入该句柄
nrows	输入	一次操作需要获取的行数
orientation	输入	行集提取的方式，可以有以下几种方式： GCI_FETCH_NEXT：从当前游标位置向下进行提取操作 GCI_FETCH_FIRST：（仅用于和Oracle保持兼容） GCI_FETCH_LAST：（仅用于和Oracle保持兼容） GCI_FETCH_PRIOR：（仅用于和Oracle保持兼容）
scrollOffset	输入	提取模式，取值如下： ● GCI_DEFAULT：缺省模式 ● GCI_THREADED：多线程模式
mode	输入	提取模式，取值如下： GCI_DEFAULT：缺省模式 GCI_THREADED：多线程模式

返回值：

如果执行成功，但在提取结果集时出现警告性错误(如字符串截断等)，则返回GCI_SUCCESS_WITH_INFO，如果执行成功，但结果集返回的行数小于iters参数指定的行数，则返回GCI_NO_DATA，执行正常返回GCI_SUCCESS，执行出错返回GCI_ERROR。

7.1.41 GCIStmtRelease

函数原型：

```
sword GCIStmtRelease (
                        GCIStmt         *stmtp,
                        GCIError        *errhp,
                        const GCIText   *key,
                        ub4             key_len,
                        ub4             mode
                    );
```

功能描述：

释放通过调用 GCIStmtPrepare2() 获得的语句句柄。

参数说明：

参数	输入/输出	说明
stmtp	输入/输出	GCIStmtPrepare2() 返回的语句句柄
errhp	输入	用于诊断的错误句柄
key	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
key_len	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
mode	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.1.42 GCIStmtPrepare2

函数原型：

```
sword   GCIStmtPrepare2(
                    GCISvcCtx           *svchp,
                    GCIStmt             **stmthp,
                    GCIError            *errhp,
                    const GCIText       *stmttext,
                    ub4                 stmt_len,
                    const GCIText       *key,
                    ub4                 keylen,
                    ub4                 language,
                    ub4                 mode
                );
```

功能描述：

准备一条SQL语句， 以便随后调用GCIStmtExecute来执行。

参数说明：

参数	输入/输出	说明
svchp	输入	要与语句关联的服务上下文
stmtph	输出	指向返回的语句句柄的指针
errhp	输入	指向诊断错误句柄的指针
stmttext	输入	语句文本。stmttext 的语义与 GCIStmtPrepare() 的 stmt 语义相同；也就是说，字符串必须以 NULL 结尾
stmt_len	输入	语句文本长度
key	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
keylen	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
language	输入	保留参数，目前不使用(仅用于和Oracle保持兼容)
mode	输入	准备模式，取值如下： ● GCI_DEFAULT：缺省模式 ● GCI_THREADED：多线程模式

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.1.43 GCIConnectionPoolCreate

函数原型：

```
sword   GCIConnectionPoolCreate (
                    GCIEnv          *envhp,
                    GCIError        *errhp,
                    GCICPool        *poolhp,
                    GCIText         **poolName,
                    sb4             *poolNameLen,
                    const GCIText   *dblink,
                    sb4             dblinkLen,
                    ub4             connMin,
                    ub4             connMax,
                    ub4             connIncr,
                    const GCIText   *poolUserName,
                    sb4             poolUserLen,
                    const GCIText   *poolPassword,
                    sb4             poolPassLen,
                    ub4             mode
                );
```

功能描述：

初始化连接池。

参数说明：

参数	输入/输出	说明
envhp	输入	指向要在其中创建连接池的环境句柄的指针
errhp	输入/输出	可以传递给 GCIErrorGet()的错误句柄
poolhp	输入	指向已初始化的连接池句柄的指针
poolName	输出	返回连接池的名称，它在环境中的所有连接池中唯一
poolNameLen	输出	poolName 指向的连接池名称的长度（以字节为单位）
dblink	输入	要连接的数据库的名称
dblinkLen	输入	dblink 指向的数据库名称的长度(字节为单位)
connMin	输入	指定连接池中的最小连接数
connMax	输入	指定可以在连接池中打开的最大连接数。达到此值后，将不再建立任何连接。有效值为 1 或更高
connIncr	输入	如果当前连接数小于 connMax 时，则允许应用程序为要启动的连接设置下一个增量 connIncr。有效值为 0 或更高。
poolUserName	输入	指定用于启动连接的用户标识
poolUserLen	输入	用户 ID 的长度（以字节为单位）
poolPassword	输入	相应用户标识的密码
poolPassLen	输入	密码的长度（以字节为单位）
mode	输入	支持的模式有： ● GCI_DEFAULT -用于创建新的连接池 ● GCI_CPOOL_REINITIALIZE-改变连接池的属性

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.1.44 GCIConnectionPoolDestroy

函数原型：

```
sword   GCIConnectionPoolDestroy (
                    GCICPool        *poolhp,
                    GCIError        *errhp,
                    ub4             mode
                );
```

功能描述：

释放连接池。

参数说明：

参数	输入/输出	说明
poolhp	输入	连接池句柄用于要销毁的连接池
errhp	输入/输出	可以传递给 GCIErrorGet() 的错误句柄
mode	输入	支持的模式有：GCI_DEFAULT

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.1.45 GCISessionGet

函数原型：

```
sword    GCISessionGet (
                        GCIEnv          *envhp,
                        GCIError        *errhp,
                        GCISvcCtx       **svchp,
                        GCIAuthInfo     *authhp,
                        GCIText          *poolName,
                        ub4              poolName_len,
                        const GCIText    *tagInfo,
                        ub4              tagInfo_len,
                        GCIText          **retTagInfo,
                        ub4              *retTagInfo_len,
                        boolean          *found,
                        ub4              mode
                      );
```

功能描述：

根据数据库服务名、用户名和密码，登录到一个指定的数据库服务上，并初始化相关上下文句柄。可以使用现有连接池中的连接。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	GCI 环境句柄，对于连接池和会话池，为在其中创建相应池的环境句柄
errhp	输入/输出	错误句柄
svchp	输出	GCI 服务上下文指针的地址
authhp	输入	获取连接时要使用的身份验证信息句柄
poolName	输入	● mode 为 GCI_DEFAULT 时，表示连接到 GBase8s 服务器的数据库名称 ● mode 为 GCI_SESSGET_CPOOL 时，表示连接池名称。
poolName_len	输入	poolName 的长度
tagInfo	输入	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
tagInfo_len	输入	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
retTagInfo	输出	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
retTagInfo_len	输出	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
found	输出	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
mode	输入	有效模式为： ● GCI_DEFAULT:缺省模式 ● GCI_SESSGET_CPOOL:连接池模式

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.1.46 GCISessionRelease

函数原型：

```
sword    GCISessionRelease (
                        GCISvcCtx       *svchp,
                        GCIError        *errhp,
                        GCIText         *tag,
                        ub4             tag_len,
                        ub4             mode
                                );
```

功能描述：

释放使用 GCISessionGet() 创建的连接。

参数说明：

参数	输入/输出	说明
svchp	输入	在相应的 GCISessionGet() 调用期间填充的服务上下文。 默认情况下，与此句柄关联的会话和连接将关闭。 在连接池的情况下，会话将关闭，连接将释放到池中。 对于会话池，与此服务上下文关联的会话或连接对将释放到池中。
errhp	输入/输出	错误句柄
tag	输入	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
tag_len	输入	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
mode	输入	有效模式为：GCI_DEFAULT:默认情况和连接池

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.1.47 GCIStmtGetPieceInfo

函数原型：

```
sword    GCIStmtGetPieceInfo (
                        const GCIStmt       *stmtp,
                        GCIError            *errhp,
                        void                **hndlpp,
                        ub4                 *typep,
                        ub1                 *in_outp,
                        ub4                 *iterp,
                        ub4                 *idxp,
                        ub1                 *piecep
                                );
```

功能描述：

获得分段操作中的分段信息。

参数说明：

参数	输入/输出	说明
stmtp	输入	执行时返回 GCI_NEED_DATA 的语句
errhp	输出	错误句柄
hndlpp	输出	返回一个指针，指向需要或正在提供其运行时数据的绑定或定义句柄
typep	输出	hndlpp 指向的句柄的类型： ● GCI_HDYPE_BIND-用于绑定句柄 ● GCI_HDTYPE_DEFINE-用于定义句柄
in_outp	输出	如果 in 绑定值需要数据，则返回 GCI_PARAM_IN;如果数据可用作 OUT 绑定变量或定义位置值，则返回 GCI_PARAM_OUT
iterp	输出	返回多行操作的行号

参数	输入/输出	说明
idxp	输出	PL/SQL 数组绑定操作的数组元素的索引
piecep	输出	返回以下定义的值之一：GCI_ONE_PIECE、GCI_FIRST_PIECE、GCI_NEXT_PIECE 或 GCI_LAST_PIECE

返回值：

GCI_SUCCESS、GCI_ERROR 或 GCI_NEED_DATA。

7.1.48 GCIStmtSetPieceInfo

函数原型：

```
sword    GCIStmtSetPieceInfo (
                    void            *hndlp,
                    ub4             type,
                    GCIError        *errhp,
                    const void      *bufp,
                    ub4             *alenp,
                    ub1             piece,
                    const void      *indp,
                    ub2             *rcodep
                        );
```

功能描述：

设置分段操作中的分段信息。

参数说明：

参数	输入/输出	说明
hndlp	输入/输出	绑定或定义句柄
type	输入	句柄的类型
errhp	输出	错误句柄
bufp	输入/输出	当它是 IN 绑定变量时，指向包含数据值或片段的存储器的指针；否则，bufp 是指向存储的指针，用于获取 OUT 绑定和定义变量的片段或值
alenp	输入/输出	一段的长度，不要在执行同一 SQL 语句之间更改此参数
piece	输入	有效值为： •GCI_ONE_PIECE •GCI_FIRST_PIECE •GCI_NEXT_PIECE •GCI_LAST_PIECE 此参数仅用于 IN 绑定变量
indp	输入/输出	指示器。指向 sb2 值的指针或指向命名数据类型（SQLT_NTY）和 REF（SQLT_REF）的指示符结构的指针，即取决于数据类型，*indp 要么是 sb2，要么是 void
rcodep	输入/输出	返回代码

返回值：

GCI_SUCCESS、GCI_ERROR 或 GCI_NEED_DATA。

7.1.49 GCIBindDynamic

函数原型：

```
sword   GCIBindDynamic (
                        GCIBind              *bindp,
                        GCIError             *errhp,
                        void                 *ictxp,
                        GCICallbackInBind    (icbfp)
                          (void          *ictxp,
                           GCIBind       *bindp,
                           ub4           iter,
                           ub4           index,
                           void          **bufpp,
                           ub4           *alenp,
                           ub1           *piecep,
                           void          **indpp),
                        void                 *octxp,
                        GCICallbackOutBind   (ocbfp)
                          (void          *octxp,
                           GCIBind       *bindp,
                           ub4           iter,
                           ub4           index,
                           void          **bufpp,
                           ub4           **alenpp,
                           ub1           *piecep,
                           void          **indpp,
                           ub2           **rcodepp)
                                );
```

功能描述：

注册用于动态分配数据的用户回调函数。

参数说明：

参数	输入/输出	说明
bindp	输入/输出	调用 GCIBindByName() 或 GCIBindByPos() 返回的绑定句柄
errhp	输入/输出	错误句柄，当出现错误时，可以传递给 GCIErrorGet() 以获取诊断信息
ictxp	输入	回调函数 icbfp 所需的上下文指针
icbfp	输入	在运行时返回指向 IN 绑定值或段的指针的回调函数。 回调采用以下参数： ● ictxp(输入/输出) 此回调函数的上下文指针 ● bindp（输入） 传入以唯一标识此绑定变量的绑定句柄 ● iter (输入) 从 0 开始的执行迭代值。 ● index（输入） 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● bufpp（输出） 指向缓冲区或存储的指针。 ● alenp（输出） 指向存储的指针，GCI 用于在读取绑定值或片段后填充绑定值或片段的大小 ● piecep（输出） 下列值之一： GCI_ONE_PIECE，GCI_FIRST_PIECE， GCI_NEXT_PIECE 或 GCI_LAST_PIECE ● indpp（输出） 包含指示器值，这是指向 sb2 值的指针或指向用于绑定命名数据类型的指示器结构的指针

参数	输入/输出	说明
octxp	输入	保留参数，目前不使用（仅用于和 Oracle 保持兼容）
ocbfp	输入	在运行时返回指向 OUT 绑定值或段的指针的回调函数，保留参数，目前不使用（仅用于和 Oracle 保持兼容）。 回调采用以下参数： ● octxp(输入/输出) 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● bindp（输入） 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● iter (输入) 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● index（输入） 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● bufpp（输出） 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● alenpp(输入/输出) 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● piecep(输入/输出) 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● indpp（输出） 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● rcodepp（输出） 保留参数，目前不使用（仅用于和 Oracle 保持兼容）

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.1.50 GCIDefineDynamic

函数原型：

```
sword    GCIDefineDynamic (
                        GCIDefine               *defnp,
                        GCIError                *errhp,
                        void                    *octxp,
                        GCICallbackDefine       (ocbfp)
                          (void         *octxp,
                          GCIDefine     *defnp,
                          ub4           iter,
                          void          **bufpp,
                          ub4           **alenpp,
                          ub1           *piecep,
                          void          **indpp,
                          ub2           **rcodep )
                                );
```

功能描述：

函数GCIDefineByPos选择GCI_DYNAMIC_FETCH模式时，设置一些额外的属性。

参数说明：

参数	输入/输出	说明
defnp	输入/输出	由调用 GCIDefineByPos() 返回的定义结构的句柄
errhp	输入/输出	错误句柄，当出现错误时，可以传递给 GCIErrorGet() 以获取诊断信息
octxp	输入	指向回调函数的上下文
ocbfp	输入	指向回调函数。在运行时调用它以获取指向要在其中检索提取数据或片段的缓冲区的指针。回调还指定指标、返回码以及数据片段和指标的长度。 回调参数为： ● octxp(输入/输出) 作为参数传递给所有回调函数的上下文指针 ● defnp（输入） 定义句柄 ● iter (输入) 指定当前获取的哪一行；从 0 开始 ● bufpp（输出） 返回指向用于存储列值的缓冲区的指针 ● alenpp(输入/输出) 应用程序用于设置它在 *bufpp 中提供的存储大小。将数据提取到缓冲区后，alenpp 指示数据的实际大小（以字节为单位）。如果第一次调用中提供的缓冲区长度不足以存储服务器返回的所有数据，则再次调用回调，依此类推 ● piecep(输入/输出) 从回调（应用程序）向 GCI 返回一个片段值，该值可以是 GCI_ONE_PIECE，GCI_FIRST_PIECE，GCI_NEXT_PIECE 或 GCI_LAST_PIECE ● indpp（输入） 指标符变量指针。 ● rcodep（输入） 保留参数，目前不使用（仅用于和 Oracle 保持兼容）

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.1.51 GCIStmtGetBindInfo

函数原型：

```
sword   GCIStmtGetBindInfo (
                        GCIStmt         *stmtp,
                        GCIError        *errhp,
                        ub4              size,
                        ub4              startloc,
                        sb4             *found,
                        GCIText         *bvnp[],
                        ub1              bvnl[],
                        GCIText         *invp[],
                        ub1              inpl[],
                        ub1              dupl[],
                        GCIBind         *hndl[]
                              );
```

功能描述：

获取绑定和指示符变量名称。

参数说明：

参数	输入/输出	说明
stmtp	输入	由 GCIStmtPrepare2() 准备的语句句柄。
errhp	输入	错误句柄，当出现错误时，可以传递给 GCIErrorGet() 以获取诊断信息
size	输入	每个数组中元素的数量
startloc	输入	绑定变量的位置，从该位置开始获取绑定信息。
found	输入	表达式 abs（found）给出语句中绑定变量的总数，而与起始位置无关。如果返回的绑定变量的数量小于提供的大小，则为正值，否则为负。
bvnp	输出	用于保存绑定变量名称的指针数组
bvnl	输出	用于保存每个 bvnp 元素长度的数组，长度以字节为单位
invp	输出	用于保存指示符变量名称的指针数组
inpl	输出	用于保存每个 invp 元素长度的指针数组。以字节数为单位。
dupl	输出	元素值为 0 或 1 的数组，具体取决于绑定位置是否是另一个的重复
hndl	输出	数组，如果已为绑定位置完成绑定，则返回绑定句柄。不会为重复项返回句柄。

返回值：

如果执行成功，则返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.2 Dirpath接口

7.2.1 GCIDirPathPrepare

函数原型：

```
sword   GCIDirPathPrepare (
                    GCIDirPathCtx *dpctx，
                    GCISvcCtx *svchp，
                    GCIError *errhp
                              );
```

功能描述：

在转换或加载任何行之前准备直接路径加载接口。

参数说明：

参数	输入/输出	说明
dpctx	输入	加载对象的直接路径上下文句柄
svchp	输入	服务上下文
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.2 GCIDirPathColArrayEntrySet

函数原型：

```
sword   GCIDirPathColArrayEntrySet(
                GCIDirPathColArray *dpca，
                GCIError *errhp，
                ub4 rownum，
                ub2 colIdx，
                ub1 *cvalp，
                ub4 clen，
                ub1 cflg
                );
```

功能描述：

设置写入文件数据数组上指定位置的数据内容。

参数说明：

参数	输入/输出	说明
dpca	输入	数据数组描述符指针
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
rownum	输入	设置的数据内容在数组中的行号
colIdx	输入	设置的数据内容在数组中的列号
cvalp	输入	设置的数据内容缓冲区指针
clen	输入	数据内容的长度
cflg	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.3 GCIDirPathColArrayToStream

函数原型：

```
sword   GCIDirPathColArrayToStream(
                GCIDirPathColArray *dpca，
                GCIDirPathCtx *dpctx，
                GCIDirPathStream *dpstr，
                GCIError *errhp，
                ub4 rowcnt，
                ub4 rowoff
                );
```

功能描述：

从列数组格式转换为直接路径流格式。

参数说明：

参数	输入/输出	说明
dpca	输入	直接路径列数组句柄
dpctx	输入	正在加载的对象的直接路径上下文句柄
dpstr	输入/输出	直接路径流句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
rowcnt	输入	列数组中的行数
rowoff	输入	列数组中的起始索引

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.4 GCIDirPathLoadStream

函数原型：

```
sword   GCIDirPathLoadStream(
                        GCIDirPathCtx *dpctx，
                        GCIDirPathStream *dpstr，
                        GCIError *errhp
                        );
```

功能描述：

将数据写入到数据库文件。

参数说明：

参数	输入/输出	说明
dpctx	输入	文件操作环境
dpstr	输入	数据流描述符指针
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.5 GCIDirPathDataSave

函数原型：

```
sword   GCIDirPathDataSave(
                    GCIDirPathCtx *dpctx，
                    GCIError *errhp，
                    ub4 action
                            );
```

功能描述：

根据请求的操作，执行数据保存点， 或提交加载的数据并完成直接路径加载操作。

参数说明：

参数	输入/输出	说明
dpca	输入	加载对象的直接路径上下文句柄
errhp	输入/输出	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上

参数	输入/输出	说明
action	输入	操作行为， 取值如下： • GCI_DIRPATH_DATASAVE_SAVEONLY - 仅执行数据保存点 • GCI_DIRPATH_DATASAVE_FINISH - 提交加载的数据并调用直接整理函数

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.6 GCIDirPathColArrayReset

函数原型：

```
sword   GCIDirPathColArrayReset(
                        GCIDirPathColArray *dpca，
                        GCIError *errhp
                              );
```

功能描述：

重置数据数组，以便再次重新设置新的内容。

参数说明：

参数	输入/输出	说明
dpca	输入	数据数组描述符指针
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.7 GCIDirPathStreamReset

函数原型：

```
sword   GCIDirPathStreamReset(
                        GCIDirPathStream *dpstr，
                        GCIError *errhp
                              );
```

功能描述：

重置直接路径流状态。

参数说明：

参数	输入/输出	说明
dpstr	输入	直接路径流句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.8 GCIDirPathFinish

函数原型：

```
sword   GCIDirPathFinish(
                    GCIDirPathCtx *dpctx，
                    GCIError *errhp
                        );
```

功能描述：

完成直接路径加载操作。

参数说明：

参数	输入/输出	说明
dpctx	输入	加载对象的直接路径上下文句柄
errhp	输入/输出	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.9 GCIDirPathAbort()

函数原型：

```
sword   GCIDirPathAbort (
                    GCIDirPathCtx *dpctx，
                    GCIError *errhp
                        );
```

功能描述：

取消已经准备好的直接路径加载操作。

参数说明：

参数	输入/输出	说明
dpctx	输入	直接路径上下文句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.2.10 GCIDirPathFlushRow()

函数原型：

```
sword   GCIDirPathFlushRow (
                    GCIDirPathCtx   *dpctx，
                    GCIError        *errhp
                        );
```

功能描述：

从服务器刷新部分加载的行。

参数说明：

参数	输入/输出	说明
dpctx	输入	加载对象的直接路径上下文句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3 大对象接口

7.3.1 GCILobAppend

函数原型：

```
sword    GCILobAppend(
                        GCISvcCtx       *svchp,
                        GCIError        *errhp,
                        GCILobLocator   *dst_locp,
                        GCILobLocator   *src_locp
                    );
```

功能描述：

将大对象值追加到指定的另一个大对象的末尾。数据将从源复制到目标的末尾。源和目标大对象必须存在。

扩展目标大对象以容纳新写入的数据。将目标大对象扩展到允许的最大长度（4TB）或尝试从空的大对象复制是错误的。

源和目标大对象定位器的类型必须相同（即，它们必须都是 BLOB 或都是 CLOB）。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
dst_locp	输入/输出	唯一引用目标大对象的内部大对象定位器。此定位器必须是从 svchp 指定的服务器获取的定位器。
src_locp	输入	唯一引用源大对象的内部大对象定位器。此定位器必须是从 svchp 指定的服务器获取的定位器。

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.2 GCILobCharSetForm

函数原型：

```
sword    GCILobCharSetForm (
                        GCIEnv                  *envhp,
                        GCIError                *errhp,
                        const GCILobLocator     *locp,
                        ub1                     *csfrm
                                );
```

功能描述：

获取大对象定位器的字符集形式（如果有）。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	要获取其字符集形式的大对象定位器。
csfrm	输出	输入大对象定位器的字符集形式。如果输入定位器 locp 用于 BLOB，则 csfrm 设置为 0，因为BLOB没有字符集的概念。调用方必须为 csfrm （ub1） 分配空间。 csfrm 参数有两个可能的非零值： • SQLCS_IMPLICIT - 数据库字符集 ID，默认值 • SQLCS_NCHAR - NCHAR 字符集 ID

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.3 GCILobCharSetId

函数原型：

```
sword    GCILobCharSetForm (
                        GCIEnv                  *envhp,
                        GCIError                *errhp,
                        const GCILobLocator  *locp,
                        ub2                     *csid
                                );
```

功能描述：

获取大对象定位器的数据库字符集 ID（如果有）。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	要获取其字符集 ID 的大对象定位器。
csid	输出	输入大对象定位器的数据库字符集 ID。如果输入定位器用于 BLOB，则 csid 设置为 0，因为 BLOB 没有字符集的概念。调用方必须为 csid (ub2) 分配空间。

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.4 GCILobClose

函数原型：

```
sword GCILobClose (
            GCISvcCtx *svchp,
            GCIError *errhp,
            GCILobLocator *locp
                    );
```

功能描述：

关闭大对象句柄。

参数说明：

参数	输入/输出	说明
svchp	输入	指定打开连接的上下文， 在此之前， 上下文必须已经被关联到了连接句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
locp	输入	待关闭的大对象字段描述符句柄

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.5 GCILobCopy

函数原型：

```
sword GCILobCopy (
                    GCISvcCtx       *svchp,
                    GCIError        *errhp,
                    GCILobLocator   *dst_locp,
                    GCILobLocator   *src_locp,
                    ub4             amount,
                    ub4             dst_offset,
                    ub4             src_offset
                                );
```

功能描述：

将大对象值的全部或部分复制到另一个大对象值中，数据将从源复制到目标。源 （src_locp ） 和目标（dst_locp）大对象必须存在。推荐使用 GCILobCopy2。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
dst_locp	输入/输出	唯一引用目标大对象的内部大对象定位器。此定位器必须是从 svchp 指定的服务器获取的定位器。
src_locp	输入	唯一引用源大对象的内部大对象定位器。此定位器必须是从 svchp 指定的服务器获取的定位器。
amount	输入	要从源大对象复制到目标大对象的 CLOB 的字符数或 BLOB 的字节数。
dst_offset	输入	这是目标大对象的绝对偏移量。对于 CLOB ，它是从 CLOB 开头开始写入的字符数。对于 BLOB，它是从 BLOB 开头开始写入的字节数，偏移量从 1 开始。
src_offset	输入	这是源大对象的绝对偏移量。对于 BLOB，它是来自 BLOB 的字符数。对于BLOB，它是字节数，从 1 开始。

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.6 GCILobCopy2

函数原型：

```
sword GCILobCopy2 (
                    GCISvcCtx       *svchp,
                    GCIError        *errhp,
                    GCILobLocator   *dst_locp,
                    GCILobLocator   *src_locp,
                    ub4             amount,
                    ub4             dst_offset,
                    ub4             src_offset
                                );
```

功能描述：

将大对象值的全部或部分复制到另一个大对象值中，数据将从源复制到目标。源(src_locp)和目标(dst_locp)大对象必须存在。暂不支持大于4G的LOB对象。

如果数据存在于目标的起始位置，则源数据将覆盖该数据。如果目标的起始位置超出当前数据的末尾，则零字节填充符(对于BLOB)或空格(对于CLOB)将从当前数据的末尾写入源中新写入的数据的开头。如果目标大对象超出目标大对象的当前长度，则会扩展目标大对象以容纳新写入的数据。将目标大对象扩展到允许的最大长度（即 4 TB）之外或尝试从 NULL 大对象复制是错误的。

源和目标大对象定位器必须属于同一类型（即，它们必须是 BLOB 或都是 CLOB）。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
dst_locp	输入/输出	唯一引用目标大对象的内部大对象定位器。此定位器必须是从 svchp 指定的服务器获取的定位器。
src_locp	输入	唯一引用源大对象的内部大对象定位器。此定位器必须是从 svchp 指定的服务器获取的定位器。
amount	输入	要从源大对象复制到目标大对象的 CLOB 的字符数或 BLOB 的字节数。
dst_offset	输入	这是目标大对象的绝对偏移量。对于 CLOB，它是从 CLOB 开头开始写入的字符数。对于 BLOB，它是从 BLOB 开头开始写入的字节数，偏移量从 1 开始。
src_offset	输入	这是源大对象的绝对偏移量。对于 BLOB，它是来自 BLOB 的字符数。对于BLOB，它是字节数，从 1 开始。

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.7 GCILobCreateTemporary

函数原型：

```
sword GCILobCreateTemporary (
                    GCISvcCtx       *svchp,
                    GCIError        *errhp,
                    GCILobLocator   *locp,
                    ub2             csid,
                    ub1             csfrm,
                    ub1             lobtype,
                    boolean         cache,
                    GCIDuration     duration
                                    );
```

功能描述：

创建一个临时 LOB 对象。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入/输出	指向临时 LOB 对象
csid	输入	保留参数，目前不使用（仅用于和 Oracle 保持兼容）
csfrm	输入	保留参数，目前不使用（仅用于和 Oracle 保持兼容）
lobtype	输入	大对象类型，可能的值： ● GCI_TEMP_BLOB，临时 BLOB ● GCI_TEMP_CLOB，临时 CLOB
cache	输入	保留参数，目前不使用（仅用于和 Oracle 保持兼容）
duration	输入	保留参数，目前不使用（仅用于和 Oracle 保持兼容）

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.8 GCILobErase

函数原型：

```
sword GCILobErase (
                    GCISvcCtx       *svchp,
                    GCIError        *errhp,
                    GCILobLocator   *locp,
                    ub4             *amount,
                    ub4             offset
                                    );
```

功能描述：

从指定的偏移量开始删除内部 LOB 数据的指定部分。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入/输出	被删除数据的 LOB 对象
amount	输入/输出	要擦除的 CLOB 的字符数或 BLOB 的字节数。在 IN 上，该值表示要擦除的字符数或字节数。在 OUT 上，该值标识擦除的实际字符数或字节数。
offset	输入	从擦除数据的 LOB 值的开头开始，CLOB 的字符绝对偏移量，或 BLOB 的字节绝对偏移量。从 1 开始。

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.9 GCILobErase2

函数原型：

```
sword GCILobErase2 (
                    GCISvcCtx       *svchp,
                    GCIError        *errhp,
                    GCILobLocator   *locp,
                    gbsub8          *amount,
                    gbsub8          offset
                                        );
```

功能描述：

从指定的偏移量开始擦除内部 LOB 数据的指定部分，暂不支持大于 4G 的 LOB 对象。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入/输出	被删除数据的 LOB 对象
amount	输入/输出	删除的字节数
offset	输入	擦除数据的字节绝对偏移量，从 1 开始

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.10 GCILobFreeTemporary

函数原型：

```
sword GCILobFreeTemporary (
                    GCISvcCtx       *svchp,
                    GCIError        *errhp,
                    GCILobLocator   *locp
                                        );
```

功能描述：

释放临时大对象。如果在 locp 参数中传递的 LOB 定位器未指向临时 LOB，则此函数将返回错误。

参数说明：

参数	输入/输出	说明
svchp	输入/输出	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入/输出	唯一引用要释放的 LOB 的定位器

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.11 GCILobGetChunkSize

函数原型：

```
sword   GCILobGetChunkSize (
                GCISvcCtx       *svchp，
                GCIError        *errhp，
                GCILobLocator   *locp，
                Ub4              *chunk_size
                        );
```

功能描述：

获取大对象的 chunk 大小。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	要获取其可用块大小的 LOB 对象
chunk_size	输出	存储内部 LOB 值的块空间量

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.12 GCILobGetLength

函数原型：

```
sword   GCILobGetLength(
                GCISvcCtx       *svchp，
                GCIError        *errhp，
                GCILobLocator   *locp，
                Ub4              *lenp
                        );
```

功能描述：

返回大对象字段的大小，按字节计算。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	存储大字段描述符指针
lenp	输出	返回的大字段长度， 按字节计算

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.13 GCILobGetLength2

函数原型：

```
sword    GCILobGetLength2(
                    GCISvcCtx          *svchp,
                    GCIError           *errhp,
                    GCILobLocator   *locp,
                    gbsub8             *lenp
                          );
```

功能描述：

返回大对象字段的长度。获取 LOB 的长度。如果 LOB 为 NULL，则长度未定义。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄的指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	唯一引用 LOB 的 LOB 定位器。此定位器必须是从 svchp 指定的服务器获取的定位器
lenp	输出	返回的大字段长度，对于CLOB，返回字符数;对于BLOB ，返回字节数

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.14 GCILobIsEqual

函数原型：

```
sword GCILobIsEqual (
                    GCIEnv                      *envhp,
                    const GCILobLocator         *x,
                    const GCILobLocator         *y,
                    boolean                     *is_equal
                          );
```

功能描述：

判断给定的 LOB 句柄是否相等。此函数认为两个 NULL 定位符不相等。

参数说明：

参数	输入/输出	说明
envhp	输入	环境句柄
x	输入	待比较的大对象定位符
y	输入	待比较的大对象定位符
is_equal	输出	如果大对象定位符相等，则为 TRUE;如果不是，则为 FALSE。

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.15 GCILobIsOpen

函数原型：

```
sword GCILobIsOpen (
                GCISvcCtx *svchp,
                GCIError *errhp,
                GCILobLocator *locp,
                boolean *flag
                        );
```

功能描述：

打开大对象。

参数说明：

参数	输入/输出	说明
svchp	输入	指定打开连接的上下文，在此之前，上下文必须已经被关联到了连接句柄
errhp	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	待打开的大字段描述符指针
flag	输入/输出	打开是否成功

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

注：调用该接口进行连接后，须在适当的位置调用GCILobClose关闭该大对象。

7.3.16 GCILobIsTemporary

函数原型：

```
sword GCILobIsTemporary (
                GCIEnv          *envhp,
                GCIError        *errhp,
                GCILobLocator   *locp,
                boolean         *is_temporary
                        );
```

功能描述：

检测是否为临时 LOB。

参数说明：

参数	输入/输出	说明
envhp	输入	环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	待检测的大字段描述符指针
is_temporary	输出	如果 LOB 定位器指向临时大对象，则返回 TRUE;如果不是，则为 FALSE

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.17 GCILobLocatorIsInit

函数原型：

```
sword GCILobLocatorIsInit (
                        GCIEnv *envhp,
                        GCIError *errhp,
                        const GCILobLocator *locp,
                        boolean *is_initialized
                            );
```

功能描述：

判断大对象句柄是否初始化过。

参数说明：

参数	输入/输出	说明
envhp	输入	操作所在的环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
locp	输入	大对象句柄
is_initialized	输出	初始化是否成功，大对象句柄内容已经获取到了正确的大对象数据描述信息后，返回是初始化成功

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.18 GCILobOpen

函数原型：

```
sword    GCILobOpen (
                    GCISvcCtx          *svchp,
                    GCIError           *errhp,
                    GCILobLocator      *locp,
                    ub1                 mode
                        );
```

功能描述：

在指示的模式下打开内部或外部 LOB。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入/输出	要打开的 LOB。定位器可以引用内部或外部 LOB。
mode	输入	打开 LOB 的模式，有效模式如下： ● GCI_LOB_READONLY ● GCI_LOB_READWRITE

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.19 GCILobRead

函数原型：

```
sword   GCILobRead(
                    GCISvcCtx *svchp，

                    GCIError *errhp，
                    GCILobLocator *locp，
                    Ub4 *amtp，

                    Ub4 offset，
                    Dvoid *bufp，
                    Ub4 bufl
                    Dvoid *ctxp，
                    Sb4 (*cbfp)(dvoid *ctxp ， CONST dvoid *bufp ， ub4 len ， ub1 piece)，
                    Ub2 csid，
                    Ub1 csfrm
                            );
```

功能描述：

连续向前读取某个大字段中指定长度的内容。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄指针
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
locp	输入	存储大字段描述符指针
amtp	输入/输出	调用函数时，该参数表示想读取的字节数，函数执行完以后，该参数为实际读到的字节数
offset	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
bufp	输入	存放读到数据的缓冲区指针
bufl	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
ctxp	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
cbfp	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
csid	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)
csfrm	输入	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功，在还有数据未读取完的情况下，返回 GCI_NEED_DATA，如果数据已经都 读完，则返回 GCI_SUCCESS，执行失败返回 GCI_ERROR。

7.3.20 GCILobRead2

函数原型：

```
sword    GCILobRead2(
                    GCISvcCtx                   *svchp,
                    GCIError                    *errhp,
                    GCILobLocator               *locp,
                    gbsub8                      *byte_amtp,
                    gbsub8                      *char_amtp,
                    gbsub8                      offset,
                    void                        *bufp,
                    gbsub8                      bufl,
                    ub1                         piece,
                    void                        *ctxp,
                    GCICallbackLobRead2         (cbfp)
                        (void               *ctxp,
                        const void          *bufp,
                        gbsub8              lenp,
                        ub1                 piecep
                        void                **changed_bufpp,
                        gbsub8              *changed_lenp
                        ),
                    ub2                         csid,
                    ub1                         csfrm
                            );
```

功能描述：

连续向前读取某个大对象中指定长度的内容。暂不支持大于4G的LOB对象读取。

参数说明：

参数	输入/输出	说明
svchp	输入/输出	上下文句柄指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
locp	输入	存储大字段描述符指针
byte_amtp	输入/输出	调用函数时，该参数表示想读取的字节数，函数执行完以后，该参数为实际读到的字节数，对于 CLOB，仅在 char_amtp 为零时使用
char_amtp	输入/输出	对于 CLOB,该参数表示想读取的字节数，函数执行完以后，该参数为实际读到的字节数
offset	输入	输入时，这是从 LOB 值开始的绝对偏移。它是字节数。第一位是 1。 如果使用流式传输（通过轮询或回调），请在第一个调用中指定偏移量；在随后的轮询调用中，将忽略 offset 参数。使用回调时，没有 offset 参数
bufp	输入/输出	存放读到数据的缓冲区指针
bufl	输入	缓冲区的长度,以字节为单位指定
piece	输入	GCI_ONE_PIECE-呼叫永远不会进行轮询。如果指示的数量大于缓冲区长度，则缓冲区将尽可能填充。要进行轮询，请 GCI_FIRST_PIECE 在第一次和 GCI_NEXT_PIECE 以后的通话中传递。GCI_FIRST_PIECE 应该在使用回调时传递

参数	输入/输出	说明
ctxp	输入	回调函数的上下文指针，可以为 NULL
cbfp	输入	可以注册为每个片段调用的回调。如果是 NULL，GCI_NEED_DATA 则为每块返回。 回调函数必须返回 GCI_CONTINUE 才能继续读取。如果返回任何其他错误代码，则将终止 LOB 读取。 回调采用以下参数： ● ctxp（输入） 回调函数的上下文。可以 NULL。 ● bufp (输入/输出) 片段的缓冲区指针。 ● lenp（输入） bufp 指向的当前片段的长度（以字节为单位） ● piecep (输入) 可能的值：GCI_FIRST_PIECE，GCI_NEXT_PIECE 或 GCI_LAST_PIECE。 ● changed_bufpp(输出) 保留参数，目前不使用（仅用于和 Oracle 保持兼容） ● changed_lenp(输出) 保留参数，目前不使用（仅用于和 Oracle 保持兼容）
csid	输入	保留参数，目前不使用(仅用于和 Oracle 保持兼容)
csfrm	输入	保留参数，目前不使用(仅用于和 Oracle 保持兼容)

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.21 GCILobTrim

函数原型：

```
sword GCILobTrim(
                    GCISvcCtx           *svchp,
                    GCIError             *errhp,
                    GCILobLocator       *locp,
                    ub4                 newlen
                );
```

功能描述：

大对象数据进行截断。

参数说明：

参数	输入/输出	说明
svchp	输入/输出	该接口内会自动生成一个当前环境句柄下的上下文句柄， 并将地址保存到该指针
errhp	输入	错误信息句柄， 该接口调用失败时将错误码及错误信息写入该句柄
locp	输入/输出	存储大对象描述符句柄
newlen	输出	大对象数据截断后的长度，一般该长度需要小于或等于现有大对象长度

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.22 GCILobTrim2

函数原型：

```
sword GCILobTrim2(
                    GCISvcCtx           *svchp,
                    GCIError            *errhp,
                    GCILobLocator       *locp,
                    gbsub8              newlen
                        );
```

功能描述：

将 LOB 值截断为较短的长度。如果 newlen 大于当前 LOB 长度，则该函数将返回错误。

参数说明：

参数	输入/输出	说明
svchp	输入/输出	该接口内会自动生成一个当前环境句柄下的上下文句柄，并将地址保存到该指针
errhp	输入	错误信息句柄，该接口调用失败时将错误码及错误信息写入该句柄
locp	输入/输出	存储大对象描述符句柄
newlen	输出	大对象数据截断后的长度，一般该长度需要小于或等于现有大对象长度。对于CLOB，它是字符数；对于BLOB，它是字节数

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.3.23 GCILobWrite

函数原型：

```
sword GCILobWrite(
                GCISvcCtx *svchp,
                GCIError *errhp,
                GCILobLocator *locp，
                ub4 *amtp,
                ub4 *offset,
                dvoid *bufp,
                ub4 buflen,
                ub1 piece,
                dvoid *ctxp,
                sb4 (*cbfp)(void *ctxp,void *bufp,ub4 *lenp,ub1 *piecep),
                ub2 csid,
                ub1 csfrm
                    );
```

功能描述：

连续的写入内容到一个大字段的存储描述符中。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄指针
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会保存在错误句柄中
locp	输入	存储大字段描述符指针
amtp	输入/输出	该参数为输入输出参数， 当调用函数时， 该参数表明想写入的字节数，当函数执行完成后， GCI会回填实际写入的字节数
offset	输入	从大字段开始位置到当前位置的偏移量， 以字节为单位
bufp	输入	存放要写的数据缓冲区指针
buflen	输入	bufp参数指向的缓冲区大小
piece	输入	保留参数， 目前不适用(仅用于和Oracle调用保持兼容)
ctxp	输入	保留参数， 目前不适用(仅用于和Oracle调用保持兼容)
cbfp	输入	保留参数， 目前不适用(仅用于和Oracle调用保持兼容)
csid	输入	保留参数， 目前不适用(仅用于和Oracle调用保持兼容)
csfrm	输入	保留参数， 目前不适用(仅用于和Oracle调用保持兼容)

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

注：可以调用 GCIStmtFetch 来移动游标，以便对其他行的大字段进行写入操作。

7.3.24 GCILobWrite2

函数原型：

```
sword GCILobWrite2(
                    GCISvcCtx             *svchp,
                    GCIError              *errhp,
                    GCILobLocator         *locp,
                    gbsub8                *byte_amtp,
                    gbsub8                *char_amtp,
                    gbsub8                offset,
                    void                  *bufp,
                    gbsub8                buflen,
                    ub1                   piece,
                    void                  *ctxp,
                    GCICallbackLobWrite2  (cbfp)
                      (void       *ctxp,
                      void        *bufp,
                      oraub8      *lenp,
                      ub1         *piecep
                      void        **changed_bufpp,
                      oraub8      *changed_lenp
                      ),
                    ub2                   csid,
                    ub1                   csfrm
                      );
```

功能描述：

写入内容到一个大对象中，暂不支持大于4G的LOB对象。

参数说明：

参数	输入/输出	说明
svchp	输入/输出	上下文句柄指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会保存在错误句柄中
locp	输入/输出	存储大对象描述符指针
byte_amtp	输入/输出	该参数为输入输出参数，当调用函数时，该参数表明想写入的字节数，当函数执行完成后，GCI 会回填实际写入的字节数.对于 CLOB，仅在 char_amtp 为零时使用
char_amtp	输入/输出	对于CLOB,该参数表示想读取的字节数，函数执行完以后，该参数为实际读到的字节数
offset	输入	从大对象开始位置到当前位置的偏移量，以字节为单位
bufp	输入	存放要写的数据缓冲区指针
buflen	输入	bufp参数指向的缓冲区大小
piece	输入	正在写入缓冲区的哪一部分。此参数的默认值为 GCI_ONE_PIECE，表示将缓冲区写入单个块中。 对于分段或回调模式可以使 GCI_FIRST_PIECE，GCI_NEXT_PIECE，和 GCI_LAST_PIECE
ctxp	输入	回调函数的参数，可以 NULL
cbfp	输入	可以注册的回调函数，可以在逐段写入 LOB 数据。如果是 NULL，则使用标准轮询方法。 回调函数必须返回 GCI_CONTINUE 才能继续读取。如果返回任何其他错误代码，则将终止 LOB 写入。 回调采用以下参数： ● ctxp（输入） 回调函数的参数。可以 NULL。 ● bufp (输入/输出) 片段的缓冲区指针。 ● lenp (输入/输出) 缓冲区（输入）中数据的长度（以字节为单位）。 ● piecep(输出) 哪一块：GCI_NEXT_PIECE 或 GCI_LAST_PIECE。 ● changed_bufpp(输出) 保留参数，目前不适用(仅用于和 Oracle 调用保持兼容) ● changed_lenp(输出) 保留参数，目前不适用(仅用于和 Oracle 调用保持兼容)
csid	输入	保留参数，目前不适用(仅用于和Oracle调用保持兼容)
csfrm	输入	保留参数，目前不适用(仅用于和Oracle调用保持兼容)

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.25 GCILobWriteAppend

函数原型：

```
sword GCILobWriteAppend (
                    GCISvcCtx               *svchp,
                    GCIError                *errhp,
                    GCILobLocator           *locp,
                    ub4                     *amtp,
                    void                    *bufp,
                    ub4                     buflen,
                    ub1                     piece,
                    void                    *ctxp,
                    GCICallbackLobWrite     (cbfp)
                      (
                      void      *ctxp,
                      void      *bufp,
                      ub4       *lenp,
                      ub1       *piecep
                      ),
                    ub2                     csid,
                    ub1                     csfrm
                      );
```

功能描述：

从 LOB 的末尾开始写入数据。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会保存在错误句柄中
locp	输入/输出	内部 LOB 对象
amtp	输入/输出	该参数为输入输出参数，当调用函数时，该参数表明想写入的字节数，当函数执行完成后，GCI 会回填实际写入的字节数
bufp	输入	指向从中写入片段的缓冲区的指针
buflen	输入	bufp参数指向的缓冲区大小，（以字节为单位）
piece	输入	正在写入缓冲区的哪一部分。此参数的默认值为 GCI_ONE_PIECE，表示将缓冲区写入单个块中。对于分段或回调模式 GCI_FIRST_PIECE，以下其他值也是可能的：GCI_NEXT_PIECE 和 GCI_LAST_PIECE
ctxp	输入	回调函数的上下文，可以 NULL
cbfp	输入	可以注册的回调，可以在逐段写入中为每段调用。如果是 NULL，则使用标准轮询方法。回调函数必须返回 GCI_CONTINUE 才能继续写操作。如果返回任何其他错误代码，则 LOB 写入将终止。回调采用以下参数： ● ctxp（输入） 回调函数的上下文，可以 NULL。 ● bufp (输入/输出) 片段的缓冲区指针。 ● lenp (输入/输出) 缓冲区（输入）中数据的长度（以字节为单位），以及 bufp（输出）中当前数据的长度（以字节为单位） ● piecep(输出) 哪一块，可能的值：GCI_NEXT_PIECE 或 GCI_LAST_PIECE

csid	输入	保留参数，目前不适用(仅用于和Oracle调用保持兼容)
csfrm	输入	保留参数，目前不适用(仅用于和Oracle调用保持兼容)

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.3.26 GCILobWriteAppend2

函数原型：

```
sword GCILobWriteAppend2 (
                    GCISvcCtx               *svchp,
                    GCIError                *errhp,
                    GCILobLocator           *locp,
                    gbsub8                  *byte_amtp,
                    gbsub8                  *char_amtp,
                    void                    *bufp,
                    gbsub8                  buflen,
                    ub1                     piece,
                    void                    *ctxp,
                    GCICallbackLobWrite2    (cbfp)
                      (
                      void      *ctxp,
                      void      *bufp,
                      ub4       *lenp,
                      ub1       *piecep
                      void      **changed_bufpp,
                      oraub8    *changed_lenp
                      ),
                    ub2                     csid,
                    ub1                     csfrm
                       );
```

功能描述：

从 LOB 的末尾开始写入数据。

参数说明：

参数	输入/输出	说明
svchp	输入	上下文句柄指针
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会保存在错误句柄中
locp	输入/输出	内部 LOB 对象
byte_amtp	输入/输出	该参数为输入输出参数，当调用函数时，该参数表明想写入的字节数，当函数执行完成后，GCI 会回填实际写入的字节数.对于 CLOB，仅在 char_amtp 为零时使用
char_amtp	输入/输出	对于 CLOB,该参数表示想读取的字节数，函数执行完以后，该参数为实际读到的字节数
bufp	输入	指向从中写入片段的缓冲区的指针
buflen	输入	缓冲区中数据的长度（以字节为单位）。请注意，此参数假定为 8 位字节。如果您的操作系统使用较长的字节，则必须相应地调整 buflen 的值
piece	输入	正在写入缓冲区的哪一部分。此参数的默认值为 GCI_ONE_PIECE，表示将缓冲区写入单个块中。对于分段或回

		调模式，以下其他值也是可能的：GCI_FIRST_PIECE、GCI_NEXT_PIECE 和 GCI_LAST_PIECE
ctxp	输入	回调函数的上下文，可以 NULL
参数	**输入/输出**	**说明**
cbfp	输入	可以注册的回调，可以在逐段写入中为每段调用。如果是 NULL，则使用标准轮询方法。回调函数必须返回 GCI_CONTINUE 才能继续写操作。如果返回任何其他错误代码，则 LOB 写入将终止。回调采用以下参数： ● ctxp（输入） 回调函数的上下文，可以 NULL。 ● bufp (输入/输出) 片段的缓冲区指针。 ● lenp (输入/输出) 缓冲区（输入）中数据的长度（以字节为单位），以及 bufp（输出）中当前片段的长度（以字节为单位） ● piecep(输出) 哪一块，可能的值：GCI_NEXT_PIECE 或 GCI_LAST_PIECE ● changed_bufpp(输出) 保留参数， 目前不适用(仅用于和 Oracle 调用保持兼容) ● changed_lenp(输出) 保留参数， 目前不适用(仅用于和 Oracle 调用保持兼容)
csid	输入	保留参数， 目前不适用(仅用于和 Oracle 调用保持兼容)
csfrm	输入	保留参数， 目前不适用(仅用于和 Oracle 调用保持兼容)

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.4 时间类型接口

7.4.1 GCIDateAddDays

函数原型：

```
sword GCIDateAddDays (
                    GCIError *err,
                    const GCIDate *date,
                    sb4 num_days,
                    GCIDate *result
                          );
```

功能描述：

对给定的日期值，增加或减去相应的天数。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date	输入	日期值的被加数或被减数
num_days	输入	日期值的加数或减数
result	输出	计算结果

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.2 GCIDateAddMonths

函数原型：

```
sword GCIDateAddMonths (
                        GCIError *err,
                        const GCIDate *date,
                        sb4 num_months,
                        GCIDate *result
                            );
```

功能描述：

对给定的日期值，增加或减去相应的月数。注意： 如果输入日期值为当月最后一天，则进 行加减运算后得出的结果值也调整为对应月份的最后一天。如2021/2/28 加1个月得到 2021/3/31 ，2021/11/30 减3个月得到 2021/8/31。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date	输入	日期值的被加数或被减数
num_months	输入	日期值的加数或减数
result	输出	计算结果

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.3 GCIDateAssign

函数原型：

```
sword GCIDateAssign (
                        GCIError *err,
                        const GCIDate *from,
                        GCIDate *to
                            );
```

功能描述：

拷贝from值至to。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
from	输入	要拷贝的数据源
to	输出	要拷贝的数据目标

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.4 GCIDateCheck

函数原型：

```
sword GCIDateCheck(
                    GCIError *err,
                    const GCIDate * date,
                    uword   *valid
                      );
```

功能描述：

检测指定的日期值是否有效。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
date	输入	要检查的日期值
valid	输出	对于合法日期值，返回0。对于非法日期值，返回错误码小节中所示的错误位进行或运算的结果 例如： 如果输入的日期为13/0/1900, 格式串为month/day/year，会返回如下错误GCI_DATE_INVALID_MONTH | GCI_DATE_INVALID_DAY

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.5 GCIDateCompare

函数原型：

```
sword GCIDateCompare(
                    GCIError *err,
                    const GCIDate * date1,
                    const GCIDate * date2,
                    sword   *result
                      );
```

功能描述：

比较两个日期值大小。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
date1,date2	输入	两个要比较的日期值
result	输出	比较结果如下： ● date1 < date2，输出 - 1 ● date1 = date2，输出 0 ● date1 > date2，输出 1

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.6 GCIDateDaysBetween

函数原型：

```
sword GCIDateDaysBetween (
                        GCIError *err,
                        const GCIDate * date1,
                        const GCIDate * date2,
                        sb4   *num_days
                              );
```

功能描述：

获取两个日期之间的天数(可正可负)。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
date1,date2	输入	日期值
num_days	输出	date1与date2之间间隔天数(date1-date2)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.7 GCIDateFromText

函数原型：

```
sword GCIDateFromText (
                        GCIError *err,
                        const GCIText *date_str,
                        ub4 d_str_length,
                        const GCIText *fmt,
                        ub1 fmt_length,
                        const GCIText *lang_name,
                        ub4 lang_length,
                        GCIDate *date
                              );
```

功能描述：

将包含日期的Text类型字符串date_str按照fmt格式转换成date类型。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date_str	输入	要转换成日期值的字符串
d_str_length	输入	字符串长度
fmt	输入	转换格式， 支持的格式与数据库服务中TO_CHAR和TO_DATE函数的format_string参数一致，详见《GBase 8s SQL 指南:语法》
fmt_length	输入	格式串长度
lang_name	输入	兼容保留参数， 一般填写NULL
lang_length	输入	兼容保留参数， 一般填写0
date	输出	转换后的日期

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.8 GCIDateLastDay

函数原型：

```
sword GCIDateLastDay (
                    GCIError *err,
                    const GCIDate *date,
                    GCIDate *last_day
                        );
```

功能描述：

获取指定月份的最后一天的日期。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date	输入	指定的日期值
last_day	输出	date中指定的月份中最后一天的日期值

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.9 GCIDateNextDay

函数原型：

```
sword GCIDateNextDay (
                    GCIError *err,
                    const GCIDate *date,
                    const GCIText *day,
                    ub4 day_length,
                    GCIDate *next_day
                        );
```

功能描述：

获取指定日期后的第一个周day的日期值。如获取2021年7月1日（周4） 后的第一个周3的日 期值， 即2021年7月7日。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date	输入	指定的日期值
day	输入	指定要返回的下一个周期几
day_length	输入	day的字符串长度值
next_day	输出	返回的日期值

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.10 GCIDateSysDate

函数原型：

```
sword GCIDateSysDate (
                    GCIError *err,
                    GCIDate *sys_date
                        );
```

功能描述：

获取客户端的当前时期和时间。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
sys_date	输出	获取到的客户端系统日期值

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.11 GCIDateTimeAssign

函数原型：

```
sword GCIDateTimeAssign (
                        void *hndl,
                        GCIError *err,
                        const GCIDateTime *from,
                        GCIDateTime *to
                            );
```

功能描述：

两个日期时间变量的赋值。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
from	输入	复制操作数据源
to	输出	复制操作目的地

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.12 GCIDateTimeCheck

函数原型：

```
sword GCIDateTimeCheck (
                        void *hndl,
                        GCIError *err,
                        const GCIDateTime *date,
                        ub4 *valid
                            );
```

功能描述：

检测给定的日期时间值是否有效。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date	输入	要检查的日期时间值
valid	输出	对于合法日期值，返回 0。对于非法日期值，返回错误码小节所示的错误位进行或运算的结果。 例如： 如果输入的日期为2/0/1900 25:61:10, 格式串为 month/day/year hours:minutes:seconds，会返回如下错误 GCI_DT_INVALID_DAY | GCI_DT_INVALID_HOUR |GCI_DT_INVALID_MINUTE

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.13 GCIDateTimeCompare

函数原型：

```
sword GCIDateTimeCompare (
                        void *hndl,
                        GCIError *err,
                        const GCIDateTime *date1,
                        const GCIDateTime *date2,
                        sword *result
                            );
```

功能描述：

两个日期时间变量的比较。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
date1，date2	输入	待比较的两个日期时间值
result	输出	比较结果如下： ● date1 < date2，输出 - 1 ● date1 = date2，输出 0 ● date1 > date2，输出 1

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.14 GCIDateTimeConstruct

函数原型：

```
sword GCIDateTimeConstruct (
                        void *hndl,
                        GCIError *err,
                        GCIDateTime *datetime,
                        sb2 year,
                        ub1 month,
                        ub1 day,
                        ub1 hour,
                        ub1 min,
                        ub1 sec,
                        ub4 fsec,
                        GCIText *timezone,
                        size_t timezone_length
                              );
```

功能描述：

设置GCIDateTime日期时间对象的数据。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
datetime	输入/输出	日期时间
year	输入	单位： 年
month	输入	单位： 月
day	输入	单位： 日
hour	输入	单位： 时
min	输入	单位： 分
sec	输入	单位： 秒
fsec	输入	总共为5位
timezone	输出	保留参数， 目前不使用(仅用于和Oracle保持兼容)
timezone_length	输出	保留参数， 目前不使用(仅用于和Oracle保持兼容)

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.15 GCIDateTimeGetDate

函数原型：

```
sword GCIDateTimeGetDate(
                        dvoid *hndl,
                        GCIError *err,
                        CONST GCIDateTime *date,
                        sb2 *yr,
                        ub1 *mnth,
                        ub1 *dy
                            );
```

功能描述：

获取GCIDateTime中的日期，包括年、月、日。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
datetime	输入	日期时间
yr	输出	单位： 年
mnth	输出	单位： 月
dy	输出	单位： 日

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.16 GCIDateTimeGetTime

函数原型：

```
sword GCIDateTimeGetTime (
                        void *hndl,
                        GCIError *err,
                        GCIDateTime *datetime,
                        ub1 *hour,
                        ub1 *min,
                        ub1 *sec,
                        ub4 *fsec
                            );
```

功能描述：

获取GCIDateTime对象中的时间，包含亚秒。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
datetime	输入	日期时间
hour	输出	单位： 时

参数	输入/输出	说明
min	输出	单位： 分
sec	输出	单位： 秒
fsec	输出	总共为5位

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.17 GCIDateTimeGetTimeZoneOffset

函数原型：

```
sword GCIDateTimeGetTimeZoneOffset (
                        void                *hndl,
                        GCIError            *errhp,
                        const GCIDateTime   *datetime,
                        sb1                 *hour,
                        sb1                 *min
                        );
```

功能描述：

获取日期时间值的时区（小时、分钟）部分。如果给定的日期时间不包含时间信息，则此函数返回错误。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
datetime	输入	指向GCIDateTime 描述符的指针
hour	输出	检索到的时区小时值
min	输出	检索到的时区分钟值

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.4.18 GCIDateTimeIntervalAdd

函数原型：

```
sword GCIDateTimeIntervalAdd (
                        void                *hndl,
                        GCIError            *errhp,
                        GCIDateTime         *datetime,
                        GCIInterval         *interval,
                        GCIDateTime         *outdatetime
                        );
```

功能描述：

向日期时间添加间隔以生成新的日期时间。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
datetime	输入	指向输入日期时间的指针
interval	输入	指向输入间隔的指针
outdatetime	输出	指向输出日期时间的指针。输出日期时间与输入日期时间的类型相同。

返回值：

如果执行成功，返回GCI_SUCCESS；如果错误是空指针，返回GCI_INVALID_HANDLE；如果生成的日期早于 -4713 年 1 月 1 日或晚于 9999 年 12 月 31 日，返回GCI_ERROR。

7.4.19 GCIDateTimeIntervalSub

函数原型：

```
sword GCIDateTimeIntervalSub (
                        void            *hndl,
                        GCIError        *errhp,
                        GCIDateTime     *datetime,
                        GCIInterval     *interval,
                        GCIDateTime     *outdatetime
                        );
```

功能描述：

从日期时间中减去间隔并将结果存储在日期时间中。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
datetime	输入	指向输入日期时间的指针
interval	输入	指向输入间隔的指针
outdatetime	输出	指向输出日期时间的指针。输出日期时间与输入日期时间的类型相同。

返回值：

如果执行成功，返回GCI_SUCCESS；如果错误是空指针，返回GCI_INVALID_HANDLE；如果生成的日期早于 -4713 年 1 月 1 日或晚于 9999 年 12 月 31 日，返回GCI_ERROR。

7.4.20 GCIDateTimeSubtract

函数原型：

```
sword GCIDateTimeSubtract (
                        void            *hndl,
                        GCIError        *errhp,
                        GCIDateTime     *indate1,
                        GCIDateTime     *indate2,
                        GCIInterval     *inter
                        );
```

功能描述：

将两个日期时间作为输入，并以间隔存储它们的差异。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
indate1	输入	指向减数的指针（要减去的数字）
indate2	输入	指向被减数的指针
inter	输出	指向输出间隔的指针

返回值：

如果执行成功，返回GCI_SUCCESS；如果错误是空指针，返回GCI_INVALID_HANDLE；如果生成的日期早于 -4713 年 1 月 1 日或晚于 9999 年 12 月 31 日，返回GCI_ERROR。

7.4.21 GCIDateToText

函数原型：

```
sword GCIDateToText (
                GCIError *err,
                const GCIDate *date,
                const GCIText *fmt,
                ub1 fmt_length，
                const GCIText *lang_name,
                ub4 lang_length,
                ub4 *buf_size,
                GCIText *buf
                        );
```

功能描述：

将日期date按照fmt格式转换成字符串类型。

参数说明：

参数	输入/输出	说明
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date	输入	要转换成字符串的日期值
fmt	输入	转换格式， 支持的格式与数据库服务中TO_CHAR和TO_DATE函数的format_string参数一致， 详见《GBase 8s SQL 指南:语法》
fmt_length	输入	格式串长度
lang_name	输入	兼容保留参数，一般填写NULL
lang_length	输入	兼容保留参数，一般填写0
buf_size	输入/输出	buf定义的长度(输入) 转换后的字符串长度(输出)
buf	输出	转换后的字符串

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.22 GCIDateTimeToArray

函数原型：

```
sword GCIDateTimeToArray (
                        void *hndl,
                        GCIError *errhp,
                        const GCIDateTime *datetime,
                        const GCIInterval *reftz,
                        ub1 *outarray,
                        ub4 *len
                        ub1 fsprec
                            );
```

功能描述：

将日期时间转换为数组。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
datetime	输入	指向GCIDateTime描述符的指针
reftz	输入	保留参数，目前不使用（仅用于和Oracle保持兼容）
outarray	输出	包含日期的字节数组
len	输出	outarray的长度
fsprec	输入	保留参数，目前不使用（仅用于和Oracle保持兼容）

返回值：如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.23 GCIDateTimeToText

函数原型：

```
sword GCIDateTimeToText (
                        void *hndl,
                        GCIError *err,
                        const GCIDateTime *date,
                        const GCIText *fmt,
                        ub1 fmt_length,
                        ub1 fsprec,
                        const GCIText *lang_name,
                        size_t lang_length,
                        ub4 *buf_size,
                        GCIText *buf
                            );
```

功能描述：

将日期时间date按照fmt格式转换成字符串类型。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
date	输入	要转换成字符串的日期时间值
fmt	输入	转换格式， 支持的格式与数据库服务中TO_CHAR和TO_DATE函数的format_string参数一致， 详见《GBase 8s SQL 指南:语法》
fmt_length	输入	格式串长度
fsprec	输入	指定要保留的小数部分秒的精度
lang_name	输入	兼容保留参数，一般填写NULL
lang_length	输入	兼容保留参数，一般填写0
buf_size	输入/输出	转换前的buf大小(输入) 转换后字符串的长度(输出)
buf	输出	转换后的字符串

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.24 GCIIntervalGetDaySecond

函数原型：

```
Sword GCIIntervalGetDaySecond(
                        dvoid *hndl,
                        GCIError *err,
                        sb4 *dy,
                        sb4 *hr,
                        sb4 *mm,
                        sb4 *ss,
                        sb4 *fsec,
                        CONST GCIInterval *result
                            );
```

功能描述：

从GCIInterval中获得数据。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
dy	输出	单位： 日
hr	输出	单位： 时
mm	输出	单位： 分
ss	输出	单位： 秒
fsec	输出	总共为5位
result	输入	输入的时间

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注：请参INTERVAL数据类型阅相关手册。

7.4.25 GCIIntervalFromText

函数原型：

```
sword GCIIntervalFromText (
                        void *hndl,
                        GCIError *err,
                        const GCIText *inpstring,
                        size_t str_len,
                        GCIInterval *result
                            );
```

功能描述：

把字符串的日期时间转换成INTERNAL。字符串分为日期字符串和时间字符串。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄

errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
inpstring	输入	格式为”y-m”或者”d hh:mm:ss.f9”或”d hh:mm:ss”
str_len	输入	字符串长度
result	输入/输出	日期或时间的输出

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注：请参INTERVAL数据类型阅相关手册。

7.4.26 GCIIntervalGetYearMonth

函数原型：

```
sword    GCIIntervalGetYearMonth (
                                    void *hndl,
                                    GCIError *err,
                                    sb4 *yr,
                                    sb4 *mnth,
                                    const GCIInterval *interval
                                        );
```

功能描述：

从GCIInterval中获得的年和月的值。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
yr	输出	单位： 年
mnth	输出	单位： 月
interval	输入	输入的GCIInterval 对象数据

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注：请参INTERVAL数据类型阅相关手册。

7.4.27 GCIIntervalSetDaySecond

函数原型：

```
sword GCIIntervalSetDaySecond (
                                    void *hndl,
                                    GCIError *err,
                                    sb4 dy,
                                    sb4 hr,
                                    sb4 mm,
                                    sb4 ss,
                                    sb4 fsec,
                                    GCIInterval *result
                                        );
```

功能描述：

设置GCIInterval的时间。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
dy	输入	单位： 日
hr	输入	单位： 时
mm	输入	单位： 分
ss	输入	单位： 秒
fsec	输入	总共为5位
result	输出	在此结构中得到时间

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注：请参INTERVAL数据类型阅相关手册。

7.4.28 GCIIntervalSetYearMonth

函数原型：

```
sword    GCIIntervalSetYearMonth (
                                    void *hndl,
                                    GCIError *err,
                                    sb4 yr,
                                    sb4 mnth,
                                    GCIInterval *result
                                        );
```

功能描述：

设置GCIInterval的日期。

参数说明：

参数	输入/输出	说明
hndlp	输入	上下文或者环境句柄
errhp	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
yr	输入	单位： 年
mnth	输入	单位： 月
result	输出	GCIInterval对象句柄

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

注释： 请参INTERVAL数据类型阅相关手册。

7.4.29 GCIIntervalCheck

函数原型：

```
sword GCIIntervalCheck (
                        void *hndl,
                        GCIError *err,
                        const GCIInterval *interval,
                        ub4 *valid
                            );
```

功能描述：

检测指定的时间间隔值是否有效。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
interval	输入	要检查的时间间隔值
valid	输出	对于合法时间间隔值，返回0。对于非法值， 返回错误码小节所示的错误位进行或运算的结果。 例如： 如果输入的时间间隔为‘-50’,子类型为yearto month，会返回如下错误GCI_INTER_INVALID_MONTH

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.30 GCIIntervalAssign

函数原型：

```
sword GCIIntervalAssign (
                        void *hndl,
                        GCIError *err,
                        const GCIInterval *inpinter,
                        GCIInterval *outinter
                            );
```

功能描述：

两个时间间隔的赋值。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
inpinter	输入	复制操作数据源
outinter	输出	复制操作目的地

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.31 GCIIntervalCompare

函数原型：

```
sword GCIIntervalCompare (
                        void *hndl,
                        GCIError *err,
                        GCIInterval *inter1,
                        GCIInterval *inter2,
                        sword *result
                            );
```

功能描述：

两个日期时间变量的比较。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
inter1，inter2	输入	待比较的两个时间间隔值
result	输出	比较结果如下： ● inter1 < inter2，输出 - 1 ● inter1 = inter2，输出 0 ● inter1 > inter2，输出 1

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.32 GCIIntervalAdd

函数原型：

```
sword GCIIntervalAdd (
                        void *hndl,
                        GCIError *err,
                        GCIInterval *addend1,
                        GCIInterval *addend2,
                        GCIInterval *result
                            );
```

功能描述：

将addend1和addend2相加，结果写入result中。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
addend1，addend2	输入	时间间隔值加数和被加数
result	输出	计算结果

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.33 GCIIntervalSubtract

函数原型：

```
sword GCIIntervalSubtract (
                        void *hndl,
                        GCIError *err,
                        GCIInterval *minuend,
                        GCIInterval *subtrahend,
                        GCIInterval *result
                            );
```

功能描述：

将两个时间间隔minuend和subtrahend相减，结果写入result中。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
minuend，subtrahend	输入	时间间隔值减数和被减数
result	输出	计算结果

返回值：

如果执行成功， 返回GCI_SUCCESS，否则返回GCI_ERROR。

7.4.34 GCIIntervalToText

函数原型：

```
sword GCIIntervalToText (
                        void *hndl,
                        GCIError *err,
                        const GCIInterval *interval,
                        ub1 lfprec,
                        ub1 fsprec,
                        GCIText *buffer,
                        size_t buflen,
                        size_t *resultlen
                            );
```

功能描述：

将interval值转换成字符串。

参数说明：

参数	输入/输出	说明
hndl	输入	用户会话句柄或环境句柄，在此函数中不做验证
err	输入	错误信息句柄， 该接口调用失败时， 错误信息会存在该句柄上
interval	输入	待转换的时间间隔值

参数	输入/输出	说明
lfprec	输入	年（天）的精度
fsprec	输入	秒的精度
buffer	输出	保存转换结果的字符串
buflen	输入	buffer的长度值
resultlen	输出	转换结果的长度值

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5 数值类型接口

7.5.1 GCINumberAbs

函数原型：

```
sword GCINumberAbs (
                    GCIError            *errhp,
                    const GCINumber     *number,
                    GCINumber           *result
                          );
```

功能描述：

计算NUMBER的绝对值。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	输入数字
result	输出	输入数字的绝对值

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.2 GCINumberAdd

函数原型：

```
sword GCINumberAdd (
                    GCIError            *errhp,
                    const GCINumber     *number1,
                    const GCINumber     *number2,
                    GCINumber           *result
                          );
```

功能描述：

加法运算，将 number1 加上 number2 并在结果中返回。如果任何 number 参数为 NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number1，number2	输入	两个加数
result	输出	输出数字1和数字2的和

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.3 GCINumberAssign

函数原型：

```
sword GCINumberAssign (
                        GCIError            *errhp,
                        const GCINumber  *from,
                        GCINumber          *to
                                );
```

功能描述：

复制运算，将 from 标识的 NUMBER 复制到由 to 标识的 NUMBER，如果from或to参数为 NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
from	输入	要分配的 NUMBER
to	输出	要复制到的 NUMBER

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.4 GCINumberCeil

函数原型：

```
sword GCINumberCeil (
                        GCIError              *errhp,
                        const GCINumber     *number,
                        GCINumber             *result
                                );
```

功能描述：

计算 NUMBER 数值的向上取整值，如果 number 参数为 NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	输入 NUMBER 数值

result	输出	输出已输入 NUMBER 的向上取整值

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.5 GCINumberCmp

函数原型：

```
sword GCINumberCmp (
                        GCIError            *errhp,
                        const GCINumber     *number1,
                        const GCINumber     *number2,
                        sword               *result
                                );
```

功能描述：

比较运算，将 number1 与 number2 进行比较，如果任何 number 参数为 NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number1，number2	输入	进行比较的两个 NUMBER 数值
result	输出	比较结果，可能值如下： ● number1 < number2，返回 negative ● number1 = number2，返回 0 ● number1 > number2，返回 positive

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.6 GCINumberDiv

函数原型：

```
sword GCINumberDiv (
                        GCIError            *errhp,
                        const GCINumber     *number1,
                        const GCINumber     *number2,
                        GCINumber           *result
                                );
```

功能描述：

除法运算，将 number1 除以 number2 并在结果中返回。如果任何 number 参数为 NULL，或除数为0，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number1，number2	输入	被除数与除数
result	输出	商

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.7 GCINumberFloor

函数原型：

```
sword GCINumberFloor (
                        GCIError              *errhp,
                        const GCINumber  *number,
                        GCINumber          *result
                                );
```

功能描述：

计算 NUMBER 数值的向下取整值。如果 number 参数为 NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	输入 NUMBER 数值
result	输出	输出已输入 NUMBER 的向下取整值

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.8 GCINumberFromInt

函数原型：

```
sword GCINumberFromInt (
                        GCIError        *errhp,
                        const void      *inum,
                        uword            inum_length,
                        uword            inum_s_flag,
                        GCINumber    *number
                                 );
```

功能描述：

将 int (如 ub4 或 sb2)转化成 number 类型。如果inum或number 参数为NULL，或者 inum_s_flag 参数输入非法，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
inum	输入	int 指针
inum_length	输入	Int 缓冲区的大小
inum_s_flag	输入	指定整数符号的标志，可能的值如下： ● GCI_NUMBER_UNSIGNED：无符号数值

		● GCI_NUMBER_SIGNED：有符号数值
number	输出	转化后的 number

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.9 GCINumberFromReal

函数原型：

```
sword GCINumberFromReal (
                        GCIError      *errhp,
                        const void    *rnum,
                        uword         rnum_length,
                        GCINumber  *number
                                );
```

功能描述：

将实数转化成数值类型, 如果number或 rnum 参数为 NULL，或者rnum_length参数等于零，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
rnum	输入	real 指针
rnum_length	输入	real 缓冲区的大小
number	输出	转化后的 number

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.10 GCINumberIsInt

函数原型：

```
sword GCINumberIsInt (
                        GCIError              *errhp,
                        const GCINumber    *number,
                        boolean               *result
                                );
```

功能描述：

判断number是否为整数。如果number或result 参数为NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待判断的数值
result	输出	判断结果，如果是整数返回TRUE，否则为FALSE

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.11 GCINumberIsZero

函数原型：

```
sword GCINumberIsZero (
                        GCIError            *errhp,
                        const GCINumber     *number,
                        boolean             *result
                                );
```

功能描述：

判断number是否为0。如果 number 参数为NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待判断的数值
result	输出	判断结果，如果是0返回TRUE，否则为FALSE

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.12 GCINumberMul

函数原型：

```
sword GCINumberMul (
                        GCIError            *errhp,
                        const GCINumber     *number1,
                        const GCINumber     *number2,
                        GCINumber           *result
                                );
```

功能描述：

乘法运算，计算 number1 与 number2 的乘积并在结果中返回。如果任何 number 参数为 NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number1，number2	输入	两个乘数
result	输出	输出 number1 与 number2 的乘积

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.13 GCINumberNeg

函数原型：

```
sword GCINumberNeg (
                    GCIError            *errhp,
                    const GCINumber     *number,
                    GCINumber           *result
                                );
```

功能描述：

对一个数值进行求反操作。如果 number 参数为NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待求反的数值
result	输出	对numbe执行求反操作的结果

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.14 GCINumberRound

函数原型：

```
sword GCINumberRound (
                    GCIError            *errhp,
                    const GCINumber     *number,
                    sword               decplace,
                    GCINumber           *result
                                );
```

功能描述：

将一个数值四舍五入到指定的小数点位。如果 number 参数为NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待执行四舍五入操作的数值
decplace	输入	小数点右侧要保留的小数位数，允许使用负值
result	输出	对numbe执行四舍五入操作的结果

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.15 GCINumberSetZero

函数原型：

```
sword GCINumberSetZero (
                        GCIError        *errhp,
                        GCINumber       *number
                                         );
```

功能描述：

将一个数值初始化为0。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待执行初始化为 0 操作的数值

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.16 GCINumberSign

函数原型：

```
sword GCINumberSign (
                        GCIError              *errhp,
                        const GCINumber       *number,
                        sword                 *result
                                             );
```

功能描述：

获取数值的符号，如果 number 或 result 参数为NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待获取符号的数值
result	输出	数值的符号，可能的取值如下： ● number < 0：-1 ● number == 0：0 ● number > 0： 1

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.17 GCINumberSqrt

函数原型：

```
sword GCINumberSqrt (
                        GCIError              *errhp,
                        const GCINumber       *number,
                        GCINumber              *result
                                              );
```

功能描述：

计算一个数值的平方根，如果 number 参数为NULL或为负值，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待计算平方根的数值
result	输出	对 number 求取平方根的结果

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.18 GCINumberSub

函数原型：

```
sword GCINumberSub (
                        GCIError              *errhp,
                        const GCINumber       *number1,
                        const GCINumber       *number2,
                        GCINumber             *result
                                );
```

功能描述：

减法运算，将 number1 减去 number2 并在结果中返回。如果任何 number 参数为 NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number1，number2	输入	被减数与减数
result	输出	差

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.19 GCINumberToInt

函数原型：

```
sword GCINumberToInt (
                        GCIError              *err,
                        CONST GCINumber       *number,
                        uword                 rsl_length,
                        uword                 rsl_flag,
                        dvoid                 *rsl
                                );
```

功能描述：

将一个数值转换成一个整数。如果 rsl 或 number 参数为 NULL，或者 rsl_flag 参数输入非法，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
inum	输入	要转换为整数的number
inum_length	输入	Int 缓冲区的大小
inum_s_flag	输入	指定输出整数符号的标志，可能的值如下： ● GCI_NUMBER_UNSIGNED：无符号数值 ● GCI_NUMBER_SIGNED：有符号数值
number	输出	转化后的 int 指针

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.20 GCINumberToReal

函数原型：

```
sword GCINumberToReal (
                        GCIError              *errhp,
                        CONST GCINumber       *number,
                        uword                 rsl_length,
                        dvoid                 *rsl
                                );
```

功能描述：

将一个数值转换成一个实数。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	要转换为实数的number
rsl_length	输入	结果缓冲区的大小
rsl	输出	指向实数的指针。

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.21 GCINumberToRealArray

函数原型：

```
sword GCINumberToRealArray (
                        GCIError              *errhp,
                        const GCINumber       **number,
                        uword                 elems,
                        uword                 rsl_length,
                        void                  *rsl
                                );
```

功能描述：

将一个数值型数组转换成一个实数型数组。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	指向要转换的数值型数组的指针
elems	输入	number指针的最大数目
rsl_length	输入	结果缓冲区的大小
rsl	输出	指向用于存储结果的数组的指针

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.5.22 GCINumberTrunc

函数原型：

```
sword GCINumberTrunc (
                        GCIError            *errhp,
                        const GCINumber     *number,
                        sword               decplace,
                        GCINumber           *result
                                );
```

功能描述：

在指定的小数位截断一个数值。如果 number 参数为NULL，则此函数返回错误。

参数说明：

参数	输入/输出	说明
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
number	输入	待截断的数值
decplace	输入	要截断的小数点右侧的小数位数，允许使用负值
result	输出	对 number 执行截断操作的结果

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.6 字符串接口

7.6.1 GCIStringAllocSize

函数原型：

```
sword GCIStringAllocSize (
                        GCIEnv              *envhp,
                        GCIError            *errhp,
                        const GCIString     *vs,
                        ub4                 *allocsize
                                );
```

功能描述：

获取字符串内存的分配大小（以代码点 (Unicode) 或字节为单位），分配的大小大于或等于实际字符串大小。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
vs	输入	返回其分配大小（以字节为单位）的字符串。vs 参数必须是非 NULL 指针
allocsize	输出	返回分配的字符串内存大小（以字节为单位）

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.6.2 GCIStringAssign

函数原型：

```
sword GCIStringAssign (
                        GCIEnv              *envhp,
                        GCIError            *errhp,
                        const GCIString     *rhs,
                        GCIString           **lhs
                                        );
```

功能描述：

将一个字符串分配给另一个字符串。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
rhs	输入	分配的右侧（源）
lhs	输入/输出	分配的左侧（目标）

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.6.3 GCIStringAssignText

函数原型：

```
sword GCIStringAssignText (
                        GCIEnv              *envhp,
                        GCIError            *errhp,
                        const GCIText       *rhs,
                        ub4                 rhs_len,
                        GCIString           **lhs
                                        );
```

功能描述：

将源文本字符串分配给目标字符串。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
rhs	输入	分配的右侧（源）
rhs_len	输入	rhs 字符串的长度（以字节为单位）
lhs	输入/输出	分配的左侧（目标）

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.6.4 GCIStringPtr

函数原型：

```
sword GCIStringPtr (
                        GCIEnv              *envhp,
                        const GCIString     *vc
                                        );
```

功能描述：

获取指向给定字符串文本的指针。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
vc	输入	指向返回其字符串的 GCIString 对象的指针

7.6.5 GCIStringResize

函数原型：

```
sword GCIStringResize (
                        GCIEnv      *envhp,
                        GCIError    *errhp,
                        ub4          new_size,
                        GCIString   **str
                                        );
```

功能描述：

调整给定字符串的内存大小。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
errhp	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
new_size	输入	字符串的新内存大小（以字节为单位），new_size 参数必须包含 NULL 字符作为字符串终止符的空间
str	输入/输出	字符串分配的内存

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.6.6 GCIStringSize

函数原型：

```
sword GCIStringSize (
                        GCIEnv *envhp,
                        const GCIString *vc
                                        );
```

功能描述：

获取给定字符串的大小。

参数说明：

参数	输入/输出	说明
envhp	输入/输出	环境句柄
vc	输入	返回其大小的字符串, 以字节为单位

返回值：

无符号整型

7.7 线程管理接口

7.7.1 GCIThreadClose

函数原型：

```
sword    GCIThreadClose (
                    void                *hndl,
                    GCIError            *err,
                    GCIThreadHandle     *tHnd
                                    );
```

功能描述：

结束指定的线程。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tHnd	输入/输出	要关闭的线程句柄

返回值：

如果执行成功，返回GCI_SUCCESS，否则返回GCI_ERROR。

7.7.2 GCIThreadCreate

函数原型：

```
sword    GCIThreadCreate (
                    void                *hndl,
                    GCIError            *err,
                    Void                (*start)(void *),
                    void                *arg,
                    GCIThreadId         *tid,
                    GCIThreadHandle     *tHnd
                                    );
```

功能描述：

创建一个新的线程。

新线程首先执行对 start 指向的函数的调用，该函数由 arg 给出的参数启动。当该函数返回时，新线程将终止。该函数不应返回值，而应接受一个参数，即 void。

当且仅当 tHnd 为非 NULL时，对 GCIThreadCreate() 的调用需与 GCIThreadClose() 的调用匹配；如果 tHnd 为 NULL，则放置在 *tid 中的线程 ID 在调用线程中无效，因为生成的线程终止的时间未知。

tid 参数应由 GCIThreadIdInit() 初始化，tHnd 应由 GCIThreadHndInit() 初始化。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
start	输入	新线程开始执行的函数
arg	输入	给出由 start 指向的函数的参数
tid	输入/输出	要创建的线程 id 指针
tHnd	输入/输出	要创建的线程句柄指针

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.3 GCIThreadHndDestroy

函数原型：

```
sword    GCIThreadHndDestroy (
                 void                *hndl,
                 GCIError            *err,
                 GCIThreadHandle     **thnd
                              );
```

功能描述：

销毁并解除分配线程句柄。thnd 参数应由 GCIThreadHndInit() 初始化

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
thnd	输入/输出	指向要销毁的线程句柄的指针地址

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.4 GCIThreadHndInit

函数原型：

```
sword    GCIThreadHndInit (
                 void                *hndl,
                 GCIError            *err,
                 GCIThreadHandle     **thnd
                              );
```

功能描述：

分配并初始化线程句柄。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
thnd	输出	指向要初始化的线程句柄的指针地址

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.5 GCIThreadIdDestroy

函数原型：

```
sword    GCIThreadIdDestroy (
                    void                *hndl,
                    GCIError            *err,
                    GCIThreadId         **tid
                                );
```

功能描述：

销毁和解除分配线程 ID。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tid	输入/输出	指向要销毁的线程 ID 的指针

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.6 GCIThreadIdInit

函数原型：

```
sword    GCIThreadIdInit (
                    void                *hndl,
                    GCIError            *err,
                    GCIThreadId         **tid
                                );
```

功能描述：

分配和初始化线程 ID。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tid	输入/输出	指向要初始化的线程 ID 的指针

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.7 GCIThreadInit

函数原型：

```
sword   GCIThreadInit (
                void            *hndl,
                GCIError        *err,
                );
```

功能描述：

初始化线程上下文。空函数，用于与Oracle程序保持兼容。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.8 GCIThreadJoin

函数原型：

```
sword   GCIThreadJoin (
                void            *hndl,
                GCIError        *err,
                GCIThreadHandle *tHnd
                );
```

功能描述：

允许调用线程与另一个线程联接。此函数阻止调用方，直到指定的线程终止。tHnd 参数应由 GCIThreadHndInit() 初始化。多个线程都尝试与同一线程联接的结果是未定义的。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tHnd	输入	要连接线程的线程句柄

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.9 GCIThreadMutexAcquire

函数原型：

```
sword   GCIThreadMutexAcquire (
                void            *hndl,
                GCIError        *err,
                GCIThreadMutex  *mutex
                );
```

功能描述：

获取调用它的线程的互斥锁。

如果互斥锁由另一个线程持有，则调用线程将被阻塞，直到它可以获取互斥锁。

尝试获取未初始化的互斥锁是非法的。

如果线程使用它来获取该线程已持有的互斥锁，则此函数的行为是未定义的。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
mutex	输入/输出	要获取的互斥锁

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.10 GCIThreadMutexDestroy

函数原型：

```
sword    GCIThreadMutexDestroy (
                    void                *hndl,
                    GCIError            *err,
                    GCIThreadMutex      **mutex
                                    );
```

功能描述：

销毁和解除分配互斥锁。

每个互斥锁在不再需要后必须销毁。

销毁未初始化或当前由线程持有的互斥锁是不合法的。

销毁互斥锁不得与对互斥锁执行任何其他操作同时进行。互斥锁在销毁后不得使用。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
mutex	输入/输出	要销毁的互斥锁

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.11 GCIThreadMutexInit

函数原型：

```
sword    GCIThreadMutexInit (
                    void                *hndl,
                    GCIError            *err,
                    GCIThreadMutex      **mutex
                                    );
```

功能描述：

分配和初始化互斥锁。

所有互斥锁在使用前必须初始化。

多个线程不得同时初始化同一个互斥锁。此外，互斥锁必须在被销毁之前不会重新初始化（参见 GCIThreadMutexDestroy()）。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
mutex	输入/输出	要初始化的互斥锁

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.12 GCIThreadMutexRelease

函数原型：

```
sword    GCIThreadMutexRelease (
                    void                *hndl,
                    GCIError            *err,
                    GCIThreadMutex      *mutex
                                    );
```

功能描述：

释放互斥锁。如果互斥锁上有任何线程被阻塞，则其中一个线程会获取该线程并解除阻塞。

尝试释放未初始化的互斥锁是非法的。线程释放它不持有的互斥锁也是非法的。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
mutex	输入/输出	要释放的互斥锁

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.13 GCIThreadProcessInit

函数原型：

```
sword    GCIThreadProcessInit ();
```

功能描述：

执行 GCIThread 进程初始化。空函数，用于与Oracle程序保持兼容。

返回值：

无

7.7.14 GCIThreadTerm

函数原型：

```
sword    GCIThreadTerm (
                    void                *hndl,
                    GCIError            *err
                                    );
```

功能描述：

释放 GCIThread 上下文。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.15 GCIThreadIdSame

函数原型：

```
sword   GCIThreadIdSame (
                void            *hndl,
                GCIError        *err,
                GCIThreadId     *tid1,
                GCIThreadId     *tid2,
                boolean         *result
                        );
```

功能描述：

确定两个 GCIThreadId 类型变量是否代表同一个线程。

如果 tid1 和 tid2 表示同一线程，则结果设置为 TRUE。否则，结果设置为 FALSE。如果 tid1 和 tid2 都是空线程 ID，则结果参数设置为 TRUE。参数 tid1 和 tid2 应该由 GCIThreadIdInit() 初始化。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tid1	输入	指向第一个 ThreadId 的指针
tid2	输入	指向第二个 ThreadId 的指针
result	输入/输出	指向结果的指针

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.16 GCIThreadIdGet()

函数原型：

```
sword   GCIThreadIdGet (
                void            *hndl,
                GCIError        *err,
                GCIThreadId     *tid
                        );
```

功能描述：

检索调用它的线程的 GCIThreadId。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tid	输出	此变量指向用于放置调用线程ID的位置

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.17 GCIThreadIdNull()

函数原型：

```
sword    GCIThreadIdNull (
                    void            *hndl,
                    GCIError        *err,
                    GCIThreadId     *tid,
                    boolean         *result
                            );
```

功能描述：

确定给定的 GCIThreadId 是否为 NULL 线程 ID。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tid	输入	指向要检查的 GCIThreadId 的指针
result	输入/输出	指向结果的指针，如果 tid 是空线程 ID，则结果设置为 TRUE。否则，结果设置为 FALSE。

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.18 GCIThreadIdSet()

函数原型：

```
sword    GCIThreadIdSet (
                    void            *hndl,
                    GCIError        *err,
                    GCIThreadId     *tidDest,
                    GCIThreadId     *tidSrc
                            );
```

功能描述：

将一个 GCIThreadId 设置为另一个 GCIThreadId。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tidDest	输出	指向目标 GCIThreadId 的位置
tidSrc	输入	指向源 GCIThreadId 的位置

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.7.19 GCIThreadIdSetNull()

函数原型：

```
sword   GCIThreadIdSetNull (
                    void            *hndl,
                    GCIError        *err,
                    GCIThreadId     *tid
                            );
```

功能描述：

将 NULL 线程 ID 设置为给定的 GCIThreadId。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄
err	输入/输出	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
tid	输出	一个指向 GCIThreadId 变量的指针，用于设置为 null

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.8 其他接口

7.8.1 GCIPing

函数原型：

```
sword   GCIping (
                    GCISvcCtx   *svchp,
                    GCIError    *errhp,
                    ub4         mode
                            );
```

功能描述：

确认连接和服务器处于活动状态。

参数说明：

参数	输入/输出	说明
svchp	输入	服务上下文句柄
errhp	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上
mode	输入	模式选择，默认为GCI_DEFAULT

返回值：

如果执行成功，返回 GCI_SUCCESS，否则返回 GCI_ERROR。

7.8.2 GCINlsCharSetNameToId

函数原型：

```
ub2   GCINlsCharSetNameToId (
                    void            *hndl,
                    const GCIText   *name
                                );
```

功能描述：

返回数据库字符集ID，本接口仅支持判断AL32UTF8。

参数说明：

参数	输入/输出	说明
hndl	输入/输出	GCI 环境或用户会话句柄。如果句柄无效，则函数返回零
name	输入	指向以 null 结尾的数据库字符集名称的指针。如果字符集名称无效，则该函数返回零

返回值：

如果执行成功，则返回charset_id，否则返回GCI_ERROR。

7.8.3 GCIRowidToChar

函数原型：

```
sword    GCIRowidToChar (
                    GCIRowid     *rowidDesc,
                    GCIText      *outbfp,
                    ub2          *outbflp,
                    GCIError     *errhp
                                  );
```

功能描述：

将 ROWID 数据类型转换为字符型数据。

参数说明：

参数	输入/输出	说明
rowidDesc	输入	GCIDescriptorAlloc()分配并由先前执行的 SQL 语句填充的 ROWID 描述符
outbfp	输出	成功执行此调用后，指向存储字符表示的缓冲区的指针
outbflp	输入/输出	指向输出缓冲区长度的指针。在执行之前，缓冲区长度包含 outbfp 的大小。执行后，它包含转换的字节数。如果在转换过程中有截断，outbfp 包含使转换成功所需的长度。还会返回一个错误
errhp	输入	错误信息句柄，该接口调用失败时，错误信息会存在该句柄上

返回值：

如果执行成功，则返回charset_id，否则返回GCI_ERROR。

8 附录

8.1 错误码

#define GCI_BIND_ERROR	1001
#define GCI_DEFINE_ERROR	1002
#define GCI_ATTRGET_ERROR	1003
#define GCI_ATTRSET_ERROR	1004
#define GCI_HANDLEALLOC_ERROR	1005
#define GCI_HANDLEFREE_ERROR	1006
#define GCI_ENVCREATE_ERROR	1007
#define GCI_LOGONNODB_ERROR	1009
#define GCI_LOGOFF_ERROR	1010
#define GCI_SERVERATTACH_ERROR	1011
#define GCI_SESSIONEND_ERROR	28
#define GCI_SESSIONBEGIN_ERROR	1013
#define GCI_SERVERVERSION_ERROR	1008
#define GCI_TRANSCOMMIT_ERROR	1015
#define GCI_TRANSSTART_ERROR	1016
#define GCI_TRANSROLLBACK_ERROR	1017
#define GCI_STMTPREPARE_ERROR	1018
#define GCI_STMTEXECUTE_ERROR	1019
#define GCI_STMTFETCH_ERROR	1020
#define GCI_DESCALLOC_ERROR	1021
#define GCI_DESCFREE_ERROR	1022
#define GCI_PARAMGET_ERROR	1023
#define GCI_PARAMSET_ERROR	1024
#define GCI_DESCANY_ERROR	1025
#define GCI_DIRPATH_LOADSTREAM_ERROR	1026
#define GCI_LOB_GETLENGTH_ERROR	1027
#define GCI_LOB_READ_ERROR	1028
#define GCI_LOG_WRITE_ERROR	1029
#define GCI_LOG_PROCDESC_ERROR	1030
#define GCI_DB_CONNECT_ERROR	3114
#define GCI_NOT_LOGGED_ON	1012
#define GCI_SHUTDOWN_IN_PROGRESS	1014
#define GCI_DATETIME_CONVERT	1040

注：当厂家设置为3时， 错误码设置为数据库返回的错误码。

1)新增错误消息， 指示函数执行过程中遇到的错误详细信息。如：

"ERROR: [GCI-21560][GCI Driver].parameter %d is NULL\n"

"ERROR: [GCI-1846][GCI Driver].week format is invalid \n"

"ERROR: [GCI-11104][GCI Driver].date format is invalid\n"

"ERROR: [GCI-11105][GCI Driver].type is invalid\n"

"ERROR: [GCI-21405][GCI Driver].type is diff\n"

"ERROR: [GCI-11107][GCI Driver].get year month fun err\n"

"ERROR: [GCI-11108][GCI Driver]. get day second fun err \n"

2)新增日期类型的错误定义，指示日期值的错误之处。

```
#define GCI_DATE_INVALID_DAY        0x1     /* Bad DAy */
#define GCI_DATE_INVALID_MONTH      0x4     /* Bad MOnth */
#define GCI_DATE_INVALID_YEAR       0x10    /* Bad YeaR */
#define GCI_DATE_INVALID_HOUR       0x40    /* Bad HouR */
#define GCI_DATE_INVALID_MINUTE     0x100   /* Bad MiNute */
#define GCI_DATE_INVALID_SECOND     0x400   /* Bad SeCond */
#define GCI_DATE_YEAR_ZERO          0x2000  /* Year may not equal zero
                                               */
#define GCI_DATE_INVALID_FORMAT     0x8000  /* Bad date format input */
```

3)新增日期时间类型的错误定义，指示日期时间值的错误之处。

```
#define  GCI_DT_INVALID_DAY         0x1     /* Bad day */
#define  GCI_DT_INVALID_MONTH       0x4     /* Bad MOnth */
#define  GCI_DT_INVALID_YEAR        0x10    /* Bad YeaR */
#define  GCI_DT_INVALID_HOUR        0x40    /* Bad HouR */
#define  GCI_DT_INVALID_MINUTE      0x100   /* Bad MiNute */
#define  GCI_DT_INVALID_SECOND      0x400   /* Bad SeCond */
#define  GCI_DT_INVALID_FRACCTION   0x800
#define  GCI_DT_YEAR_ZERO           0x2000  /* Year may not equal zero */
#define  GCI_DT_INVALID_FORMAT      0x8000  /* Bad date format input */
```

4)新增时间间隔类型的错误定义，指示时间间隔值的错误之处。

```
#define  GCI_INTER_INVALID_DAY      0x1     /* Bad day */
#define  GCI_INTER_INVALID_MONTH    0x4     /* Bad MOnth */
#define  GCI_INTER_INVALID_HOUR     0x40    /* Bad HouR */
#define  GCI_INTER_INVALID_MINUTE   0x100   /* Bad MiNute */
#define  GCI_INTER_INVALID_SECOND   0x400   /* Bad SeCond */
#define  GCI_INTER_INVALID_FRACSEC  0x1000  /* Bad Fractional second */
```