# 排队叫号模块改进分析

1. **引言：**

排队叫号模块设计之初只为了满足PACS的排队功能的基本应用，之后由于用户应用的需要，在门诊医生工作站中，也使用了相同的排队叫号模块，但同时排队叫号模块也相应做了大量的修改，使得与门诊医生工作站的具体业务有了一定的关联，而目前排队叫号问题的修改，有部分问题则是因为具体业务发生变化而进行修改，并且由于现在需要使用排队叫号的模块的也越来越多，包括体检、检验、输液等，因此需要对目前的排队叫号模块进行适当调整，以满足更多的应用需要。

1. **现存问题：**

排队叫号模块经过一段时间的应用及根据用户反馈的一些需求，并结合当前开发的需要，整理出排队叫号模块主要存在的一些问题，目前排队叫号模块主要问题如下：

1. 与具体业务结合得比较紧密，如门诊排队业务，因此门诊业务的调整将影响到排队叫号模块的修改；
2. 不易扩展和修改，由于与具体业务相关，因此有的功能的增加调整，势必会影像其他业务的修改调整；
3. 不同模块使用排队叫号时，由于业务不同，因此对当前业务来看存在大量无效的字段设计；
4. 不能满足无界面的排队叫号需求，如有的业务模块，不需要显示排队管理界面，如检验，输液等；
5. 当前排队接口诊断其他模块使用比较复杂，对于只是简单使用排队叫号的模块来说，限制和要求太多；
6. **最终目的：**

为解决以上排队叫号模块在应用开发过程中出现的问题，因此排队叫号模块需要最终达到如下目的：

1. 与业务分离，保持排队叫号模块的独立性，复用性，业务的具体变化不影响排队叫号模块的修改；
2. 提供更加完善的排队叫号模块接口以满足各个模块（如PACS、门诊、体检、检验、输液等）对排队叫号的使用；
3. 避免因后续业务扩展，使排队叫号模块变得臃肿且不易于维护；
4. **流程分析：**

由于使用排队叫号模块对应的业务及在使用方式上都有一定区别，因此在对排队叫号拟定修改方案前，充分了解不同业务的具体流程和所需数据，将对排队叫号模块的调整有较高的帮助，更便于提高排队叫号模块的复用度和易用性，使其与各调用模块的结合更加适用，因此下面将分别介绍个模块使用排队叫号的具体流程。

1. **门诊排队叫号业务流程**

在所有涉及到排队叫号的业务中，门诊叫号业务是相对比较复杂的业务之一，在门诊排队叫号中，排队叫号的应用方式可分为如下几种：

1. 先分诊台分诊，然后医生呼叫；

病人挂号后，先到指定的科室分诊处进行报到，报到后，护士根据各诊室的就诊情况，将病人分配到指定的诊室中，分诊后，当医生完成一个病人诊断后，就呼叫下一个病人进行就诊。

分诊后------报到------入队（队列名称按科室生成）

医生-----显示本诊室---呼叫-----本诊室

此流程包括专家门诊，直接挂号到具体专家本人。

1. 分诊台不分诊，医生主动呼叫；

在该模式下，病人挂号后，病人统一到挂号指定科室的候诊大厅候诊，医生根据当前情况，主动呼叫下一个需要就诊的病人到指定诊室就诊。

挂号----入队（队列名称按科室生成）

医生-----显示本科室----呼叫-----本科室-------根据排队号码优先选择。

1. 挂号自动分诊，医生主动呼叫；

挂号时，自动根据门诊就诊情况，分配检查诊室，此时病人信息将显示到科室候诊信息队列中，医生根据当前检查情况呼叫下一病人进行就诊。

挂号-----（自动）----入队（队列名称按科室生成）

医生----显示本诊室----呼叫-----本诊室。

1. 分诊和医生都不呼叫，也就是不进行叫号操作；

有的医院当不启用排队叫号相关操作时，应隐藏对应的排队叫号相关功能。

不使用排队叫号。

门诊叫号总结：

1. 挂号入队或者分诊台报到入队-------入队（队列名称按科室规则生成）
2. 医生------显示本诊室------呼叫本诊室；

医生------显示本科室-----呼叫本科室；

医生----显示本人-----呼叫本人；

在这几种使用方式中，门诊可通过相关参数配置实现不同的处理流程。在门诊叫号业务中，涉及到的相关功能主要包括，顺乎、直呼、广播、暂停、接诊、回诊、优先（插队）、恢复、完成、转诊(更改就诊科室)、查找、修改、弃号等功能。其中部分功能与具体业务相关，如回诊、转诊、接诊等功能，因此在排队叫号模块中，需要同时考虑到与具体业务功能相关的处理。

另外在门诊排队叫号业务的排队信息显示中，根据不同的使用流程，也分别对应几种不同的显示方式，不同的显示方式，在门诊叫号中都有对应参数设置，几种显示方式如下：

1. 显示当前科室下，所有未就诊和当前医生已接诊的排队信息；
2. 显示当前诊室下，所有未就诊和当前医生已接诊的排队信息；
3. 显示当前诊室下，所有未就诊和当前医生已接诊或诊室为空和医生为空的排队信息；
4. 显示医生属于当前医生的所有未就诊和当前医生已接诊的排队信息；

排队号码的生成则由门诊业务这边产生，门诊业务通过调用相应存储过程，并根据其中的生成规则，取得当前可以使用的排队号码，排队号码生成规则则可通过修改存储过程的方式，实现自定义排队号码的生成。呼叫方式可分为集中呼叫和诊室呼叫，可通过设置站点来实现。

在门诊业务中使用到的回诊序号，也是由门诊业务这边产生，而产生规则则是取当前业务中最大的回诊序号加一的方式。

在门诊业务中，每次运行排队叫号时，都需要对其中的部分排队数据进行清除，避免因时间过久，而产生大量无效的排队数据，而数据清除方式则是删除指定天数之前的所有数据。

具体门诊叫号业务分析，可参考《分诊台和医生站排队叫号修改方案》



1. **PACS排队叫号业务流程**

PACS排队叫号目前主要有两种应用模式，医院根据具体的使用情况，在流程管理中配置具体的应用模式，目前的两种应用方式如下：

1. 按检查科室排队，各执行间主动呼叫；

此种模式下，病人报到后，不需要指定具体的检查执行间，病人在候诊大厅等待检查，各执行间根据具体检查的繁忙程度，呼叫对应的检查患者进行检查。

1. 前台报到分诊，然后执行间呼叫；

病人到科室检查时，先到前台报到，报到时，由护士或者医生根据具体情况分配检查执行间，病人到相应的检查执行间外等待检查，当技师做完一个检查后，呼叫下一个检查患者进行检查。

在PACS排队叫号中，不涉及到回诊功能，液晶屏中排队信息的显示，可根据当前不同的处理方式，在每个执行间外显示当前执行间的排队信息，也可以在候诊大厅中，显示各执行间的检查情况。

PACS业务中，排队队列中的数据由于每天都会清除前一天的所有数据，因此排队号码的生成则取当天排队队列中的最大号码加一的数字。并且由于PACS业务中不涉及回诊功能，因此也不需要产生回诊序号。

1. **体检排队叫号业务流程**

在体检排队叫号的业务流程中，排队叫号的使用方式与门诊和PACS业务中的应用方式又有一些区别。体检过程中，一个患者将会同时产生多个体检项目，而每个体检项目都需要进行一次排队操作，但又不能将患者同时安排进入多个体检排队队列，必须等到一个体检项目完成后，对另外体检项目报到检查时，才进入对应的排队队列，因此体检这边需要控制同一时刻，一个患者只能在一个排队队列中。

目前体检排队使用的排队号由体检业务产生，相关数据也是由体检业务直接访问排队叫号相关数据表生成，体检排队业务的数据生成，并未使用排队叫号相关接口，只是利用了排队叫号模块的操作界面，对已存在的队列数据进行相关操作。

排队叫号的具体业务流程可参考分析文档《中联体检排队叫号方案》



1. **检验排队叫号业务流程**
2. **输液排队叫号业务流程**
3. **相关要点：**
4. **排队号码生成规则**

患者排队时，默认排队号码由排队模块自动产生（取当前排队队列中的最大排队号码加一），也可由各调用模块根据各自所需的规则传入所需的排队号码。

1. **排队数据保存时间**

排队数据的保存时间可根据不同的业务进行设置，默认将清除当天之前的所有数据，由此避免产生过多的且无效的排队数据。

1. **回诊的应用**

在门诊就诊后的病人，在执行相应检查后，再回到门诊诊断的过程，可叫做回诊。有的医院在排队叫号过程中，使用了回诊病人优先原则，即回诊的病人优先于普通排队就诊的病人；而有的医院，对于回诊病人，则需要重新排队就诊。

回诊过程中产生的回诊号码，则由相关业务产生，比如门诊业务中，回诊序号取当前业务中的最大回诊序号加一的方式，而在有的业务中则不需要回诊功能，也不需要生成此数据。

1. **接诊的应用**

接诊则主要指，呼叫一个病人后，病人到达了指定诊室或检查间进行就诊检查，接诊后，检查数据的相关状态也需要随之改变。

1. **转诊的应用**

有时候可能存在就诊科室安排错误，或者因其他原因造成的检查科室安排有误，或者因诊室具体情况，需要将检查病人转到其他诊室就诊的情况下，就需要转诊功能。将病人转移到其他诊室就诊的处理可理解位转诊。

1. **排队序号的应用**

按照入队的顺序，自动产生排队序号，然后提供接口，可以提取GetOrder和修改排队序号ChangeOrder对于回诊的病人，如果允许回诊病人优先，则可将该病人根据排队序号重新进行排队处理，否则更新当前的排队序号。

另外就是由于排队序号主要用于对检查病人进行排队，而有的医院在排队时，有特定的排队规则，如前两个是初诊病人，然后是回诊病人，再然后是优先病人，所以针对排队序号，需要提供一个接口能够根据具体情况进行灵活的调节和操作。

注：排队序号插队时的生成规则和实现如下：

---------------------------------------------------------------------

如需要在序号a和序号b之间插入一个数字，则这个数字c可通过以下方式取得

设置x:表示b-a的差，如26-25,此时X=1; 25.5-25,此时X=0.5

设置y:表示b-a的差且只获取小数部分并转换为整数后的值，

如25.5-25=0.5,获取小数部分转整，此时Y=5；25.68-25.65=0.03，获取小数部分转整，此时Y=3; 26-25=1,此时Y=1

那么要获取的序号C可以表示为：c=a+X/2+iif(Y=5,X/10,iif(Y=3,X/6,0))

只所以这么计算，是因为在同一个地方重复插队时，避免小数位数过快增长

----------------------------------------------------------------------

create or replace function zl\_排队叫号队列\_获取插队序号(

队列ID\_In 排队叫号队列.ID%Type,

序号1\_In 排队叫号队列.排队序号%Type,

序号2\_In 排队叫号队列.排队序号%Type

)return varchar2 as

n\_decResult Number;

n\_div Number;

n\_where Number;

n\_result Number;

begin

n\_decResult := To\_Number(序号2\_In) - To\_Number(序号1\_In);

n\_where := substr(n\_decResult, instr(n\_decResult, '.') + 1, 6 ) \* 1;

if n\_where = 5 or n\_where=3 then

n\_div := 10;

if n\_where=3 then

n\_div := 6;

end if;

n\_result := To\_Number(序号1\_In)+n\_decResult/2+n\_decResult/n\_div;

else

n\_result := To\_Number(序号1\_In)+n\_decResult/2;

end if;

return n\_result;

Exception

When Others Then return -1;

end zl\_排队叫号队列\_获取插队序号;

/

1. **修改思路：**

要实现第三节中的所有目的，因此排队叫号模块需要做较大的调整，并且相应关联业务也需随之进行部分调整，下面将列举出本次调整的应用规则和主要修改方向，如下：

1. 使用排队叫号的模块，所有操作功能必须通过排队叫号模块的接口进行操作，特定的相关查询，可与排队叫号队列中的数据进行关联查询；
2. 当前排队状态的数据显示，可由渠道修改LCDSHOW部件，并直接访问排队叫号队列数据的方式进行查询和显示；
3. 排队叫号模块将主要由三种类型构成，分别如下：
4. clsQueueManage1类：负责排队叫号相关功能的处理，如入队、顺乎、暂停等相应操作；
5. ucQueueManage组件：带界面的排队叫号操作组件，集成了排队数据的显示和相应的操作功能界面；
6. clsLCDShow类：单独的一个工程项目，负责当前排队情况的数据显示，由渠道或者用户自行修改，可直接访问排队叫号队列中的数据；
7. 排队叫号模块涉及的数据结构将做相应调整，减少与具体业务的数据关联；
8. 排队叫号模块将提供通用接口，供所有业务调用，尽量满足各业务模块的兼容处理，避免不同业务调用不同的排队叫号操作接口；
9. ~~新增排队叫号模块，原有排队叫号模块将不在增加相应功能，也避免进行过多维护；~~
10. **实现方法：**
11. **数据结构调整：**

由于之前的排队叫号模块与门诊业务结合相当紧密，在数据结构的涉及上，也包含了相应的业务数据，因此在本次调整中，将具体与业务相关的数据结构进行调整，比如医生姓名,科室ID,回诊序号之类，这类信息可存入排队叫号中名为“扩展数据”的XML字段中，也可存入业务相关的具体表中，排队叫号的相应功能不需要关心该字段的具体格式和内容，只由具体的业务对该字段进行相应访问处理。

由于目前有好几家较大的医院都在使用门诊排队叫号，并且在目前的使用过程中，也没有其他相关问题，同时为了避免因本次改动造成的影响，需要考虑对以前的功能进行兼容，所以在涉及到数据结构的修改方面，保持最大限度的不对原有数据结构进行改动，新的修改方案所需的数据字段，可在原有结构表中进行扩充，对于确实不需要使用的数据字段，也不能将其字段删除，只是在以后的处理过程中，不在使用该字段存储数据。

|  |  |  |  |
| --- | --- | --- | --- |
| **排队叫号队列（排队队列信息）** | | | |
| **字段名** | **字段类型(长度)** | **约束** | **备注** |
| ID | Number(18) | 主键 | 继续使用 |
| 队列名称 | Varchar2(64) |  | 继续使用 |
| 排队号码 | Varchar2(20) | 与业务类型为唯一键 | 继续使用 |
| 业务ID | NUMBER(18) | 可空，索引 | 继续使用 |
| 患者姓名 | Varchar2(64) | 可空 | 继续使用 |
| 业务类型 | Number(5) | 与排队号码为唯一键 | 继续使用 |
| 排队时间 | DateTime | 索引 | 继续使用 |
| 排队状态 | Number(1) | 索引，默认0 | -1-占位中(如预约后，可将排队信息插入该队列，直到报到后，才启用排队数据)，  0-排队中，   1. 呼叫中（当前正在呼叫的数据为呼叫中），   2-已弃号（放弃呼叫），  3-已暂停（因其他事情暂停当前呼叫），  4-已完成（完成检查后的状态）  5-广播(新方案弃用只兼容)，  6-回诊(新方案弃用只兼容)，  7-已呼叫（已执行呼叫，当并非当前呼叫的数据为已呼叫） |
| *病人ID* | *NUMBER(18)* |  | *新方案弃用，只保持兼容* |
| *科室ID* | *NUMBER(18)* |  | *新方案弃用，只保持兼容* |
| *医生姓名* | *Varchar2(64)* | *可空* | *新方案弃用，只保持兼容* |
| *优先* | *NUMBER(1)* |  | *新方案弃用，只保持兼容* |
| *回诊序号* | *NUMBER(18)* |  | *新方案弃用，只保持兼容* |
| 诊室(目的地) | Varchar2(64) | 可空 | 新方案弃用，只保持兼容 |
| *是否分时点* | *NUMBER(1)* |  | *新方案弃用，只保持兼容* |
| *排队标记* | *VARCHAR2(10)* |  | *新方案弃用，只保持兼容* |
| *呼叫医生* | *VARCHAR2(20)* |  | *新方案弃用，只保持兼容* |
| *呼叫时间* | *DATE* |  | *新方案弃用，只保持兼容* |
| **排队序号** | **Varchar2(20)** |  | **用于排队顺序，通过调整此顺序可进行优先，插队等操作** |
| **当前处理时间** | **DateTime** |  |  |
| **当前处理人** | **Varchar2(20)** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **排队语音呼叫（排队呼叫信息）** | | | |
| **字段名** | **字段类型(长度)** | **约束** | **备注** |
| ID | Number(18) | 主键 | 继续使用 |
| 队列ID | Number(18) | 可以为空 | 继续使用 |
| 呼叫内容 | Varchar2(1024) |  | 继续使用 |
| 呼叫站点 | Varchar2(64) |  | 继续使用 |
| *队列名称* | *VARCHAR2(20)* |  | *新方案弃用，只保持兼容* |
| **生成时间** | **DateTime** |  | **根据生成时间排序呼叫** |

1. 模块接口调整：

在现有排队叫号模块中，新增接口类clsQueueOperation对排队叫号数据进行操作，所有方法在该类中新增，原clsQueueManage接口类目前不做任何改动，以保持对原有功能的兼容性。

在新的排队叫号模块中，各接口的参数将不涉及到具体业务数据，不过允许其中一些接口提供与具体业务进行关联操作的相关关联数据，以便能够将排队叫号中的数据与具体业务进行所需的关联数据查询。各业务模块可通过使用clsQueueOperation类提供的接口对排队叫号的相关功能进行操作，所有操作后的数据，将保存在“排队叫号队列”的数据表中，clsQueueOperation及其他相关组件的使用接口和相关说明见下表：

|  |  |  |  |
| --- | --- | --- | --- |
| **clsQueueOperation** | | | |
| **方法名** | **方法原型** | | **说明** |
| InitQueue | Public sub InitQueue(  cnOracle As ADODB.Connection,  byval lngSysNo as long,  byval intWorkType As Integer)  cnOracle:oracle数据库连接对象  lngSysNo：系统号  intWorkType：业务类型 | | 初始化队列 |
|  |  | |  |
| ValidDays | 清除指定天数前的数据，超出该天数的数据才能够被删除 | | 数据有效天数 |
| CallWay | 0-系统默认，1-微软语言 | | 呼叫方式 |
| CallCount | 默认1 | | 呼叫次数 |
| CallSpeed |  | | 呼叫速度 |
| CallLen | 默认10秒 | | 呼叫时间长度 |
| LocalStation | 默认当前站点 | | 本机站点名 |
| CallStation | 默认当前站点 | | 带呼叫站点名 |
| CallTarget | 呼叫目的地  呼叫前，判断该属性是否为空，如果不为空，则写如数据库中的诊室(目的地)数据字段中 | | 呼叫前设置病人的检查目的地 |
|  |  | |  |
| InsertQueue | Public function InsertQueue(  Byval strQueueName as string,  Optional Byval strQueueNo as string,  Optional Byval strBusinessId as string,  Optional byval strName as string,  Optional byval strTarget as string,  Optional byval strCustomOrder as string  ) as long  strQueueName：要插入的队列名称  strQueueNo：排队号码  strBusinessId：业务ID  strName：姓名  strTarget：目的地  strCustomOrder：自定义排队序号  返回插入后的排队ID  入队后，并没有立即开始排队，需要调用startqueue才开始进入排队状态。  举例如下：  Call InsertQueue(“队列1”, 23, 2369,, “张三”, “诊室1”, “”, “性别=男”,“年龄=26”) | | 插入队列数据  需调用过程处理数据的修改 |
| InsertQueue1 | Public function InsertQueue1(  Byval strQueueName as string,  ParamArray arrExpandData() As Variant) as long  strQueueName：要插入的队列名称  aryExpandData：插入自定义扩展的数据  返回插入后的排队ID  入队后，默认排队状态为-1，需调用startqueue才开始排队。  举例如下：  Call InsertQueue(“队列1”, 23, 2369,, “张三”, “诊室1”, “”, “性别=男”,“年龄=26”) | |
| DeleteQueue | Public sub DeleteQueue(byval lngQueueID as long)  lngQueueID：排队ID  说明：根据传入的队列id，删除排队叫号队列中的数据 | | 删除队列数据  需调用过程处理数据的修改 |
| ClearQueue | Public sub ClearQueue(Byval strQueueName as string)  strQueueName:队列名称  说明：根据传入的队列名称，删除排队叫号队列中所有队列名称为该传入值的数据。清除数据时，需根据ValidDays判断删除范围。 | | 清除队列数据  需调用过程处理数据的修改 |
| UpdateQueue | Public Sub UpdateQueue(  byval lngQueueID as long,  ParamArray arrUpdateColData() As Variant)  lngQueueID：排队ID  arrUpdateColData:的格式为’姓名=张四’, ‘性别=女’  说明：根据队列id，查询数据库中的数据，并更新相应值 | | 更新队列数据  需调用过程处理数据的修改 |
| GetQueueInf | Public function GetQueueInf(  byval lngQueueId as long) as Adodb.RecordSet  lngQueueID：排队ID  说明：根据队列id，查询数据，获取该队列在表中的所有信息 | | 获取当前排队的队列详细信息 |
| WriteTarget | Public sub WriteTarget(byval lngQueueId as long)  lngQueueID：排队ID  说明：调用该方法并根据队列id，将CallTarget属性值写入数据表的诊室字段中。 | | 写入检查目的地  需调用过程处理数据的修改 |
| OrderCall | Public sub OrderCall(Byval strQueueName as string)  strQueueName:队列名称  说明：通过业务Id查询该队列下的所有排队数据，并根据排队序号排序，选择第一条数据，触发呼叫前和呼叫后事件，并将组织的呼叫内容写入“排队语音呼叫”的表中。然后改变其排队状态，将之前为呼叫中的状态修改为已呼叫，将当前呼叫的检查排队状态修改为呼叫中。写入数据时需要写入呼叫所在的站点名称。 | | 顺序呼叫  需调用过程处理数据的修改 |
| SpecifiedCall | Public sub SpecifiedCall(byval lngQueueID as long)  lngQueueID：排队ID  说明：根据队列id获取呼叫数据，触发呼叫前和呼叫后事件，并将组织的呼叫内容写入“排队语音呼叫”的表中。然后改变其排队状态，将之前为呼叫中的状态修改为已呼叫，将当前呼叫的检查排队状态修改为呼叫中。写入数据时需要写入呼叫所在的站点名称。 | | 直接呼叫  需调用过程处理数据的修改 |
| BroadcastCall | Public sub BroadcastCall(byval lngQueueID as long)  lngQueueID：排队ID  说明：根据队列id获取呼叫数据，触发呼叫前和呼叫后事件，并将组织的呼叫内容写入“排队语音呼叫”的表中。不改变队列的排队状态。写入数据时需要写入呼叫所在的站点名称。 | | 广播呼叫  需调用过程处理数据的修改 |
| CustomCall | Pubic sub CustomCall (byval strContext as string)  strContext:语音内容  说明：将strContext的呼叫内容写入“排队语音呼叫”表中，写入数据时需要写入呼叫所在的站点名称。 | | 自定呼叫  需调用过程处理数据的修改 |
| StartQueue | Public sub StartQueue(byval lngQueueID as long)  lngQueueID：排队ID  说明：修改其排队状态，将其值修改为0，表示开始排队，并触发对应事件OnWorkBefore和OnWorkAfter | | 开始排队  需调用过程处理数据的修改 |
| PauseQueue | Public sub PauseQueue(byval lngQueueID as long)  lngQueueID：排队ID  说明：修改其排队状态，将其值修改为3，表示暂停排队，并触发对应事件OnWorkBefore和OnWorkAfter | | 暂停排队  需调用过程处理数据的修改 |
| AbstainQueue | Public sub AbstainQueue(byval lngQueueID as long)  lngQueueID：排队ID  说明：修改其排队状态，将其值修改为2，表示放弃排队，并触发对应事件OnWorkBefore和OnWorkAfter | | 放弃排队  需调用过程处理数据的修改 |
| RestoreQueue | Public sub RestoreQueue(byval lngQueueID as long)  lngQueueID：排队ID  说明：修改其排队状态，将其值修改为0，再重新计算排队序号，排到最后，并触发对应事件OnWorkBefore和OnWorkAfter | | 重新排队  （排到最后位置）  需调用过程处理数据的修改 |
| CompleteQueue | Public sub CompleteQueue(byval lngQueueID as long)  lngQueueID：排队ID  说明：修改其排队状态，将其值修改为4，表示完成排队，并触发对应事件OnWorkBefore和OnWorkAfter | | 完成排队  需调用过程处理数据的修改 |
| DoCustomWork | Public sub DoCustomWork(  byval lngQueueId as long,  byval lngOperationType as long)  lngQueueId：排队ID  lngOperationType：操作类型，为5及以后的数字  说明：不修改排队状态，只触发对应事件OnWorkBefore和OnWorkAfter | | 执行自定义业务操作，如接诊、回诊、转诊等 |
| GetQueueState | Public function GetQueueState(  byval lngQueueID as long) as long  lngQueueID：排队ID  -1-占位中，0-排队中，1-呼叫中，2-已弃号，3-已暂停，4-完成就诊,5-弃用，6-弃用，7-已呼叫， | | 排队状态 |
| DoLoopCall | Pubic sub DoLoopCall ()  说明：检索排队语音呼叫表中属于本站点的数据，并根据生成时间排序，在依次读取数据调用呼叫接口进行呼叫。 | | 检索属于本站点的语音呼叫内容，并进行语音播放 |
| ChangeOrder | Public sub ChangeOrder(  byval lngQueueID as long,  byval strNewOrder as string)  lngQueueID：排队ID  strNewOrder：新的排队序号 | | 调整排队顺序 |
| GetOrder | Public function GetOrder(  byval lngQueueID as long) as string  lngQueueID：排队ID  返回对应的排队序号 | | 获取排队序号 |
| GetCustomOrderWhereStr | Public function GetCustomOrderWhereStr() as string  说明：获取当前业务下的数据排序条件，并返回排序字符串，该方法需要通过调用过程的方式获取，以便用户能够修改排序方式。调用的过程需要有“业务类型”的参数输入。 | | 获取自定义排序条件 |
| GetInsertOrder | Public function GetInsertOrder(  Byval lngInsertQueueId as long,  Byval strOrderNo1 as string,  Byval strOrderNo2 as string  ) as string  lngInsertQueueID：需要插入队列的当前排队ID  strOrderNo1：插队时的前一个排队队列的排队序号  strOrderNo2：插队时的后一个排队队列的排队序号  说明：该方法需要调用“排队序号的插队序号获取”过程获取序号，以便用户可通过直接修改过程的方式修改序号生成规则。 | | 获取插队序号 |
| GetMinOrder | Public function GetMinOrder(  Byval strQueueName as string) as string  strQueueName:队列名称  说明：根据业务id及传入的队列名称，获取该队列中最小的排队序号 | | 获取当前队列最小排队序号。 |
| GetMaxQueueNo | Public function GetMaxQueueNo(  Optional byval strFilter as string) as string  strFilter:获取最大号码的自定义条件，默认为空表示当前业务下的最大排队号码  如：‘队列名称=超声科’，则获取队列名称为超声科的当前最大号码  ‘排队标记=黑白超’，则获取排队标记为黑白超的当前最大号码 | | 获取指定条件下的最大排队号码（排队号） |
| SerachQueue | Public function SerachQueue(  byval strFilter as string) as long  strFilter：过滤条件  返回检索到的排队ID | | 根据条件检索，并返回排队ID |
|  |  | |  |
| OnCallBefore | Event OnCallBefore(  rsQueue as adodb.recordsets,  Byval lngCallWay as long,  strCallContext as string,  blnCancel as Boolean)  rsQueue：当前呼叫的数据信息  lngCallWay：呼叫方式，1-顺乎，2-直呼，3-广播  strCallContext：具体业务返回的呼叫内容，当内容为空时，将根据对应的存储过程生成呼叫内容  blnCancel：是否撤销本次呼叫 | | 呼叫前事件 |
| OnCallAfter | Event OnCallAfter(  rsQueue as adodb.recordsets,  Byval lngCallWay as long)  rsQueue：当前呼叫的数据信息  lngCallWay：呼叫方式，1-顺乎，2-直呼，3-广播 | | 呼叫后事件 |
|  |  | |  |
| OnWorkBefore | Event OnWorkBefore(  rsQueue as adodb.recordsets,  Byval lngOperationType as long,  blnCancel as Boolean)  rsQueue：当前操作的排队数据  lngOperationType：当前操作类型   1. 暂停，2-弃号，3-恢复，4-完成，5及以后为自定义功能类型   blnCancel：是否运行当前操作 | | 操作前事件 |
| OnWorkAfter | Event OnWorkAfter(  rsQueue as adodb.recordsets,  Byval lngOperationType as long)  rsQueue：当前操作的排队数据  lngOperationType：当前操作类型  1-暂停，2-弃号，3-恢复，4-完成，5及以后为自定义功能类型 | | 操作后事件 |
|  |  | |  |
| OnCalling | Event OnCalling(  byval lngQueueId as long,  byval strCallContext as string)  lngQueueID：当前正在呼叫的排队ID  strCallContext:呼叫的内容 | | 呼叫事件 |
|  |  | |  |
| **UcQueueManage**  默认显示本业务类型下的所有队列中的排队数据  界面中，默认功能包含：  打号|顺乎，直呼，广播|优先，插队，接诊|暂停、弃号、恢复，完成|刷新、查找、修改|设置 | | | |
|  | |  |  |
| InitQueue | | Public sub InitQueue(  cnOracle As ADODB.Connection,  byval lngWorkType as long) | 初始化队列 |
|  | |  |  |
| DisplayColNames | | Property Get DisplayColNames() as String  Property Let DisplayColNames (value as String) | 界面显示字段 |
| ValidDays | | Property Get ValidDays() as long  Property Let Validays(value as long) | 数据有效天数 |
| IsGroup | | Property Get IsGroup() as Boolean  Property Let IsGroup(value as Boolean) | 是否分组显示 |
| Interval | | Property Get Interval() as long  Property Let Interval (value as long) | 轮询间隔时长 |
| objQueueManage | | Property Get objQueueManage() as clsQueueManage | 排队处理对象 |
| objCmdBar | | Property Get objCmdBar() as object | Cmdbar对象 |
| objWaitList | | Property Get objWaitList() as object | Reportcontrol对象 |
| objCallList | | Property Get objCallList() as object |
| DisplayQueueNames | | Property Get DisplayQueueNames() as String  Property Let DisplayQueueNames (value as String)  如果为空，则显示当前业务类型下的所有队列中的排队和呼叫数据 | 允许显示的队列名称 |
| GroupName | | Propety Get GroupName() as string  Property Let GroupName(value as string) | 组名（对排队数据分组显示） |
| CustomOrderName | | Property Get CustomOrderName() as string  Property Let CustomOrderName(value as string)  **如果为空，排序默认使用排队序号** | 自定义排序字段 |
|  | |  |  |
|  | |  |  |
| Start | | Public sub Start() | 开始轮询数据显示 |
| Stop | | Public sub Stop() | 停止数据轮询显示 |
| ShowParameter | | Public sub ShowParameter() | 显示参数设置 |
|  | | Private sub Timer() | 轮询数据显示和呼叫 |
|  | |  |  |
|  | |  |  |
| OnCallBefore | | Event OnCallBefore(  rsQueue as adodb.recordsets,  Byval lngCallWay as long,  strCallContext as string,  blnCancel as Boolean)  rsQueue：当前待呼叫的数据  lngCallWay：当前呼叫方式  strCallContext：呼叫的内容文本  blnCancel：是否撤销本次呼叫 | 呼叫前事件 |
| OnCallAfter | | Event OnCallAfter(  rsQueue as adodb.recordsets,  Byval lngCallWay as long)  rsQueue：当前呼叫的数据信息  lngCallWay：呼叫方式，1-顺乎，2-直呼，3-广播 | 呼叫后事件 |
| OnWorkBefore | | Event OnWorkBefore(  rsQueue as adodb.recordsets,  Byval lngOperationType as long,  blnCancel as Boolean)  rsQueue：当前操作的排队数据  lngOperationType：当前操作类型   1. 暂停，2-弃号，3-恢复，4-完成，5及以后为自定义功能类型   blnCancel：是否运行当前操作 | 操作前事件 |
| OnWorkAfter | | Event OnWorkAfter(  rsQueue as adodb.recordsets,  Byval lngOperationType as long)  rsQueue：当前操作的排队数据  lngOperationType：当前操作类型  1-暂停，2-弃号，3-恢复，4-完成，5及以后为自定义功能类型 | 操作后事件 |
| OnQueryQueue | | Event OnQueryQueue(  Byval lngShowModle as long,  rsData as adodb.recordsets,  blnUseCustom as Boolean)  rsData:返回OnQueryQueue事件中查询的数据，rsData返回时必须包含排队id数据，如果rsData中不包含排队ID，界面中将不加载数据。  lngShowModle：排队数据的显示模式  0-显示排队数据，1-显示暂停数据，  2-显示弃号数据，3-显示完成数据，4-显示未分队数据，  5-显示呼叫中和已呼叫数据  blnUseCustom:是否使用当前事件的查询数据 | 查询呼叫数据事件 |
| OnReadBefore | | Event OnReadBefore(  rsData as adodb.recordsets,  Byval lngListType as long,  blnCancel as Boolean)  lngListType：0-排队列表数据读取，1-呼叫列表数据读取 | 读取数据前事件 |
| OnReadAfter | | Event OnReadAfter(  rsData as adodb.recordsets,  Byval lngListType as long)  lngListType：0-排队列表数据读取，1-呼叫列表数据读取 | 读取数据后事件 |
| OnItemClick | | Event OnItemClick(  byval lngListType as long,  Byval lngQueueId as long,  objReoprtRow as IReportRow,  objReportItem as IReportRecordItem)  lngListType:列表类型0-排队列表，1-呼叫列表  lngQueueId:当前选中的队列ID  objReoprtRow：ReportRow对象  objReportItem：ReportItem对象 | 列表单击事件 |
| OnItemDblClick | | Event OnItemDblClick(  byval lngListType as long,  Byval lngQueueId as long,  objReoprtRow as IReportRow,  objReportItem as IReportRecordItem)  lngListType:列表类型0-排队列表，1-呼叫列表  lngQueueId:当前选中的队列ID  objReoprtRow：ReportRow对象  objReportItem：ReportItem对象 | 列表双击事件 |
| OnQueueListChange | | Event OnQueueListChange(  Byval lngListType as long,  ……) | 列表切换事件 |
| OnCmdBarInit | | Event OnCmdBarInit(  Byval CmdBar as ICommandBar) | 按钮初始化 |
| OnCmdBarUpdate | | Event OnCmdBarUpdate(  ByVal Control As ICommandBarControl) | 按钮更新事件 |
| OnCmdBarExecute | | Event OnCmdBarExecute(  ByVal Control As ICommandBarControl) | 按钮执行事件 |
|  | |  |  |
| **clsLCDShow**  对排队情况进行显示，由渠道或者用户自行修改维护 | | | |
|  | |  |  |
| InitLCDShow | | Public sub InitLCDShow(  cnOracle As ADODB.Connection,  byval lngWorkType as long) | InitLCDShow |
|  | |  |  |
| Interval | | Property Get Interval() as long  Property Let Interval(value as long) | 轮询间隔时长 |
| ModuleNumber | | Property Get ModuleNumber() as long  Property Let ModuleNumber(value as long) | 模块号 |
| SysNumber | | Property Get SysNumber() as long  Property Let SysNumber(value as long) | 系统号 |
| objParameter | | Property Get objParameter() as variant  Property Set objParameter (value as variant) | 参数对象 |
|  | |  |  |
|  | |  |  |
| Start | | Public sub Start() | 开始显示 |
| Stop | | Public sub Stop() | 停止显示 |
| ShowParameter | | Public Sub ShowParameter() | 显示参数配置 |
|  | |  |  |
| OnQuery | | Event OnQuery(rsData as adodb.recordsets,  blnUseCustom as Boolean) | 数据显示前查询事件 |
| OnReadBefore | | Event OnReadBefore(rsData as adodb.recordsets,  blnCancel as Boolean) | 数据读取前事件 |
| OnReadAfter | | Event OnReadAfter(rsData as adodb.recordsets) | 数据读取后事件 |

1. **关联模块修改：**

略……

## 总结

通过排队叫号的修改，可以看出在做任何产品开发之前，都需要尽量考虑到以后可能的扩展或者具体业务带来的相应变化。减少对象间关联性、耦合度，提升其复用能力，对以后的维护和功能的扩展都会带来很大的帮助，否则就可能造成过多的去修改以前的设计和实现，降低了相应的开发效率。