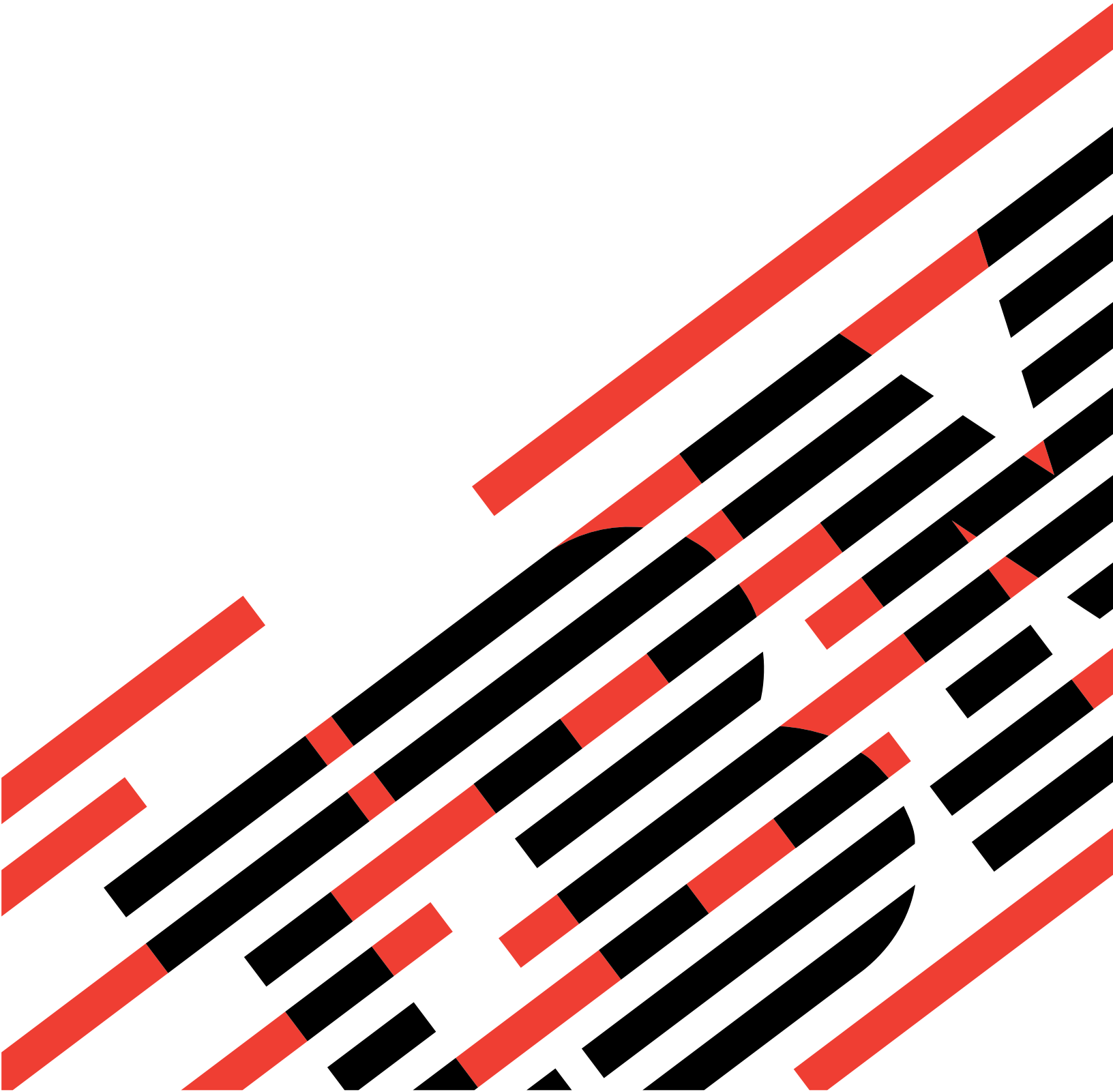
IBM

eserver

iSeries

落实控制

版本 5 发行版 3



IBM

@server

iSeries

落实控制

版本 5 发行版 3

注意

在使用本资料及其支持的产品之前，请阅读第 91 页的『声明』中的信息。

第三版（2005 年 8 月）

本版本适用于 IBM Operating System/400 V5.3.0（5722-SSI）及所有后续发行版和修订版，直到在新版本中另有声明为止。本版本不能在所有精简指令集计算机（RISC）机型上运行，也不能在 CISC 机型上运行。

目录

落实控制

落实控制具有确保数据完整性的功能。它允许您作为事务来定义和处理一组对资源（如数据库文件或表）的更改。落实控制确保在参与的所有系统上发生整组的个别更改或这些更改一个都不发生。DB2 通用数据库(TM) iSeries(TM) 版使用落实控制功能来落实和回滚数据库事务（使用非 *NONE（不落实）隔离级别运行的数据库事务）。

可以使用落实控制来设计应用程序以便作业、作业中的激活组或系统异常结束时系统可以重新启动应用程序。借助落实控制，可以保证当应用程序再次启动时，数据库中不会由于先前故障导致的未完成事务而存在部分更新。

请参阅以下信息以启动落实控制并使它在 iSeries(TM) 服务器上运行。

V5R3 的新增内容
本主题列示落实控制的新信息。

打印此主题
打印此全部信息。

落实控制概念
请阅读此信息以了解落实控制如何工作。

启动落实控制
请阅读此信息以获取有关启动落实控制的信息

结束落实控制
请阅读此信息以了解结束落实控制需要哪些先决条件以及如何结束落实控制。

系统启动的落实控制结束
请阅读当系统启动落实控制结束时您需要执行的任务。

管理事务和落实控制
请阅读需要执行来管理具有落实控制的系统的任务。

方案和示例：落实控制
请阅读这些方案和示例以了解一个公司如何设置落实控制。请阅读使用落实控制的程序的代码示例。

对事务和落实控制进行故障诊断
当需要对落实控制进行故障诊断时，请阅读此信息。

落实控制的相关信息
请参阅与落实控制有关的主题、手册、IBM(R) Redbooks(TM) 和外部 Web 站点。

注意： 请阅读代码示例免责以获取重要的法律信息。

V5R3 的新增内容

在 V5R3 中对落实控制进行了改进和补充。以下项包含改进和补充内容的摘要。

- **结束长时间运行的回滚**
  结束消耗大量处理器时间、锁定资源或占用存储器空间的长时间运行的回滚。

如何查看新增内容或已更改的内容

为帮助您查看在哪里进行了技术更改，此信息使用:

- 在新的或已更改的信息开始处具有 » 图像标记。
- 在新的或已更改的信息结束处具有 « 图像标记。

» 要找到关于此发行版的新增内容或更改内容的其它信息，请参阅用户备忘录。 «

打印此主题

要查看或下载 PDF 版本，选择《落实控制》（889 KB）。

保存 PDF 文件

要将 PDF 保存在工作站上以查看或打印:

1. 右键单击浏览器中的 PDF（右键单击上面的链接）。
2. 单击**目标另存为...**
3. 浏览至想要在其中保存 PDF 的目录。
4. 单击**保存**。

下载 Adobe Acrobat Reader

如果需要 Adobe Acrobat Reader 来查看或打印这些 PDF，可以从 Adobe Web 站点（www.adobe.com/products/acrobat/readstep.html）  下载副本。

落实控制概念

下列各页提供了一些信息来帮助您了解落实控制如何工作、落实控制如何与系统进行交互以及落实控制如何与网络中的其它系统进行交互:

- 落实控制如何工作
- 落实和回滚操作如何工作
- 落实定义
- 落实控制如何处理对象
- 落实控制和独立磁盘池
- 落实控制的注意事项和限制
- 批处理应用程序的落实控制
- 两阶段落实控制
- 落实控制的 XA 事务支持
- SQL 服务器方式和线程作用域的事务

落实控制如何工作

落实控制是一个允许您作为事务来定义和处理一组对资源（如数据库文件或表）的更改的功能。落实控制确保在参与的所有系统上发生整组的个别更改或这些更改一个都不发生。例如，当将基金从储蓄帐户转移到支出帐户时，多个更改会作为一组更改发生。对于您来说，此转移就如同单个更改。然而，对于数据库来说，就发生了多个更改，原因是同时更新了储蓄帐户和支出帐户。要保持两个帐户都准确无误，必须对支出帐户和储蓄帐户发生所有更改或不发生任何更改。

落实控制允许您:

- 确保对受影响的所有资源完成了事务中的所有更改。
- 如果处理被中断，则确保除去了事务中的所有更改。
- 当应用程序确定事务发生错误时，除去在事务期间所作的更改。

还可以设计应用程序，以便落实控制可以重新启动应用程序（如果作业、作业中的激活组或系统异常结束）。借助落实控制，可以保证当应用程序再次启动时，不会由于先前故障引起的未完成事务而在数据库中存在部分更新。

事务

事务是对系统上的对象的一组个别更改，而对用户看来则是一个极其微小的改动。

注意:

iSeries(TM) 导航器使用术语事务，而基于字符的接口使用术语“逻辑工作单元”（LUW）。这两个术语是可以互换的。除非特别指明是基于字符的接口，否则此主题使用术语事务。

事务可以是下列任何项:

- 没有在其中发生数据库文件更改的查询。
- 更改一个数据库文件的简单事务。
- 更改一个或多个数据库文件的复杂事务。
- 更改一个或多个数据库文件但这些更改只表示逻辑事务组的一部分的复杂事务。
- 涉及多个位置的数据库文件的简单或复杂事务。数据库文件可以:
  - 在单个远程系统上。
  - 在本地系统和一个或多个远程系统上。
  - 分配给本地系统上的多个日志。每个日志都可视为一个**本地位置**。
- 本地系统上涉及非数据库文件的对象的简单或复杂事务。

落实和回滚操作如何工作

有两个操作影响在落实控制下进行的更改:

- 落实操作
  落实操作使自前一落实或回滚操作以来在落实控制下执行的所有更改成为永久更改。系统还将释放与事务相关的所有锁定。
- 回滚操作
  回滚操作除去自前一落实或回滚操作以来所执行的所有更改。系统还将释放与事务相关的所有锁定。

以下编程语言和 API 支持落实和回滚操作:

语言或 API	落实	回滚
CL	COMMIT 命令	ROLLBACK 命令
ILE RPG/400(R)	COMIT 操作码	ROLBK 操作码
ILE COBOL/400(R)	COMMIT 动词	ROLLBACK 动词
ILE C/400(R)	_Rcommit 函数	_Rrollbck 函数
PL/I	PLICOMMIT 子例程	PLIROLLBACK 子例程
SQL	COMMIT 语句	ROLLBACK 语句
SQL 调用级别接口（CLI）	SQLTransact() 函数（用来落实和回滚事务）	
XA API	db2xa_commit() API	db2xa_rollback() API

下列链接提供关于这些编程语言和 API 的更多信息:

- COBOL/400 User's Guide （在 V5R1 补充手册 Web 站点上）
- RPG/400 User's Guide （在 V5R1 补充手册 Web 站点上）
- WebSphere(R) Development Studio: ILE C/C++ Programmer's Guide
- CL Programming
- API 主题
- DB2(R) UDB iSeries(TM) 版 SQL 调用级别接口（ODBC）
- 数据库编程

落实操作

落实操作使自前一落实或回滚操作以来在落实控制下执行的所有更改成为永久更改。系统还将释放与事务相关的所有锁定。

系统在接收到要落实的请求时执行下列步骤:

- 系统保存落实标识（如果提供的话）以在恢复时使用。
- 在执行落实操作之前，如果下列两个条件成立，则系统将记录写入文件:
  - 记录已添加至落实控制下的本地或远程数据库文件。
  - 当打开文件时指定 SEQONLY(*YES) 以便系统使用分块的 I/O 反馈并且记录的部分块也会存在。

  否则，不更改 I/O 反馈区域和 I/O 缓冲区。
- 系统对存在于落实定义中的每个 API 落实资源调用落实和回滚出口程序。如果某位置注册了多个出口程序，则系统按它们注册的顺序调用该位置的出口程序。
- 如果对指定给日志的资源执行任何记录更改，则系统将 C CM 日志项写入与落实定义相关联的每个本地日志。落实控制下日志项的顺序显示通常在落实定义活动时写入的项。
- 系统使暂挂的对象级别更改成为永久更改。
- 系统对已获取的并为落实控制保留的记录和对象锁定进行解锁。使那些资源可用于其他用户。
- 系统更改落实定义中的信息以显示当前事务已结束。

系统必须正确地执行前面的所有步骤才能使落实操作成功。

回滚操作

回滚操作除去自前一落实或回滚操作以来所执行的所有更改。系统还将释放与事务相关的所有锁定。系统在接收到要回滚的请求时执行下列步骤:

- 如果同时满足下列两个条件，则系统将清除 I/O 缓冲区中的记录:
  - 如果记录已添加至落实控制下的本地或远程数据库文件。
  - 如果在打开文件时指定了 SEQONLY(*YES) 以便系统使用分块的 I/O，并且存在尚未写入数据库的部分记录块。

  否则，I/O 反馈区域和 I/O 缓冲区将保持不变。
- 系统调用显示在落实定义中的每个 API 落实资源的落实或回滚出口程序。如果某位置注册了多个出口程序，则系统按与它们注册的顺序相反的顺序调用该位置的出口程序。
- 如果从文件中删除了记录，则系统将把记录添加回文件。
- 系统除去在此事务期间对记录执行的任何更改，并将原始记录（前映像）放回文件中。
- 如果在此事务期间将任何记录添加至文件，则它们将作为已删除记录保留在文件中。
- 如果在该事务期间对分配给日志的资源执行了任何记录更改，则系统将把日志项 C RB 添加至日志以指示将发生回滚操作。日志还将包含已回滚的记录更改的映像。在请求回滚操作前，已将已更改记录的前映像和后映像放入日志中。如果将任何可落实资源分配给该日志，则系统还将把 C RB 项写入缺省日志。
- 系统在下列某位置将打开的文件置于落实控制下:
  - 先前事务中最后访问的记录
  - 打开位置处（如果未使用此落实定义对该文件执行落实操作）

  如果正在进行顺序处理，则此注意事项很重要。
- 系统不会回滚数据库文件的不可落实的更改。例如，不关闭打开的文件和不恢复已清除的文件。系统不会重新打开或重新定位在此事务期间关闭的任何文件。
- 系统对用于落实控制而获取的记录锁定进行解锁并使这些记录可用于其他用户。
- 系统当前保存的落实标识与同一落实定义的最后落实操作提供的落实标识保持相同。
- 系统将撤销或回滚在此事务期间进行的对象级别的可落实更改。
- 将对用于落实控制而获取的对象锁定进行解锁并使这些对象可用于其他用户。
- 系统将先前落实边界建立为当前落实边界。
- 系统更改落实定义中的信息来显示已经结束了当前事务。

系统必须正确执行所有先前步骤才能使回滚操作成功。

落实定义

当使用“启动落实控制”（STRCMTCTL）命令来在系统上启动落实控制时将创建落实定义。另外，DB2(R) UDB iSeries(TM) 版在隔离级别不是“不落实”时自动创建落实定义。落实定义包含与在该作业中处于落实控制下正在更改的资源有关的信息。系统维护当落实资源更改时落实定义中的落实控制信息，直到落实定义结束时为止。系统上的每个活动事务由落实定义表示。后续事务可以在活动事务的每个落实或回滚之后重新使用落实定义。

落实定义通常包括:

- STRCMTCTL 命令上的参数。
- 落实定义的当前状态。
- 有关数据库文件和其它可落实资源（包含在当前事务期间所做的更改）的信息。

对于具有作业作用域锁定的落实定义，只有启动落实控制的作业了解该落实定义。其它作业都不了解该落实定义。

程序可以启动和使用多个落实定义。作业的每个落实定义标识具有与其相关联的可落实资源的独立事务。可从某些事务（与对作业启动的其它落实定义相关联的事务）独立地落实或回滚这些事务。

下面提供了有关落实定义的其它详细信息:

- 落实定义的作用域
- 落实定义名称
- 示例: 作业和落实定义

有关落实定义和独立磁盘池的规则，请参阅落实控制和独立磁盘池。

落实定义的作用域

落实定义的**作用域**确定哪些程序将使用该落实定义，以及在确定事务作用域期间如何获取锁定。启动落实定义的接口确定落实定义的作用域。对于落实定义，有四个可能的作用域，它们属于两个总的类别:

具有作业作用域锁定的落实定义

- 激活组级别落实定义
- 作业级别落实定义
- 显式命名的落实定义

具有事务作用域锁定的落实定义。

- 事务作用域落实定义

具有作业作用域锁定的落实定义只可以由运行在启动了落实定义的作业中的程序使用。相比较而言，多个作业可以使用具有事务作用域锁定的落实定义。

应用程序通常使用激活组级别或作业级别的落实定义。这些落实定义是使用“启动落实控制”（STRCMTCTL）命令显式创建的，或由系统在 SQL 应用程序以隔离级别（*NONE 除外）运行时隐式创建的。

激活组级别落实定义

最常见的作用域是激活组。当 STRCMTCTL 命令显式启动落实定义时，或者以隔离级别（“不落实”除外）运行的 SQL 应用程序隐式启动落实定义时，激活组级别落实定义将是缺省作用域。只有在该激活组内运行的程序可以使用该落实定义。对于作业，同时可以有许多激活组级别落实定义是活动的。但是，每个激活组级别落实定义只可以与单个激活组相关联。在该激活组内运行的程序只可将它们的可落实更改与该激活组级别落实定义相关联。

当 iSeries(TM) 导航器、使用落实定义（WRKCMTDFN）命令、显示作业（DSPJOB）命令或使用作业（WRKJOB）命令显示激活组级别落实定义时，这些字段将显示下列内容:

- 落实定义字段显示激活组的名称。它显示特殊值 *DFTACTGRP 以指示缺省激活组。
- 激活组字段显示激活组号。
- 作业字段显示启动落实定义的作业。
- 线程字段显示 *NONE。

作业级别落实定义

只可以通过发出 STRCMTCTL CMTSCOPE(*JOB) 来将落实定义的作用域限定为作业。如果作业级别的落实定义已经由该作业的另一程序启动，则任何运行在没有启动激活组级别落实定义的激活组中的程序都可以使用该作业级别落实定义。您只可以为作业启动单个作业级别落实定义。

当 iSeries 导航器、使用落实定义（WRKCMTDFN）命令、显示作业（DSPJOB）命令或使用作业（WRKJOB）命令显示作业级别落实定义时，这些字段将显示下列内容:

- 落实定义字段显示特殊值 *JOB。
- 激活组字段显示空白。
- 作业字段显示启动落实定义的作业。
- 线程字段显示 *NONE。

对于给定的激活组，运行在该激活组内的程序只能使用单个落实定义。因此，在激活组内运行的程序可以使用作业级别或激活组级别落实定义，但不能同时使用这两个落实定义。在不使用 SQL 服务器方式的多线程作业中，将把程序的事务工作的作用域限定为关于该程序激活组的适当落实定义，而不管哪个线程执行它。如果多个线程使用相同的激活组，则它们必须进行合作来执行事务性工作并确保在正确的时间发生落实和回滚。

即使当作业级别落实定义对于该作业是活动的时候，如果该激活组内运行的程序没有对该作业级别落实定义执行任何落实控制请求或操作，则该程序仍可以启动激活组级别落实定义。否则，必须首先结束作业级别落实定义，才能启动激活组级别落实定义。可以防止启动激活组级别落实定义的作业级别落实定义的落实控制请求或操作包括:

- 在落实控制下打开（完全或共享）数据库文件。
- 使用“添加落实资源”（QTNADDCR）API 来添加 API 落实资源。
- 落实事务。
- 回滚事务。
- 在落实控制下添加远程资源。
- 使用“更改落实选项”（QTNCHGCO）API 来更改落实选项。
- 使用“必需回滚”（QTNRBRQD）API 来使落实定义处于必需回滚状态。
- 通过使用具有“包括落实周期标识”参数的“发送日志项”（QJOSJRNE）API 来发送包括当前落实周期标识的用户日志项。

同样，如果激活组内的程序当前正在使用激活组级别落实定义，则必须首先结束落实定义，这样运行在同一激活组内的程序才能使用作业级别的落实定义。

当打开数据库文件时，已打开文件的打开作用域可以为激活组或具有一个限制的作业: 如果程序正在落实控制下打开文件并且该文件的作用域限定为该作业，则执行打开请求的程序必须使用作业级别的落实定义。

显式命名的落实定义

当系统需要执行自己的落实控制事务而不影响应用程序使用的任何事务时，系统将启动显式命名的落实定义。启动这些类型的落实定义的功能示例是问题记录。应用程序不能启动显式命名的落实定义。

当 iSeries 导航器、使用落实定义（WRKCMTDFN）命令、显示作业（DSPJOB）命令或使用作业（WRKJOB）命令显示显式命名的落实定义时，这些字段将显示下列内容:

- 落实定义字段显示系统提供给它的名称。
- 激活组字段显示空白。
- 作业字段显示启动落实定义的作业。
- 线程字段显示 *NONE。

事务作用域落实定义

事务作用域落实定义由“事务作用域锁定”的 XA API 启动。

这些 API 使用基于线程或基于 SQL 连接并且不基于激活组的落实控制协议。换句话说，这些 API 用于在执行事务性工作时将落实定义与特定线程或 SQL 连接相关联，并落实或回滚事务。根据 API 协议，系统将这些落实定义附加至执行事务性工作的线程。它们可以由不同作业中的线程使用。

当 iSeries 导航器、使用落实定义（WRKCMTDFN）命令、显示作业（DSPJOB）命令或使用作业（WRKJOB）命令显示事务作用域落实定义时，这些字段将显示下列内容:

- 落实定义字段显示特殊值 *TNSOBJ。
- 激活组字段显示空白。
- 作业字段显示启动落实定义的作业。
- 线程字段显示向其附加落实定义的线程（或者如果落实定义当前没有附加至任何线程，则显示 *NONE）。

落实定义名称

系统提供对作业启动的所有落实定义的名称。下表显示特定作业的各种落实定义及其相关联的名称。

激活组	落实作用域	落实定义名称
任何	作业	*JOB
缺省激活组	激活组	*DFTACTGRP
用户命名的激活组	激活组	激活组名（例如，PAYROLL）
系统命名的激活组	激活组	激活组号（例如，0000000145）
无	显式命名的	QDIR001（系统定义的落实定义的示例，仅供系统使用）。系统定义的落实定义名称以 Q 开头。
无	事务	*TNSOBJ

只有 Integrated Language Environment(R)（ILE）编译的程序可以对缺省激活组之外的激活组启动落实控制。因此，仅当作业正在运行一个或多个 ILE 编译的程序时，作业可以使用多个落实定义。有关 Integrated Language Environment(R) 的更多信息，请参阅信息中心的“编程”主题。

原始程序模型（OPM）程序在缺省激活组中运行，并且在缺省情况下它使用 *DFTACTGRP 落实定义。在混合的 OPM 和 ILE 环境中，如果要一起落实或回滚所有程序执行的所有可落实更改，则作业必须使用作业级别的落实定义。

其作用域限定为激活组的已打开的数据库文件可与激活组级别或作业级别落实定义相关联。其作用域限定为作业的已打开的数据库文件只能与作业级别落实定义相关联。因此，在落实控制下打开数据库文件（其作用域限定为作业）的任何程序（OPM 或 ILE）需要使用作业级别的落实定义。

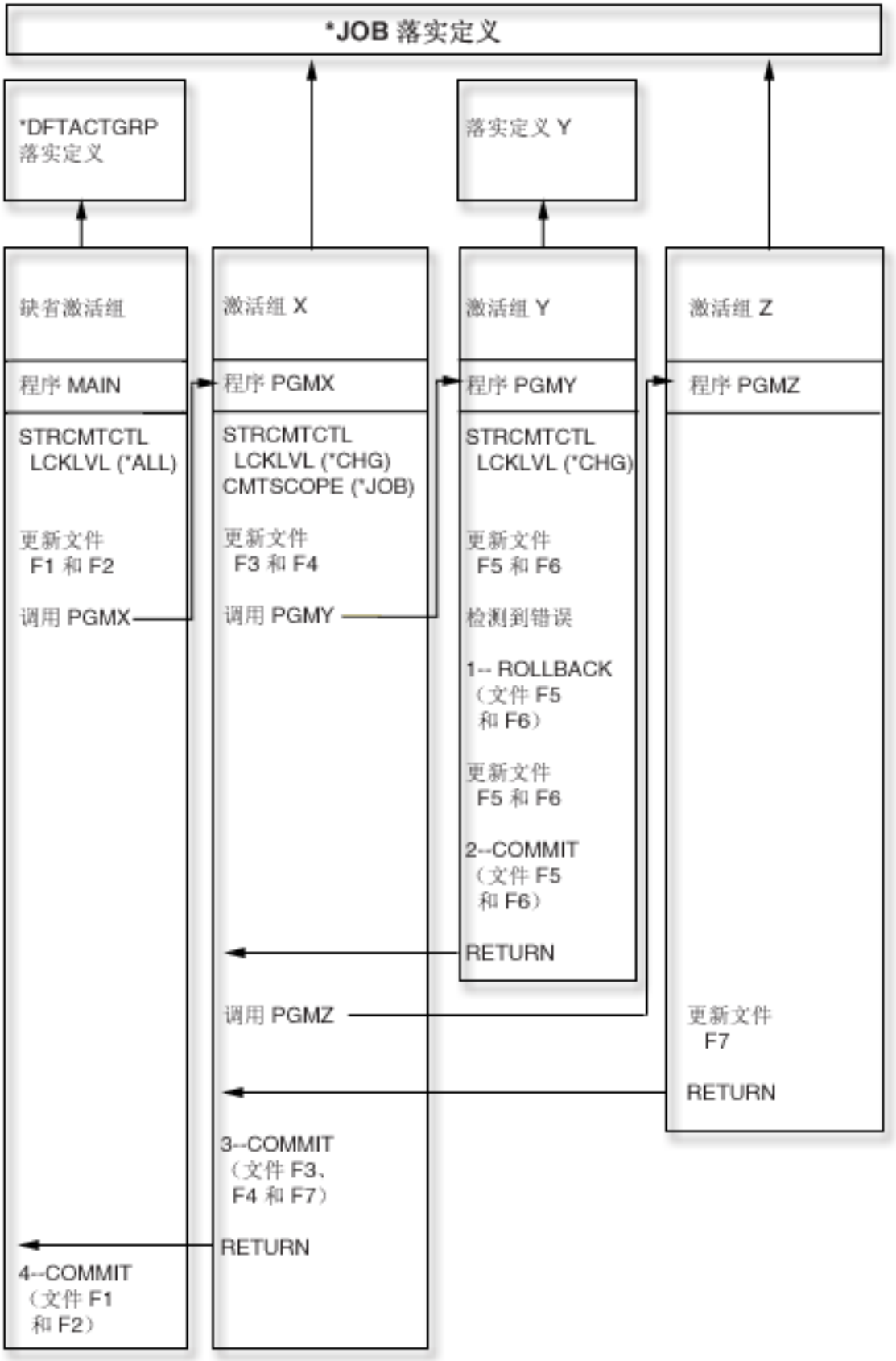
当应用程序执行落实控制请求时不使用落实定义名称来标识特定落实定义。落实定义名称主要在消息中使用来标识作业的特定落实定义。

对于激活组级别的落实定义，系统根据请求的程序正在哪个激活组中运行来确定要使用哪个落实定义。这可能是因为在任何时间点运行于激活组中的程序只可以使用单个落实定义。

对于具有事务作用域锁定的事务，添加至 CLI 的 XA API 和与事务相关的属性确定调用的线程使用哪个落实定义。

示例: 作业和落实定义

下图显示使用多个落实定义的作业的示例。它指示以每个激活组级别落实或回滚哪些文件更新。该示例假定所有程序对数据库文件执行的所有更新都是在落实控制下执行的。


*JOB 落实定义
*DFTACTGRP
落实定义
落实定义 Y
缺省激活组
激活组 X
激活组 Y
激活组 Z
程序 MAIN
程序 PGMX
程序 PGMY
程序 PGMZ
STRCMTCTL
LCKLVL (*ALL)
更新文件
F1 和 F2
调用 PGMX
STRCMTCTL
LCKLVL (*CHG)
CMTSCOPE (*JOB)
更新文件
F3 和 F4
调用 PGMY
STRCMTCTL
LCKLVL (*CHG)
更新文件
F5 和 F6
检测到错误
1-- ROLLBACK
（文件 F5
和 F6）
更新文件
F5 和 F6
2--COMMIT
（文件 F5
和 F6）
RETURN
调用 PGMZ
更新文件
F7
RETURN
3--COMMIT
（文件 F3、
F4 和 F7）
RETURN
4--COMMIT
（文件 F1
和 F2）

链接至描述

下表显示如果前一幅图中的方案发生更改，如何落实或回滚文件:

作业中多个落实定义的其它示例

方案中的更改	更改对下列文件的影响:			
	F1 和 F2	**F3 和 F4**	**F5 和 F6**	**F7**
PGMX 执行回滚操作而不是落实操作（3==COMMIT 变成 ROLLBACK）。	仍然暂挂	回滚	已落实	回滚
在返回到 PGMX 之前，PGMZ 执行落实操作。	仍然暂挂	由 PGMZ 落实	已落实	落实
在更新了文件 F7 之后，PGMZ 尝试启动指定 CMTSCOPE(*ACTGRP) 的落实控制。该尝试失败，原因是使用作业级别落实定义暂挂了更改。	仍然暂挂	仍然暂挂	已落实	仍然暂挂
PGMX 不启动落实控制并且不用 COMMIT(*YES) 打开文件 F3 和 F4。PGMZ 尝试用 COMMIT(*YES) 打开文件 F7。	仍然暂挂	不在落实控制之下	已落实	不能打开文件 F7，原因是不存在 *JOB 落实定义（PGMX 未创建它）。

落实控制如何处理对象

当将对象置于落实控制之下时，它将成为可落实资源。它将使用落实定义进行注册。它参与针对该落实定义发生的每个落实操作和回滚操作。

下面的主题描述可落实资源的这些属性:

- 资源类型
- 位置
- 落实协议
- 访问意向

以下链接具有关于落实控制下的资源的更多信息:

- 可落实资源的类型
- 远程和本地可落实资源
- 可落实资源的访问意向
- 可落实资源的落实协议
- 记录的文件和落实控制
- 落实控制下日志项的顺序

- 落实周期标识
- 记录锁定

可落实资源的类型

下表显示:

- 可落实资源的类型。
- 如何将它们置于落实控制下。
- 如何从落实控制中除去它们。
- 适用于资源类型的限制。

资源类型	如何将其置于落实控制下	如何从落实控制中除去它	哪类更改是可落实的	限制
FILE- 本地数据库文件	在落实控制下打开[1]	如果没有暂挂的更改，则关闭文件。 如果关闭文件时有暂挂的更改，则在执行下一个落实或回滚操作后关闭。	记录级别更改	对于单个事务，锁定的记录不能多于 500 000 000 条[2]。
DDL- 对本地 SQL 表和 SQL 集合的对象级别更改。	在落实控制下运行 SQL	在对象级别更改后执行落实或回滚操作。	对象级别更改，如: • 创建 SQL 包 • 创建 SQL 表 • 删除 SQL 表	只有使用 SQL 执行的对象级别更改处于落实控制之下。
DDM- 远程分布式数据管理（DDM）文件	在落实控制下打开。DDM 的落实控制支持具有关于落实控制和分布式数据管理的更多信息。	如果没有暂挂的更改，则关闭文件。 如果关闭文件时有暂挂的更改，则在执行下一个落实或回滚操作后关闭。	记录级别更改	
LU 6.2- 受保护对话	启动对话[3]	结束对话		
DRDA(R)- 分布式关系数据库	使用 SQL CONNECT 语句	结束连接		
API- 本地 API 落实资源	添加落实资源（QTNADDCR）API	除去落实资源（QTNRMVCR）API	用户程序可以确定这一点。可由用户程序使用发送日志项（QJOSJRNE）API 编写日志项来帮助跟踪这些更改。	应用程序必须落实出口程序以在落实、回滚或再同步操作期间调用它。
TCP-TCP/IP 连接	对已定义要使用 TCP/IP 连接的 RDB 使用 SQL CONNECT 语句，或打开以 TCP/IP 位置定义的 DDM 文件	如果没有暂挂的更改，结束 SQL 连接，或关闭 DDM 文件。如果在具有暂挂的更改情况下关闭 DDM 文件，则将在执行下一个落实或回滚操作之后关闭连接。		

资源类型	如何将其置于落实控制下	如何从落实控制中除去它	哪类更改是可落实的	限制
注意: [1]关于如何将数据库文件置于落实控制下的详细信息，请参阅适当的语言参考手册。落实控制的相关信息含有某些您可以使用的语言手册的链接。 [2]可以使用 QAQQINI 文件来减少对最多 500 000 000 条记录的限制。请参阅管理事务大小以获取指示信息。 [3]当启动 DDM 连接时，DDM 文件将指定 PTCCNV(*YES) 并且将以 SNA 远程位置定义 DDM 文件，DDM 资源和 LU6.2 资源一起添加。 启动 DRDA 连接时，如果同时满足下列两个条件，则 DRDA 资源和 LU6.2 资源一起添加: • 程序正在使用分布式工作单元连接协议。 • 连接的是以 SNA 远程位置定义的 RDB。有关启动受保护对话的更多信息，请参阅 APPC Programming。				

本地和远程可落实资源

可落实资源可以是本地资源，也可以是远程资源。

本地可落实资源

本地可落实资源驻留在应用程序所在的系统上。可将落实控制下与资源相关联的每个日志视为本地位置。已注册但不具有日志的所有资源（任意 DDL 资源和 API 资源）都可视为独立的本地位置。

如果可落实资源驻留在独立磁盘池上，而落实定义驻留在另一个磁盘池上，则不认为该资源是本地的。有关可落实资源和独立磁盘池的更多信息，请参阅落实控制和独立磁盘池。

远程可落实资源

远程可落实资源与应用程序驻留在不同的系统上。远程系统的每个唯一对话都存在远程位置。落实定义可能在一个或多个远程系统上具有一个或多个远程位置。

当您将本地资源置于系统磁盘池或任何独立磁盘池的落实控制之下时，必须使用 DRDA(R) 来访问任何其它独立磁盘池中的落实控制之下的资源。

下面显示可落实资源的类型及其位置:

资源类型	位置
文件	本地
DDL	本地
API	本地
DDM	远程
LU62	远程
DRDA	本地或远程
TCP	远程

可落实资源的访问意向

当资源处于落实控制之下时，资源管理器会指示将如何访问资源:

- 更新
- 只读
- 未确定

访问意向确定资源如何一起参与事务。下表显示对于特定类型的资源哪些访问意向是可能的以及系统在注册资源时如何确定资源的访问意向:

资源类型	可能的访问意向	如何确定访问意向
文件	更新，只读	取决于打开文件的方式
DDL	更新	始终更新
API	更新	始终更新
DDM	更新，只读	取决于打开文件的方式
LU62	未确定	始终未确定
DRDA(R)	更新，只读，未确定	对于 DRDA 级别 1，如果没有注册其它远程资源，则访问意向是更新的。否则，访问意向是只读的。对于 DRDA 级别 2，访问意向始终未确定。
TCP	未确定	始终未确定

已注册的资源的访问意向确定是否可以注册新资源。以下规则适用:

- 当出现下列情况时，不能注册其访问意向是更新的一阶段资源:
  - 已在其它位置注册了其访问意向是更新的资源。
  - 已在其它位置注册了未确定其访问意向的资源。
  - 已在同一位置注册了未确定其访问意向的资源并且在当前事务期间更改了这些资源。
- 当已注册其访问意向是更新的一阶段资源时，不能注册其访问意向是更新的两阶段资源。

可落实资源的落实协议

落实协议是资源具有的参与一阶段或两阶段落实处理的能力。本地资源（API 可落实资源除外）通常是两阶段资源。

如果可落实资源驻留在独立磁盘池中，并且如果落实定义驻留在另一磁盘池上，则不认为该资源为本地资源或两阶段资源。有关可落实资源和独立磁盘池的更多信息，请参阅落实控制和独立磁盘池。

两阶段资源也称为**受保护资源**。当将远程资源和 API 可落实资源置于落实控制之下时，必须将它们注册为一阶段资源或两阶段资源。下表显示哪种类型的可落实资源可以与一阶段资源在落实定义中共存:

资源类型	可以与以下项共存
一阶段 API 资源	其它本地资源。无远程资源。
一阶段远程资源	其它在同一位置的一阶段资源。无本地资源。

记录的文件和落实控制

必须记录数据库文件（资源类型为 FILE 或 DDM）才能在落实控制下打开它以用于输出或由使用 “不落实” 之外的隔离级别的 SQL 应用程序引用它。文件只有在落实控制下为了输入而打开时才不需要进行记录。如果存在下列情况，则会发生错误:

- 尝试在落实控制下打开数据库文件以进行输出，但该文件当前未记录。
- 没有启动可由正在落实控制下打开的文件使用的落实定义。

如果在落实控制下打开数据库文件时仅在记录该文件的后映像，则系统自动开始记录前映像和后映像。仅对在落实控制下发生的文件的更改编写前映像。如果同时对该文件发生不在落实控制下的其它更改，则仅对这些更改编写后映像。

系统自动将记录级别可落实更改和对象级别可落实更改写入日志。对于记录级别更改，系统将日志项用于恢复（若有必要）；系统不会将对象级别可落实更改的项用于恢复。另外，系统不会自动编写 API 落实资源的日志项。但是，API 资源的出口程序可使用“发送日志项”（QJOSJRNE）API 编写日志项以提供审计跟踪或帮助恢复。这些项的内容由用户出口程序控制。

系统使用日志以外的方法来执行对象级别落实资源的恢复。通过调用与每个特定 API 落实资源相关联的落实和回滚出口程序来完成 API 落实资源的恢复。出口程序负责执行该状况所必需的实际恢复。

有关日志记录的更多信息，请参阅日志管理主题。

落实控制下日志项的顺序

下表显示通常在落实定义活动时编写的项的顺序。可以使用日志项信息查找程序来获取关于日志项内容的更多信息。

如果至少下列其中一项成立时，将把落实控制项写入本地日志:

- 在使用启动落实控制（STRCMTCTL）命令时，将日志指定为缺省日志。
- 至少一个记录至日志的文件在落实控制下打开。
- 至少一个与日志相关联的 API 落实资源是在落实控制下注册的。

项类型	描述	写入的位置	写入的时间
C BC	启动落实控制	如果在 STRCMTCTL 命令上指定了缺省日志，则将其写入缺省日志	当使用 STRCMTCTL 命令时。
		写入日志。	当打开记录至日志的第一个文件时，或当为日志注册 API 资源时。
C SC	启动落实周期	写入日志。	当对于记录至此日志[1]的文件的事务发生第一条记录更改时。
		写入 API 资源的日志。	当将 QJOSJRNE API 第一次与*包含落实周期标识*键一起使用时。
日志码 D 和 F	DDL 对象级别项	写入与正在更新的对象相关联的日志。只有包含落实周期标识的日志项表示属于事务一部分的 DDL 对象级别更改。	当更新发生时。
日志码 R	记录级别项	写入与正在更新的文件相关联的日志。	当更新发生时。
日志码 U	用户创建的项	写入与 API 资源相关联的日志	如果首先将使用 QJOSJRNE API 的应用程序与*包含落实周期标识*键一起使用时。
C CM	落实	写入日志。	当落实成功完成时。
		写入缺省日志。	如果任何可落实资源与日志相关联时。
C RB	回滚	写入日志。	在完成了回滚操作后。
		写入缺省日志。	如果任何可落实资源与日志相关联时。

项类型	描述	写入的位置	写入的时间
C LW	结束事务	如果在 STRCMTCTL 命令上指定了缺省日志，则将其写入缺省日志。系统写入 LW 头记录以及一个或多个详细记录。只有在 STRCMTCTL 命令上指定了 OMTJRNE(*NONE) 或者发生系统错误时，才会写入这些项。	当完成落实或回滚操作时。
C EC	结束落实控制	写入日志。	当完成结束落实控制（ENDCMTCTL）命令时。
		写入不是缺省日志的本地日志。	当建立落实边界时，也就是从落实控制中除去与该日志相关联的所有可落实资源之后。
C SB	保存点或嵌套落实周期的开始。	写入日志。	当应用程序创建 SQL SAVEPOINT 时，或者当系统创建内部嵌套落实周期以将一系列数据库函数作为单个操作[2]处理时。
C SQ	释放保存点或落实嵌套落实周期。	写入日志。	当应用程序释放 SQL SAVEPOINT 时，或者当系统落实内部嵌套落实周期[2]时。
C SU	回滚保存点或嵌套落实周期。	写入日志。	当应用程序回滚 SQL SAVEPOINT 时，或者当系统回滚内部嵌套落实周期[2]时。

注意:

[1]可以通过指定“创建日志”（CRTJRN）或“更改日志”（CHGJRN）命令的“固定长度数据”（FIXLENDTA）参数的“逻辑工作单元”（*LUW）值，来指定日志项的固定长度部分包括事务信息。通过指定 FIXLENDTA (*LUW) 参数，每个 C SC 日志项的固定长度部分将包含当前事务的逻辑工作单元标识（LUWID）。同样对于 XA 事务，如果指定 FIXLENDTA(*XID) 参数，则每个 C SC 日志项的固定长度部分将包含当前事务的 XID。如果事务中涉及多个日志或系统，则 LUWID 或 XID 可以帮助您查找特定事务的所有落实周期。

[2]只有将 QTN_JRNSAVPT_MYLIB_MYJRN 环境变量设置为 *YES 时才会发送这些项，其中 MYJRN 是正在使用的日志，并且 MYLIB 是日志存储于其中的库。对于 MYLIB 和 MYJRN 值，特殊值 *ALL 受支持。可以设置这些系统范围变量或为特定作业设置这些变量。 要仅对一个作业的日志 MYLIB/MYJRN 发送项，则在该作业中使用此命令:

- ADDENVVAR ENVVAR(QTN_JRNSAVPT_MYLIB_MYJRN) VALUE(*YES)

要对所有作业的所有日志发送项:

- ADDENVVAR ENVVAR('QTN_JRNSAVPT_*ALL_*ALL') VALUE(*YES) LEVEL(*SYS)

在启动落实控制前，您需要设置环境变量。

落实周期标识

落实周期是从一个落实边界到下一个落实边界之间的时间。系统将分配**落实周期标识**以使某特定落实周期的所有日志项关联起来。参与事务的每个日志都有它自己的落实周期和它自己的落实周期标识。

落实周期标识是对落实周期所写的 C SC 日志项的日志序号。将落实周期标识放入在落实周期期间所写的每个日志项中。如果在落实周期期间使用了多个日志，则每个日志的落实周期标识各不相同。

通过指定“创建日志”（CRTJRN）或“更改日志”（CHGJRN）命令的“固定长度数据”（FIXLENDTA）参数的“逻辑工作单元”（*LUW）值，可以指定日志项的固定长度部分包括事务信息。通过指定 FIXLENDTA（*LUW）参数，每个 C SC 日志项的固定长度部分将包含当前事务的逻辑工作单元标识（LUWID）。同样，对于 XA 事务，如果指定 FIXLENDTA (*XID) 参数，则每个 C SC 日志项的固定长度部分将包含当前事务的 XID。如果事务中涉及多个日志或系统，则 LUWID 或 XID 可以帮助您查找特定事务的所有落实周期。

可以使用“发送日志项”（QJOSJRNE）API 来编写 API 资源的日志项。可选择包括那些日志项的落实周期标识。

通过使用“应用已记录的更改”（APYJRNCHG）命令或“除去已记录的更改”（RMVJRNCHG）命令，可使用落实周期标识来应用或除去对落实边界已记录的更改。存在下列限制:

- 在落实控制下执行的大多数对象级别的更改都写入日志，但不会使用 APYJRNCHG 和 RMVJRNCHG 命令应用或除去这些更改。
- QJOSJRNE API 编写具有日志码 U 的用户创建的日志项。不能使用 APYJRNCHG 和 RMVJRNCHG 命令应用或除去这些项。必须使用用户编写的程序应用或除去它们。

记录锁定

当作业持有记录锁定并且另一作业尝试检索该记录以进行更新时，请求作业将等待并将从活动的处理中除去它直到下列某种情况发生:

- 释放记录锁定。
- 指定的等待时间结束。

多个作业可以请求由另一作业锁定的记录。当释放记录锁定时，第一个请求记录的作业将接收该记录。当等待锁定的记录时，在下列创建、更改或覆盖命令上的 WAITRCD 参数中指定等待时间:

- 创建物理文件（CRTPF）
- 创建逻辑文件（CRTLF）
- 创建源物理文件（CRTSRCPF）
- 更改物理文件（CHGPF）
- 更改逻辑文件（CHGLF）
- 更改源物理文件（CHGSRCPF）
- 覆盖数据库文件（OVRDBF）

当指定等待时间时，请考虑以下内容:

- 如果不指定值，则程序等待的时间是进程的缺省等待时间。
- 对于只具有事务作用域锁定的落实定义，在下列项上指定的事务锁定等待时间可以覆盖作业缺省等待时间:
  - xa_open API。
  - JDBC 或 JTA 接口。分布式事务列示这些 API。
- 如果不能在指定的时间内分配记录，则将把通知消息发送给高级语言程序。
- 如果超出了记录的等待时间，则发送给作业记录的消息将给出持有锁定的记录（导致请求的作业等待的记录）的作业名称。如果遇到了记录锁定异常，则可以使用作业记录来帮助确定改变哪些程序以便它们不会长时间持有锁定。

程序由于下列某原因长时间持有记录锁定:

- 当工作站用户正在考虑更改时，记录将保持锁定。
- 记录锁定是长时间落实事务的一部分。请考虑执行较小的事务以便可以更频繁地执行落实操作。
- 发生了意外的锁定。例如，假定文件定义为具有唯一键的更新文件，并且程序更新并向文件添加附加记录。如果工作站用户想将记录添加至文件，则程序可能尝试访问记录以确定该键是否已存在。如果存在的话，则程序将通知工作站用户所执行的请求无效。当从文件中检索记录时，将锁定该记录直到对同一文件的另一读操作隐式释放它为止，或直到显式释放它为止。

注意:

有关如何使用每个高级语言接口以释放记录锁定的更多信息，请参阅适当的高级语言参考手册。落实控制的相关信息含有某些您在使用落实控制时可参考的高级语言手册的链接。

如果指定了 LCKLVL(*ALL)，则锁定的持续时间将更长，因为从文件检索的记录将保持锁定直到下一个落实或回滚操作为止。它不会由另一读操作隐式释放并且也不能显式释放它。

可将锁定放置在文件上的另一个功能是活动时保存功能。当服务器活动时保存它主题具有关于活动时保存功能的更多信息。

落实控制和独立磁盘池

独立磁盘池和独立磁盘池组可分别具有独立的 OS/400(R) SQL 数据库。可以将落实控制与这些数据库一起使用。然而，由于每个独立磁盘池或独立磁盘池组都具有独立的 SQL 数据库，所以要遵循下列注意事项。

- 落实定义的独立磁盘池注意事项
- XA 事务的独立磁盘池注意事项

落实定义的独立磁盘池注意事项

当使用独立磁盘池时，您必须了解以下落实定义的注意事项。

QRECOVERY 库注意事项

当启动落实控制时，就会在 QRECOVERY 库中创建落实定义。每个独立磁盘池或独立磁盘池组都具有它自己的 QRECOVERY 库版本。在独立磁盘池上，QRECOVERY 库的名称是 QRCYxxxxx，其中 xxxxx 是独立磁盘池的编号。例如，独立磁盘池 39 的 QRECOVERY 库的名称是 QRCY00039。另外，如果独立磁盘池是磁盘池组的一部分，则只有主磁盘池才有 QRCYxxxxx 库。

当启动落实控制时，就会在与该作业相关联的独立磁盘池的 QRECOVERY 库中创建落实定义，使落实控制在独立磁盘池上是活动的。

设置 ASP 组注意事项

当落实控制在独立磁盘池上是活动的时候使用“设置 ASP 组”（SETASPGRP）命令具有下列影响:

- 如果您从独立磁盘池切换并且在磁盘池上以落实控制注册了资源，则 SETASPGRP 命令失败，消息为 CPDB8EC，原因码为 2，“线程具有未落实的事务”。此消息后跟消息 CPFB8E9。
- 如果您从独立磁盘池切换并且没有以落实控制注册资源，则将落实定义移至您正切换至的独立磁盘池。
- 如果您从系统磁盘池（ASP 组 *NONE）切换，则不影响落实控制。落实定义保留在系统磁盘池上。
- 如果您使用通知对象，则该通知对象必须驻留在落实定义所在的同一独立磁盘池或独立磁盘池组上。

- 如果将落实定义移至另一个独立磁盘池或独立磁盘池组，则通知对象也必须驻留在该其它独立磁盘池或独立磁盘池组上。如果落实定义异常结束，则会更新该其它独立磁盘池或独立磁盘池组上的通知对象。如果在该其它独立磁盘池或独立磁盘池组上找不到此通知对象，则更新会失败，消息为 CPF8358。

缺省日志注意事项

以下是缺省日志注意事项:

- 如果您使用缺省日志，则该日志必须驻留在落实定义所在的同一独立磁盘池或独立磁盘池组上。
- 如果在落实控制启动时在这一其它独立磁盘池或独立磁盘池组上找不到缺省日志，则落实控制启动会失败，消息为 CPF9873。
- 如果将落实定义移至另一个独立磁盘池或独立磁盘池组，则缺省日志也必须驻留在该其它独立磁盘池或独立磁盘池组上。如果在该其它独立磁盘池或独立磁盘池组上找不到日志，则会移动落实定义，但从此之后不再使用缺省日志。

IPL 和脱机注意事项

以下是 IPL 和脱机注意事项:

- 恢复驻留在独立磁盘池上的落实定义是在独立磁盘池联机处理期间执行的，它类似于 IPL 恢复。
- 在系统 IPL 期间不会恢复独立磁盘池中的落实定义。
- 使独立磁盘池脱机对落实定义具有以下影响:
  - 与独立磁盘池相关联的作业结束。
  - 不允许在独立磁盘池上创建新的落实定义。
  - 驻留在独立磁盘池的落实定义变得不可用。
  - 驻留在独立磁盘池上但未附加至作业的落实定义释放事务作用域锁定。

远程数据库注意事项

以下是远程数据库注意事项:

- 不能使用 LU6.2 SNA 连接（受保护对话或分布式工作单元（DUW））来从独立磁盘池数据库连接至远程数据库。可以使用不受保护的 SNA 对话来从独立磁盘池数据库连接至远程数据库。
- 当落实控制对于作业或线程是活动的时候，访问落实定义所属的独立磁盘池或磁盘池组外部的数据只可能远程进行，就好像这些数据是驻留在另一个系统上一样。当发出 SQL CONNECT 语句以连接至独立磁盘池上的关系数据库（RDB）时，系统使该连接成为远程连接。
- 系统磁盘池和基本磁盘池不需要远程连接就可对驻留在独立磁盘池上的数据进行只读访问。同样，独立磁盘池不需要远程连接就可对驻留在系统磁盘池或基本磁盘池上的数据进行只读访问。

XA 事务的注意事项

在 XA 环境中，将每个数据库视为一个独立的资源管理器。当事务管理器想访问同一事务下的两个数据库时，它必须使用 XA 协议来对两个资源管理器执行两阶段落实。

由于每个独立磁盘池都是一个独立的 SQL 数据库，所以在 XA 环境中，也将每个独立磁盘池视为一个独立的资源管理器。对于要执行事务（目标为两个不同的独立磁盘池的事务）的应用程序服务器，事务管理器也必须使用两阶段落实协议。

有关独立磁盘池的更多信息，请参阅独立磁盘池主题。

落实控制的注意事项和限制

以下为落实控制的各种注意事项和限制:

数据库文件注意事项

- 如果指定在落实控制下打开共享文件，则对该文件的所有后续使用都必须在落实控制下打开。
- 如果为使用 LCKLVL(*ALL) 进行只读（隐式或通过高级语言程序，或者通过使用数据库文件覆盖（OVRDBF）命令）而打开的文件指定了 SEQONLY(*YES)，则将忽略 SEQONLY(*YES) 并且使用 SEQONLY(*NO)。
- 在落实控制下进行的记录级别的更改记录在日志中。可以通过应用已记录的更改（APYJRNCHG）命令或除去已记录的更改（RMVJRNCHG）命令将这些更改应用至数据库或从数据库中除去。
- 在落实控制下记录文件的前映像和后映像。如果指定只记录文件的后映像，则系统还将自动记录在落实控制下发生的文件更改的前映像。但是，因为没有为对文件所作的所有更改捕获前映像，所以您不能对这些文件使用 RMVJRNCHG 命令。

对象和记录级别更改的注意事项

-
- 在落实控制下使用 SQL 进行的对象级别和记录级别的更改将使用当前对于激活组（请求程序正运行于该激活组中）是活动的落实定义。如果作业级别或激活组级别落实定义都不是活动的，则 SQL 将启动激活组级别落实定义。有关在落实控制下使用 SQL 进行的更改的更多信息，请参阅 SQL 编程概念主题。

一阶段和两阶段落实注意事项

- 当建立一阶段远程对话或连接时，将不允许对其它位置的远程对话或连接。如果建立了落实边界并且除去了所有资源，则可以更改该位置。
- 如果正在使用两阶段落实，则不需要使用提交远程命令（SBMRMTCMD）命令来在远程位置启动落实控制或执行任何其它落实控制操作。系统将为您执行这些功能。
- 对于一阶段远程位置，如果 SQL 在调用堆栈中并且远程关系数据库不在系统上，则 COMMIT 和 ROLLBACK CL 命令将失败。如果 SQL 不在调用堆栈上，则 COMMIT 和 ROLLBACK 命令不会失败。
- 对于一阶段远程位置，在对远程资源进行可落实的更改之前，必须在源系统上启动落实控制。如果 SQL 程序正在以除 *NONE 之外的落实控制选项运行，则系统将在连接时对源系统上的分布式数据库 SQL 自动启动落实控制。当将第一个远程资源放在落实控制之下时，系统将在目标系统上启动落实控制。

保存注意事项

如果执行保存操作的作业具有一个或多个进行过下列任何类型的可落实更改的活动落实定义，则将不能进行保存操作:

- 对驻留在正在保存的库中的文件进行的记录更改。对逻辑文件，将检查所有相关的物理文件。
- 正在保存的库中的任何对象级别的更改。
- 使用添加落实资源（QTNADDCR）API 以及在“允许正常保存处理”字段设置为缺省值 N 的情况下添加的任何 API 资源。

这将阻止保存操作将由于部分事务而产生的更改保存至保存介质。

» **注意**: 如果以部分事务功能部件使用新的保存，则可以不结束落实定义而保存对象。 «

对象锁定和记录锁定防止将其它作业中的落实定义的暂挂更改保存至保存介质。若当对与 API 落实资源相关联的对象进行更改时获取了锁定，则这一点仅对于 API 落实资源成立。

其它注意事项和限制

- 必须完成或取消所有暂挂的再同步，才能将系统升级至新的发行版。有关更多详细信息，请参阅安装软件前，确保两阶段落实完整性主题。
- 在落实或回滚期间，COMMIT 和 ROLLBACK 值显示在“WRKACTJOB 函数”字段中。如果该函数长时间保持 COMMIT 或 ROLLBACK，则可能发生了下列一种情况:
  - 落实或回滚期间的资源故障要求再同步。直到再同步完成或取消时，才会将控制返回给应用程序。
  - 此系统在落实期间执行只读表决。直到启动落实的系统将数据发送给此系统时，才会将控制返回给应用程序。
  - 此系统在落实期间表决“确认以忽略”。直到启动落实的系统将数据发送给此系统时，才会将控制返回给应用程序。

批处理应用程序的落实控制

批处理应用程序可能需要也可能不需要落实控制。在某些情况下，批处理应用程序可执行读取输入文件和更新主文件的单个功能。但是，如果在此类应用程序异常结束之后再次启动它非常重要，则可以对它使用落实控制。

输入文件是一个更新文件，它在记录中有一个代码，指示已经处理了记录。此文件和已更新的任何文件处于落实控制之下。当代码存在于输入文件中时，它表示已完成的事务。程序读取整个输入文件并绕过具有已完成代码的任何记录。这允许同一程序逻辑在正常情况和再次启动的情况下都可以使用。

如果批处理应用程序包含互相依赖的输入记录并且包含分流或总路，则可使用通知对象来提供有关再次启动的信息。保存在通知对象中的值用于从输入文件中上次落实的事务再次启动处理。

如果输入记录互相依赖，则可将它们作为事务处理。批处理作业可以锁定最多 500 000 000 条记录。可以使用“查询选项文件”（QAQQINI）减少此限制。使用更改查询属性（CHGQRYA）命令的 QRYOPTLIB 参数来指定要使用的作业的“查询选项文件”。在“查询选项文件”中使用 COMMITMENT_CONTROL_LOCK_LEVEL 值作为作业的锁定限制。

超过 2000 个锁定的任何落实周期可能会明显降低系统性能。另外，对于交互式应用程序存在相同的锁定注意事项，但锁定在批处理应用程序中的记录的时间长度可能没有交互式应用程序中的重要。

两阶段落实控制

两阶段落实控制确保多个系统上的可落实资源保持同步。OS/400(R) 按照 SNA LU 6.2 体系结构支持两阶段落实。有关系统对两阶段落实使用的内部协议的更多详细信息，请参阅 *SNA Transaction Programmer's Reference for LU Type 6.2, GC30-3084-05*。OS/400 的所有支持的发行版都支持 SNA LU 6.2 的 Presumed Nothing 协议和 SNA LU 6.2 的 Presumed Abort 协议。

通过使用 TCP/IP 作为“分布式工作单元”（DUW）DRDA(R) 协议也支持两阶段落实。要使用 TCP/IP DUW 连接，所有的系统（应用程序请求器和应用程序服务器）必须为 V5.1.0 或更新版本。有关 DRDA 的更多信息，请参阅 Open Group Web 站点  处的 Open Group Technical Standard, *DRDA V2 Vol. 1: Distributed Relational Database Architecture*。

在两阶段落实下，系统分两个阶段来执行落实操作:

- 在**准备阶段**期间，资源管理器向其事务管理器发出落实请求。事务管理器通知它管理的任何其它资源和要落实已准备好的事务的其它事务管理器。所有资源管理器必须响应以说明它们已准备好落实。这称为**表决**。

- 在**落实阶段**期间，启动落实请求的事务管理器将根据准备阶段的结果决定要执行什么操作。如果准备阶段成功完成，并且所有参与者表决已准备好，则事务管理器将指示它管理的所有资源和要落实事务的其它事务管理器。如果准备阶段没有成功完成，则将指示所有事务管理器和资源管理器回滚事务。

具有远程资源的落实和回滚操作

当远程资源在落实控制下时，启动程序将向所有远程代理程序发送落实请求。将通过事务程序网络发送该请求。每个代理程序将以落实操作的结果响应。

如果在准备阶段期间发生错误，则启动程序将向所有代理程序发送回滚请求。如果在落实阶段期间发生错误，则系统将尝试使尽可能多的位置处于已落实状态。这些尝试可能导致启发式混合状态。有关可能出现的状态的更多信息，请参阅两阶段落实控制的事务的状态。

任何错误将发送回启动程序（在该位置会向用户发送关于这些错误的信号）。如果在启动落实控制（STRCMTCTL）命令处指定了缺省日志，则将编写 C LW 项。如果发生错误，即使指定了 OMTJRNE(*LUWID)，也会编写这些项。您可以使用这些项以及错误消息和落实定义的状态信息来尝试手工同步可落实资源。

当远程资源在落实控制下时，启动程序将向所有远程代理程序发送回滚请求。将通过事务程序网络发送该请求。每个代理程序将以回滚操作的结果响应。

有关更多信息，请参阅以下内容:

- 落实处理中的角色
- 两阶段落实控制的事务状态
- 两阶段落实控制的落实定义

落实处理中的角色

如果落实事务涉及多个资源管理器，则每个资源管理器都在事务中担任一个角色。资源管理器负责事务期间执行的落实或回滚更改。按资源类型分类的资源管理器为:

FILE
数据库管理器

DDM
数据库管理器

DDL
数据库管理器

DRDA(R)
通信事务程序

LU62
通信事务程序

API
API 出口程序

以下图形显示事务中的基本角色。图形中显示的结构称为**事务程序网络**。该结构可以为单级树和多级树。

两阶段落实处理中的角色: 单级树

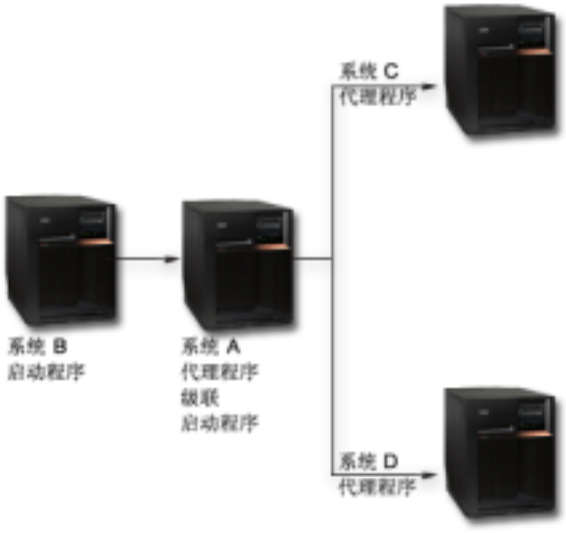
当系统 A 上的应用程序发出落实请求时，系统 A 上的资源管理器将成为**启动程序**。对于通过 TCP/IP 的 DRDA 分布式工作单元，该启动程序称为**协调程序**。

其它三个系统（B、C 和 D）的资源管理器成为此事务的**代理程序**。对于通过 TCP/IP 的 DRDA 分布式工作单元，有时代理程序称为**参与者**。

两阶段落实处理中的角色: 多级树

如果应用程序正在使用 APPC 通信来执行两阶段落实，则系统之间的关系可以从一个事务更改为另一个事务。下图显示当系统 B 上的应用程序发出落实请求时的相同系统。此配置是多级树。

此图形中的角色不适用于通过 TCP/IP 的 DRDA 分布式工作单元，原因是不支持多级事务树。



事务程序网络具有另一级别，因为系统 B 没有与系统 C 和系统 D 直接通信。系统 A 中的资源管理器现在具有代理程序和**级联启动程序**的角色。

为了提高 LU6.2 两阶段事务的性能，启动程序可能将**上一代理程序**的角色分配给某个代理程序。上一代理程序不参与准备阶段。在落实阶段中，将首先落实上一代理程序。如果没有成功落实上一代理程序，则启动程序将指示回滚其它代理程序。

对于通过 TCP/IP 的 DRDA 分布式工作单元，协调程序可能将**再同步服务器**的角色分配给参与者。所负责的再同步服务器将在协调程序出现通信故障或协调程序具有系统故障时再同步其它参与者。

两阶段落实控制的事务状态

在属于事务程序网络一部分的每个位置处建立落实定义。对于每个落实定义，系统将跟踪其当前事务和先前事务的状态。如果事务因通信或系统故障中断，则系统使用该状态来决定是落实还是回滚。如果多个位置参与事务，则可能会比较每个位置处的事务状态以确定正确的操作（落实或回滚）。这种在位置之间通信以确定正确操作的过程称为**再同步**。

下表显示:

- 在事务期间可能发生的基本状态。
- 可能发生的附加状态。
- 如果事务因通信或系统故障中断，则状态是否需要再同步。可能的值为:

  不需要
  每个位置都可以独立地作出正确的决定。

  可能需要
  每个位置都可以作出正确的决定，但是可能需要将该决定通知给启动程序。

  必需的
  必须确定每个位置的状态，才能作出正确的决定。
- 通信或系统故障采取的操作。

状态名	描述	如果中断事务，则再同步	通信或系统故障采取的操作
两阶段落实处理期间的基本状态:			
复位（**RST**）	从落实边界开始，直到程序发出落实或回滚的请求。	不需要。	回滚暂挂的更改。
正在准备（**PIP**）	启动程序启动了准备阶段。所有位置尚未表决。	可能需要。	回滚暂挂的更改。
已准备（**PRP**）	此位置和事务程序网络中的比此位置低的所有位置已经表决落实。此位置尚未从启动程序中接收到通知以落实。	必需的。	不确定。取决于再同步过程的结果。
正在进行的落实（**CIP**）	所有位置已经表决落实。启动程序已经启动了落实阶段。	必需的。	落实了暂挂的更改。执行了再同步以确保落实了所有位置。如果另一位置报告了启发式回滚，则将报告错误。
落实（**CMT**）	所有代理程序已落实并将应答返回至此节点。	可能需要。	无。

状态名	描述	如果中断事务，则再同步	通信或系统故障采取的操作
两阶段落实处理期间的附加状态:			
上一代理暂挂（**LAP**）	如果选择了上一代理程序，则此状态将发生在启动程序中的 PIP 状态和 CIP 状态之间。启动程序已经指示要落实的上一代理程序并且尚未接收到响应。	必需的	不确定。取决于再同步过程的结果。
只读表决（**VRO**）	此代理程序通过指示它不具有暂挂的更改而向准备阶段作出响应。如果允许只读表决状态，则不会将此代理程序包括在落实阶段中。	可能需要。	无。
必需回滚（**RBR**）	发生下列一种情况: • 代理程序在落实操作前发出了回滚请求。 • 发生了事务故障。 • 已使用 QTNRBRQD API 将事务置于必需回滚状态。 不允许事务程序在落实控制下执行任何附加更改。	可能需要。	回滚暂挂的更改。
因为操作员操作或错误而发生的情况:			
强制回滚	已经通过操作员的介入回滚了此位置和事务程序网络中比此位置低的所有位置（上一代理程序除外）。	可能需要	已经回滚了暂挂的更改。
强制落实	已经通过操作员的介入落实了此位置和事务程序网络中比此位置低的所有位置（上一代理程序除外）。	可能需要	已落实暂挂的更改。
启发式混合（**HRM**）	已落实一些资源管理器。已经回滚一些资源管理器。操作员介入或发生了系统错误。启发式混合在落实定义屏幕上不会显示为状态。将通知消息发送给操作员。	可能需要	操作员必须在所有参与的位置执行恢复操作以使数据库处于一致状态。

两阶段落实控制的落实定义

在启动落实控制之后，可以使用 QTNCHGCO（更改落实选项）API 来更改事务的落实选项。根据您的环境和应用程序，更改落实选项可提高系统的性能。

下列链接描述落实选项以及使用它们的原因:

- 允许只读表决
- 不等待结果
- 指示“确认以忽略”

- 不选择上一代理程序
- 可靠表决

如果您正通过 TCP/IP 连接使用 DRDA(R) 分布式工作单元，则唯一适用的选项是“允许只读表决”。

两阶段落实的落实定义：允许只读表决： 通常，事务管理器参与落实处理的两个阶段。要提高落实处理的性能，可在事务中设置一些或所有位置以允许事务管理器执行只读表决。如果在事务期间位置没有可落实的更改，则事务管理器在准备阶段期间执行只读表决。位置不参与已落实的阶段。由于在一个或多个远程位置处不执行更新的事务期间消除了在落实阶段期间正常发生的通信流，所以这可提高总体性能。

在启动落实控制之后，可以使用更改落实选项（QTNCHGCO） API 来将允许的只读表决选项更改为 Y。如果以下条件成立，则您可能想执行此操作:

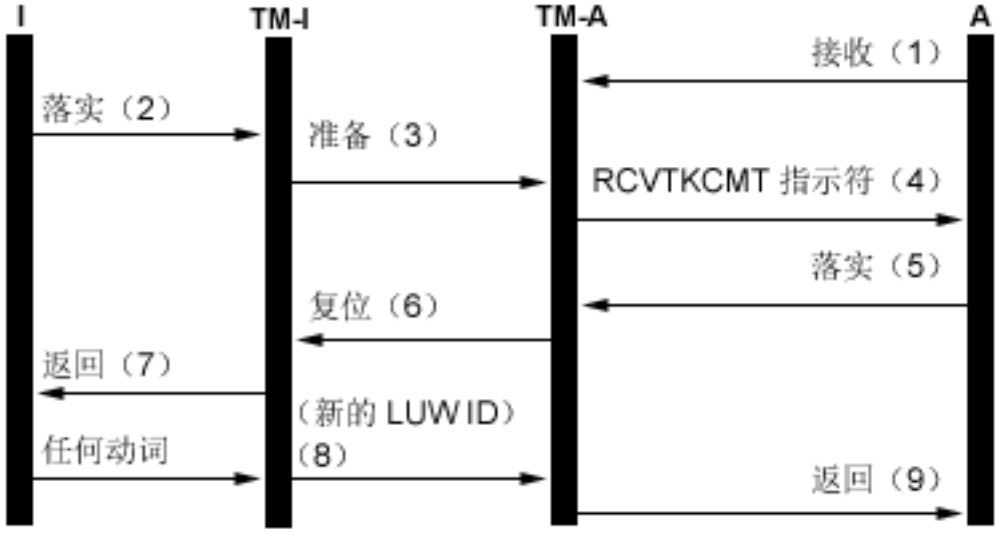
- 一个或多个远程系统通常没有事务的任何可落实更改。
- 事务不依赖于先前事务设置的文件游标（下一条记录）的位置。当位置执行只读表决时，如果回滚事务，则永远不通知应用程序。该位置已将任何读操作落实至数据库文件，因此移动了游标位置。文件游标的位置通常仅在执行顺序处理时才会非常重要。

如果落实定义设置为允许只读表决，则应用程序等待来自另一位置的下一个消息流。

允许的只读表决选项用于实质上是客户机／服务器的应用程序。如果程序 A 的目的只是为了满足来自程序 I 的请求而不执行任何独立的工作，则将只读表决选项用于程序 A 是非常正确的。

当代理程序执行只读表决时不具有上一代理优化的落实处理的流

下图显示当代理程序执行只读表决时并且当应用程序发出不具有上一代理优化的落实指令时，应用程序与事务管理器中的消息流。启动应用程序和代理应用程序都不知道两阶段落实处理。图中圆括号 () 中的数字对应于后面描述中编号的项。



图注

I = 启动程序（启动落实请求的应用程序）
TM-I = 启动程序的事务管理器
A = 代理程序（接收落实请求的启用程序）
TM-A = 代理程序的事务管理器

以下是代理程序执行只读表决时不具有上一代理优化的正常处理事件的描述。它描述了基本流。当事务程序网络具有多层或当发生错误时，事件的顺序会变得更为复杂。

1. 应用程序 A 执行接收请求以指示它已准备好接收来自程序 I 的请求。
2. 启动应用程序（I）发出落实指令。
3. 启动程序（TM-I）的事务管理器担任此事务的启动程序的角色。它通过将准备消息发送至正在参与事务的所有其它位置来启动准备阶段。
4. 每个其它位置的事务管理器担任代理程序（TM-A）的角色。TM-A 通知应用程序 A 已接收了要落实的请求。对于 ICF 文件，通知的格式为设置为“打开”的“接收采用落实”（RCVTKCMT）ICF 指示符。
5. 应用程序 A 通过发出落实指令（或回滚指令）作出响应。这是应用程序的表决。
6. 如果应用程序 A 使用了“更改落实选项”（QTNCHGCO）API 来将“允许的只读表决”落实选项设置为 Y，且事务期间在代理程序中没有作任何更改，则代理程序（TM-A）通过复位消息响应启动程序（TM-I）。代理程序将不具有落实阶段。
7. 将一个返回发送至应用程序（A）以指示代理程序 TM-A 中的事务已完成。
8. 下一次启动程序（TM-I）将任何消息（数据流或落实指令）发送至代理程序（TM-A）时，TM-I 就会将消息与它的当前事务标识一起发送。发生这种情况的原因是：如果在落实操作期间 TM-I 与其它系统之间发生了通信故障，TM-I 中可能生成了新的事务标识。
9. 将一个返回发送至应用程序（A）以指示代理程序 TM-A 中的事务已完成。该返回将被延迟，直到接收到下一条消息为止，原因是在应用程序 A 可以启动下一个事务之前必须已接收到来自 TM-I 的新事务标识。

有关两阶段落实控制的更多信息，请参阅落实处理中的角色和两阶段落实控制的事务状态。

两阶段落实的落实定义：不等待结果： 当在落实操作期间发生通信或系统故障以至于需要再同步，则缺省值是等待再同步完成，落实操作才能完成。

注意: 如果正在通过 TCP/IP 连接使用 DRDA(R) 分布式工作单元，则“不等待结果”选项不适用。通过 TCP/IP 连接的 DRDA 分布式工作单元从不等待结果。

如果下列条件成立，则考虑更改此行为:

- 参与的应用程序彼此独立。
- 程序逻辑不需要先前事务的结果来确保数据库文件保持同步。

在启动落实控制后，可以使用 QTNCHGCO（更改落实选项）API 来指定落实定义不等待再同步的结果。如果为等待结果选项指定 N（否），则系统将使用数据库服务器作业（QDBSRVnn）来异步处理再同步。

注意: 在 IPL 处理期间启动了这些数据库服务器作业。如果更改落实控制的选项，则这对系统启动的作业数没有影响。

此主题只涉及已解析的等待结果选项的两个值，Y（是）和 N（否）。实际上还有两个值可以指定，L（“是”或“从启动程序继承”）和 U（“否”或“从启动程序继承”）。当使用这些值时，在每个落实操作期间使用的实际值由系统解析为“是”或“否”。QTNCHGCO（更改落实选项）API 主题具有关于这些值的更多详细信息。

注意: 如果启动程序和代理程序支持假定的异常终止，则只可以由代理程序继承启动程序的值。

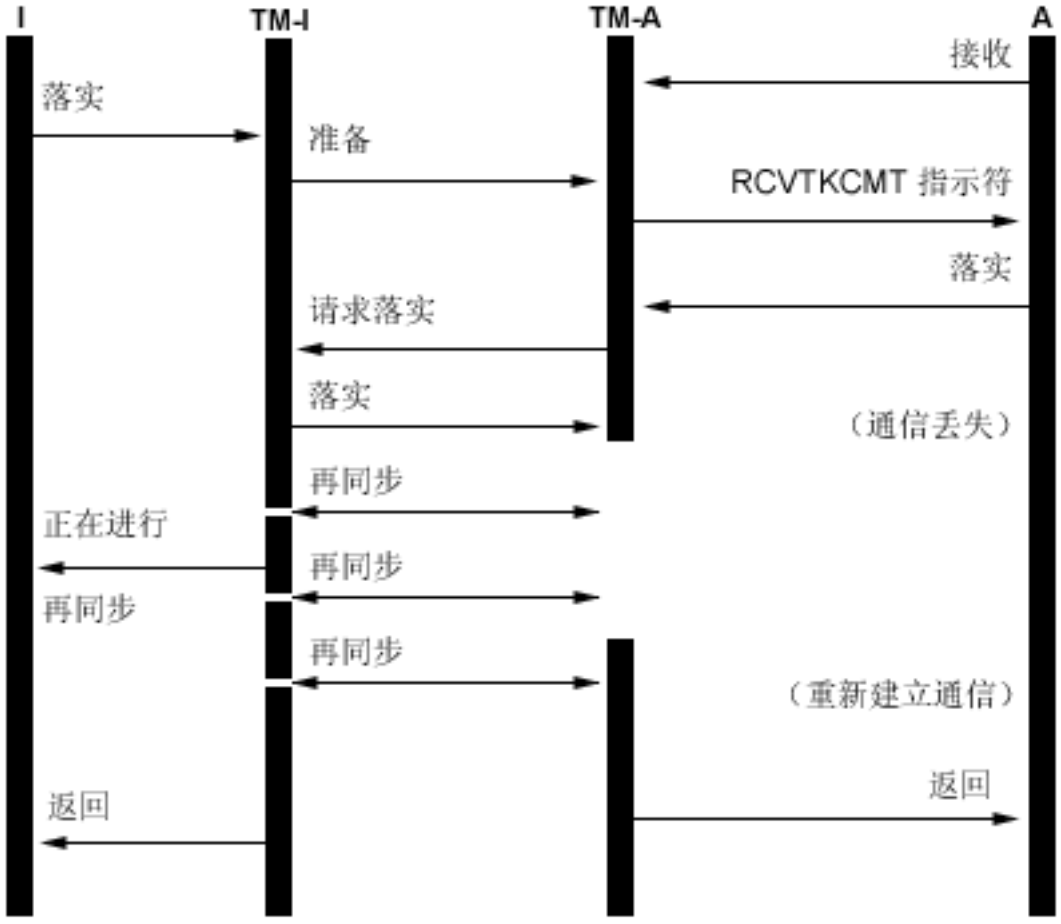
等待结果（WFO）选项不会影响正常的没有错误的落实处理。如果发生了错误，则 WFO 选项将使用下列条件来确定应用程序是否等待再同步:

- 如果解析的 WFO 选项为 Y（是），则应用程序将等待再同步的结果。
- 如果解析的 WFO 选项为 N（否）并且支持假定的异常终止协议的位置的准备阶段或回滚期间发生了通信故障，则不会执行再同步并且将回滚落实定义。
- 如果落实定义不确定（事务状态为“已准备”或“上一代理暂挂”），则应用程序将等待再同步的结果而不管解析的 WFO 值。有关不确定的落实定义的进一步信息，请参阅两阶段落实控制的事务的状态。
- 如果解析的 WFO 选项为 N 并且条件二或条件三中的任何一个都不成立时，则系统将再次尝试再同步。如果不成功的话，则系统将向应用程序发出 STATUS 消息 CPF83E6 以指示再同步正在进行中。

  因为 CPF83E6 是 STATUS 消息，所以它仅在应用程序正在监视它时起作用。通常，应用程序可以将此消息作为参考消息对待。正在参与事务的系统将尝试再同步事务直到修复了故障为止。这些后续的再同步尝试在数据库服务器作业中执行。如果在数据库服务器作业中执行的后续再同步尝试失败，则将把消息 CPI83D0 发送至 QSYSOPR。

等待结果 - 是

在下面的图形中，启动程序（I）的落实定义使用缺省值 Y（是）作为等待结果选项的值。当 TM-I 和 TM-A 之间的通信中断时，应用程序 A 和应用程序 I 将等待，直到再同步事务为止。

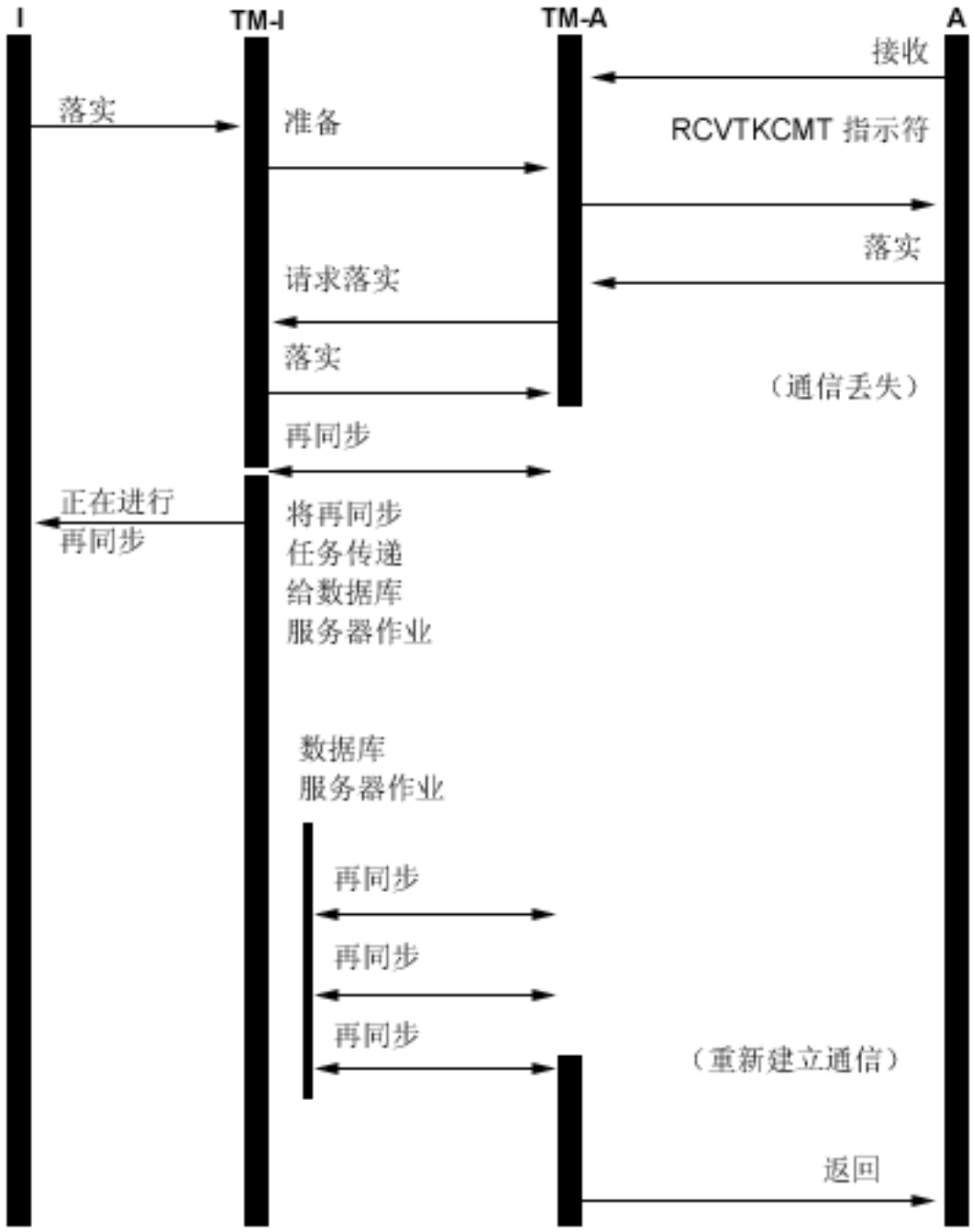


等待结果 - 否

在下面的图形中，启动程序的落实定义将解析的 WFO 设置为 N（否）。TM-A 符合先前列表中的条件 3，而 TM-I 符合条件 4。在尝试与 TM-A 再同步后，将把控制权返回给应用程序 I。数据库服务器作业尝试再同步。

当落实请求成功完成时，应用程序 I 不会再接收返回指示符。控制不会返回至代理应用程序（A），直到重新建立通信为止。这取决于故障的计时。在此情况下，从启动程序接收落实消息之前将发生通信故障，从而使 TM-A 不确定是落实还是回滚。当事务管理器不确定时，它将保留控制直到再同步完成为止，而不管在该系统所解析的 WFO 值。

如果想要所有系统的应用程序在再同步完成前继续，则必须将所有系统上解析的 WFO 选项更改为 N（否），或将启动程序设置为 N 并且将剩余的系统设置为 U（“无”或“从启动程序继承”）。但是请记住，当事务管理器不确定是落实还是回滚时，将忽略解析的 WFO 选项，并且通常它将等待，直到再同步完成时才返回控制权。



当建立了与远程关系数据库的连接，并且未启动受保护对话，则系统将等待结果值隐式更改为 N。原因是当等待结果值为 N 并且远程系统支持假定的异常终止时，落实操作的性能将提高。只对 DRDA 和 DDM 应用程序执行等待结果值的隐式更改。除非 APPC 应用程序调用 QTNCHGCO API 来更改缺省的等待结果值“是”，否则它们将使用缺省值。

两阶段落实的落实定义：指示“确认以忽略”： 通常，事务程序网络中的每个位置的事务管理器参与每个落实或回滚操作。要提高性能，可以在事务中设置部分或所有位置以允许事务管理器指示“确认以忽略”。

注意: 如果正在通过 TCP/IP 连接使用 DRDA(R) 分布式工作单元，则指示“确认以忽略”选项不适用。

如果在事务期间，未向位置发送通信流，则在执行落实或回滚操作期间将忽略该位置。这将提高整体性能，原因是在没有数据要发送至一个或多个远程位置的事务期间将省去通常发生在落实或回滚期间的通信流。

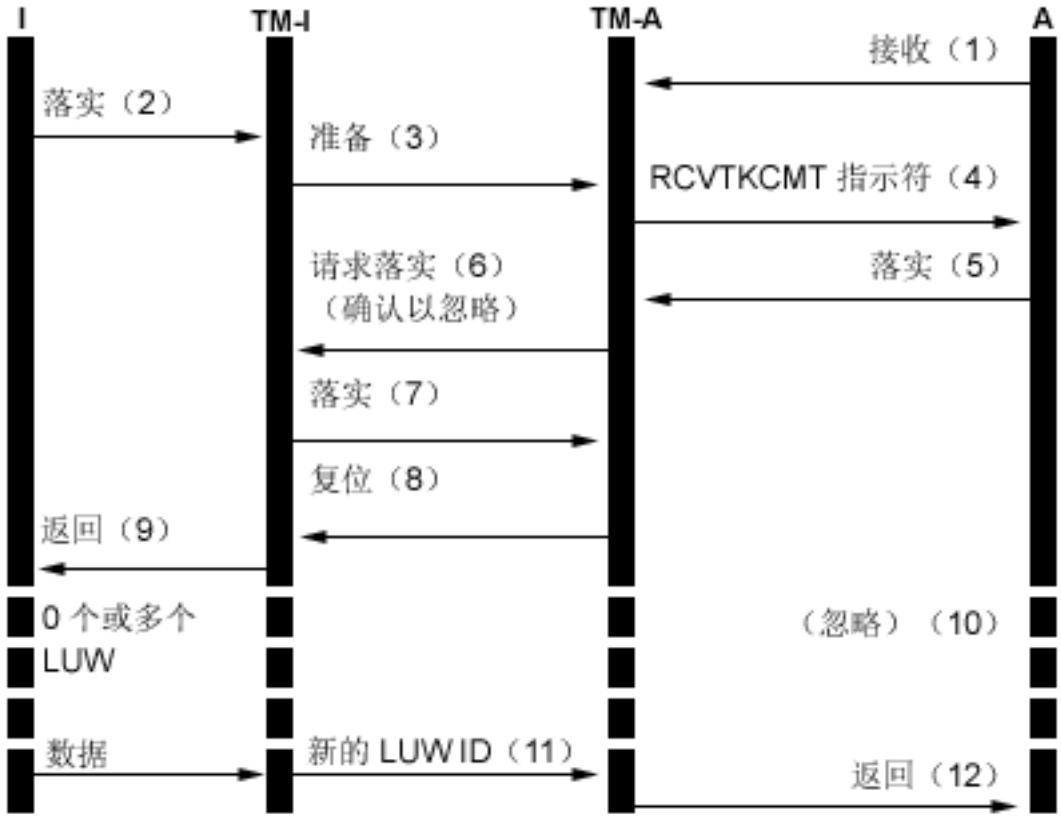
在启动落实控制后，可以使用“更改落实选项”（QTNCHGO）API 来将“确认以忽略”选项更改为 Y（是）。如果一个或多个远程系统经常不涉及事务，则您可能想这么做。

如果将落实定义设置为指示“确认以忽略”，则应用程序将等待来自另一位置的下一个消息流。

“确认以忽略”选项适用于性质上是客户机／服务器的应用程序。如果程序 A 的唯一目的是满足来自程序 I 的请求并且不执行任何独立工作，则将“确认以忽略”选项用于程序 A 是非常正确的。

当代理程序表决“确认以忽略”时，不具有上一代理优化的落实处理流程

下图显示当代理程序指示“确认以忽略”并且当应用程序没有上一代理优化而发出落实指令时应用程序和事务管理器中的消息流。启动应用程序和代理应用程序都没有意识到两阶段落实处理。图中圆括号 () 中的数字对应于后面描述中编号的项。



图注

I = 启动程序（启动落实请求的应用程序）
TM-I = 启动程序的事务管理器
A = 代理程序（接收落实请求的应用程序）
TM-A = 代理程序的事务管理器

以下是代理程序表决“确认以忽略”时不具有上一代理优化的正常处理事件的描述。它描述了基本流。当事务程序网络具有多层或当发生错误时，事件的顺序会变得更为复杂。

1. 应用程序 A 执行接收请求以指示它已准备好接收来自程序 I 的请求。
2. 启动应用程序（I）发出落实指令。
3. 启动程序（TM-I）的事务管理器担任此事务的启动程序的角色。它通过将准备消息发送至正在参与事务的所有其它位置来启动准备阶段。
4. 每个其它位置的事务管理器担任代理程序（TM-A）的角色。TM-A 通知应用程序 A 已接收了要落实的请求。对于 ICF 文件，通知的格式为设置为“打开”的“接收采用落实”（RCVTKCMT）ICF 指示符。
5. 应用程序 A 通过发出落实指令（或回滚指令）作出响应。这是应用程序的表决。
6. 如果应用程序 A 使用“更改落实选项”（QTNCHGCO）API 来将“确认以忽略”落实选项设置为 Y，则当代理程序（TM-A）以请求落实消息响应启动程序（TM-I）时，将发送指示符。

注意: 对“确认以忽略”落实选项的任何更改不会生效，直到下一个成功的落实操作完成。

7. 当启动程序（TM-I）接收所有表决时，TM-I 将发送落实消息。这将启动落实阶段。
8. 每个代理程序（TM-A）将落实并以复位消息进行响应。
9. 一个返回将发送至应用程序（I）以指示启动程序处的事务已完成。

10. 在 TM-I 上可能发生任意数目的事务，它们都不需要对 TM-A 或来自 TM-A 的数据进行更改。TM-A 不包括在这些事务中。
11. 下一次启动程序（TM-I）向代理程序（A）发送消息时，将随该消息发送新的事务标识。如果启动程序在向代理程序发送消息前执行任何落实或回滚操作，则不会在这些操作期间向代理程序发送任何消息（这些落实或回滚操作期间会“忽略”代理程序）。因为在忽略代理程序的时候，可能已经在启动程序处落实或回滚了一个或多个事务，所以当下一个消息发送至代理程序时，启动程序必须传送其当前事务标识。
12. 一个返回将发送至应用程序（A）以指示初始落实已完成并且它正在参与当前事务。

两阶段落实的落实定义：不选择上一代理程序: 缺省情况下，启动程序的事务管理器自由选择任何代理程序作为落实操作期间的上一代理程序。

注意: 如果正在通过 TCP/IP 连接使用 DRDA(R) 分布式工作单元，则“不选择上一代理程序”选项不适用。

在多级树的情况下，由其启动程序选择作为上一代理程序的任何代理程序也能够自由选择自己的上一代理程序。当在落实操作期间选择上一代理程序时会提高性能，原因是省略了启动程序和其上一代理程序之间的两个通信流（会对这些系统省略准备阶段）。

但是，当启动程序向其上一代理程序发送请求落实时，它必须等待，直到接收到上一代理程序的表决时才能继续。这与落实定义的“等待结果”值无关。在正常并且没有错误的落实处理中，这不会产生任何问题。但是，如果在此窗口中出现错误，则直到再同步完成启动程序才能继续。如果启动应用程序正在处理来自一个终端用户的请求，则这会涉及到可用性问题。

必须考虑在正常落实操作期间的性能提高是否比发生这样的错误时对可用性的影响更重要。注意，如果在将请求落实发送至上一代理程序前发生错误，则 LUW 将立即回滚并且启动程序不会等待。因此，错误导致启动程序等待时的窗口非常小，所以这样的错误不常见。

如果您决定对可用性的影响不及提高性能重要，则可以将落实定义更改为不选择上一代理程序。在启动落实控制后，可以使用更改落实选项（QTNCHGCO）API 来将允许的上一代理程序选项更改为 N。

可靠表决影响落实处理的流: 可靠表决是一个提高性能的优化方法，它通过在落实操作之后较早地返回至启动应用程序和在落实操作期间除去一个消息来提高性能。对于使用 TCP/IP 的 DRDA(R) 分布式工作单元，没有显式的可靠表决优化。但是，OS/400(R) 从不请求 TCP/IP 连接的复位（遗忘）确认。因此，复位（遗忘）对于 TCP/IP 连接总是隐式的。

在启动落实控制后，可以使用更改落实选项（QTNCHGCO）API 来将接受可靠表决选项更改为 Y。

可靠表决可以看作是代理程序对其启动程序的承诺，如果代理程序在不确定时发生通信故障，则不会对代理程序进行启发式决策。使用可靠表决优化的代理程序在落实的准备阶段期间将指示符发送给启动程序。如果启动程序也在使用可靠表决优化，则它将指示符发送给代理程序来指示不需要复位以响应落实消息。这将省略复位消息，并允许事务管理器在一发送落实消息后就返回至启动程序处的应用程序。

如果满足下列条件，则考虑使用可靠表决优化:

- 在发生系统或通信故障的情况下，不可能对不确定的代理程序做出启发式决策，除非不能修复该故障。
- 程序逻辑不需要先前事务的结果来确保数据库文件保持同步。

只有满足下列所有条件，OS/400 才会使用可靠表决优化:

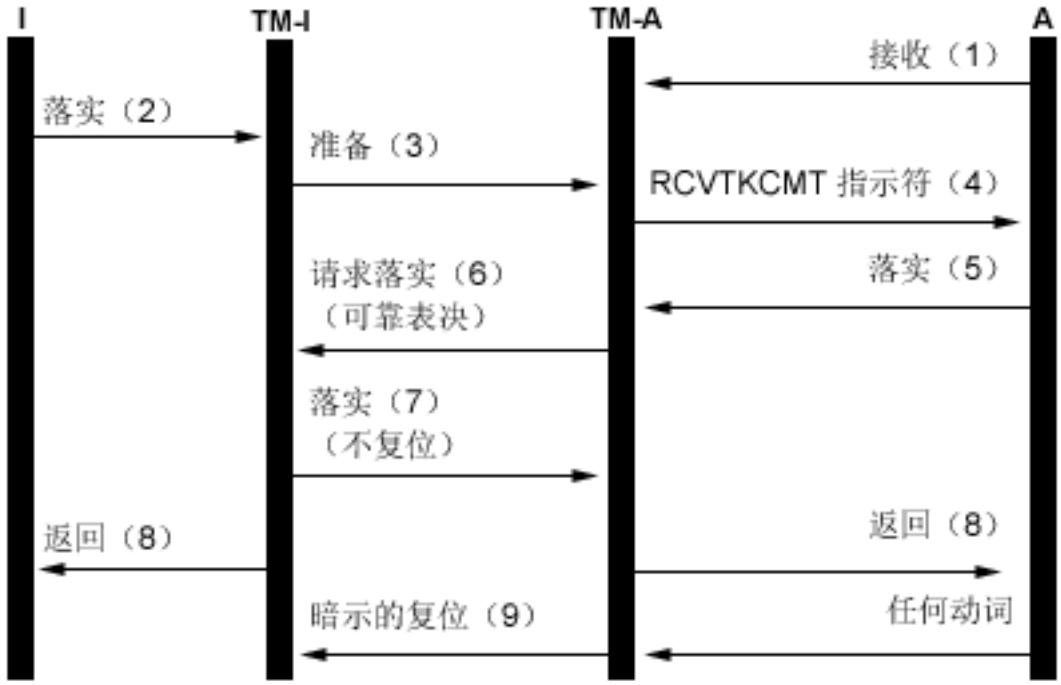
- 启动程序和代理程序位置支持落实控制的假定异常终止级别。
- 启动程序位置接受来自代理程序的可靠表决指示。在 OS/400 启动程序上，这取决于两个落实选项的值:

– “等待”结果落实选项的值必须为“否”（“是”为缺省值）。
– “接受”可靠表决落实选项的值必须为“是”（“是”为缺省值）。

- 代理程序位置在准备阶段期间进行可靠表决。OS/400 代理程序总是进行可靠表决。这是因为只可以通过手工过程进行启发式决策，该手工过程将警告进行启发式决策的可能的副作用。

以可靠表决优化进行的落实处理流程

下图显示使用可靠表决优化时应用程序和事务管理器中的消息流。启动应用程序和代理应用程序都没有意识到两阶段落实处理。图形中括号 () 中的数字与随后的描述中的编号项相对应。



图注

I = 启动程序（启动落实请求的应用程序）
TM-I = 启动程序的事务管理器
A = 代理程序（接收落实请求的应用程序）
TM-A = 代理程序的事务管理器

以下是当代理程序可靠表决时没有上一代理优化的正常处理的事件描述。它描述了基本流。当事务程序网络具有多个级别或当发生错误时，事件顺序会变得更复杂。

1. 应用程序 A 执行接收请求以指示它已准备好从程序 I 接收请求。
2. 启动应用程序（I）发出落实指令。
3. 启动程序（TM-I）的事务管理器担任此事务的启动程序的角色。它通过将准备消息发送至所有正在参与事务的其它位置来启动准备阶段。
4. 每个其它位置的事务管理器担任代理程序（TM-A）的角色。TM-A 通知应用程序 A 已接收了要落实的请求。对于 ICF 文件，通知的格式为设置为“打开”的“接收采用落实”（RCVTKCMT）ICF 指示符。
5. 应用程序 A 通过发出落实指令（或回滚指令）作出响应。这是应用程序的表决。
6. 代理程序（TM-A）以请求落实消息响应启动程序（TM-I）。OS/400 系统以请求落实发送可靠表决指示符。

7. 当启动程序（TM-I）接收所有表决时，TM-I 将发送落实消息。如果“等待”结果落实选项为 N（“否”）并且“接受”可靠表决落实选项为 Y（“是”），则将以落实消息发送“不”复位指示符。这将通知代理程序不需要复位消息来响应落实。
8. 事务已完成。将把返回发送给应用程序（I 和 A）。此返回指示落实操作是成功的。如果由于在接收到已落实消息之前进行的启发式决策而导致在系统 A 发生启发式破坏，则不会通知应用程序 I。而是把消息发送给 QSYSOPR 消息队列。但是，应用程序 A 将接收到启发式破坏指示。
9. 下一次代理程序（TM-A）向启动程序（TM-I）发送任何消息（数据流或落实指令）时，将与该消息一起发送暗示的复位指示符，以通知 TM-I TM-A 已成功完成落实。原因是 TM-I 必须保留关于已完成事务的信息，直到它确认 TM-A 已成功接收到了步骤 7 中的落实消息为止。

落实控制的 XA 事务支持

DB2(R) UDB iSeries(TM) 版可以参与 X/Open 全局事务。The Open Group 定义了事务性工作的业界标准模型，它允许对不相关资源执行的更改成为单个全局事务的一部分。此处有一个示例，是对由两个单独的供应商提供的数据库执行的更改。此模型称为“X/Open 分布式事务处理”模型。以下出版物详细描述了“X/Open 分布式事务处理”模型:

- X/Open Guide, February 1996, Distributed Transaction Processing: Reference Model, Version 3 (ISBN:1-85912-170-5, G504), The Open Group.
- X/Open CAE Specification, December 1991, Distributed Transaction Processing: The XA Specification (ISBN:1-872630-24-3, C193 or XO/CAE/91/300), The Open Group.
- X/Open CAE Specification, April 1995, Distributed Transaction Processing: The TX (Transaction Demarcation) Specification (ISBN:1-85912-094-6, C504), The Open Group.

在尝试使用 DB2 UDB iSeries 版提供的 XA 事务支持前，您应熟悉这些书中的信息，尤其是“XA 规范”。可以在 Open Group Web 站点  中找到这些书。

DTP 模型有五个组件:

应用程序（AP）
通过指定一系列涉及资源（如数据库）的操作来实现用户的必需功能。它定义全局事务的开始和结束、访问事务边界内的资源并通常决定是落实还是回滚每个事务。

事务管理器（TM）
管理全局事务并协调启动它们的决定，以及落实或回滚它们以便确保原子事务完成。TM 还在组件失效后协调与 RM 的恢复活动。

资源管理器（RM）
管理计算机共享资源的已定义部分，如数据库管理系统。AP 使用每个 RM 定义的接口来执行事务性工作。TM 使用 RM 提供的接口来执行事务完成。

通信资源管理器（CRM）
允许模型的实例访问当前 TM 域内或域外的其它实例。CRM 不属于 DB2 UDB iSeries 版的范围，此处不讨论它们。

通信协议
CRM 用来彼此通信的协议。这不属于 DB2 UDB iSeries 版的范围，此处不讨论它。

“XA 规范”是 DTP 模型的一部分，它描述 DTP 模型的 TM 和 RM 组件使用的一组接口。DB2 UDB iSeries 版将这些接口作为一组 UNIX(R) 样式 API 和出口程序来实现。请参阅 XA API 以获取关于这些 API 的详细文档以及获取关于如何将 DB2 UDB iSeries 版用作 RM 的更多信息。

iSeries 导航器和 XA 事务

iSeries 导航器支持将 XA 事务作为**全局事务**来管理

“全局事务”可能包含 DB2 UDB iSeries 版外部和内部的更改。全局事务由外部“事务管理器”通过使用“Open Group XA 体系结构”或其它相似的体系结构来协调。应用程序使用“事务管理器”提供的接口落实或回滚全局事务。“事务管理器”使用 XA 体系结构或另一体系结构定义的落实协议来完成事务。DB2 UDB iSeries 版在参与全局事务时充当“XA 资源管理器”。有两种类型的全局事务:

- **事务作用域的锁定**: 以事务名义获取的锁定的作用域为该事务。该事务可以从一个作业或线程移动到另一个作业或线程。
- **作业作用域的锁定**: 以事务名义获取的锁定的作用域为该作业。该事务不能从启动它的作业中移动。

如果正在针对驻留在本地系统上的数据库运行 XA 事务，则将 XA API 用于事务作用域锁定。这些 API 比作业作用域锁定的 XA API 有更少的限制并且在下列情况中提供更好的性能:

- 如果曾使用多个 SQL 连接来在单个 XA 事务分支上工作。
- 如果使用单个 SQL 连接来在多个并发 XA 事务分支上工作。

在这些情况下，当将 XA API 用于“作业作用域锁定”时，必须启动单独的作业来运行 XA 事务分支。

如果正在针对驻留在远程系统上的数据库运行 XA 事务，则必须将 XA API 用于“作业作用域锁定”。

XA 事务的注意事项

在将 DB2 UDB iSeries 版用作 RM 前，请了解下列注意事项和限制。术语“线程”指的是不可执行线程的作业或可执行线程的作业中的单个线程。

下列注意事项适用于带有事务作用域锁定的事务和带有作业作用域锁定的事务，除非另外声明。

DB2 UDB iSeries 版注意事项

- XA 事务只可以在正在以 SQL 服务器方式运行的作业中执行。这存在一个影响就是在 XA 事务期间，在对 DB2 UDB iSeries 版进行更改时，应用程序限制使用 SQL 接口。如果 db2xa_open() API 用于没有以 SQL 服务器方式运行的作业中，则将隐式启动 SQL 服务器方式。请参阅落实控制的 SQL 服务器方式和线程作用域事务
- 在 XA API 调用期间由 DB2 UDB iSeries 版检测到的任何错误将通过 XA 规范的返回码报告。当仅通过返回码不能清楚地了解错误的意义时，诊断消息将保留在作业记录中。

嵌入式 SQL 注意事项

- 要将“结构化查询语言（SQL）”连接用于 XA 事务，在进行 SQL 连接前，必须使用 db2xa_open() 应用程序编程接口（API）。将连接的关系数据库必须通过 Xainfo 参数传递给 db2xa_open() API。要在该连接所路由至的作业中使用的用户概要文件和密码可能也要传递给 db2xa_open() API。如果未传递它，概要文件将缺省为在连接尝试期间指定或缺省的概要文件。

- 如果嵌入式 SQL 用来执行 XA 事务，则为每个连接执行的工作将路由至不同的作业，即使这些连接是在同一线程中建立的。这与没有 XA 的 SQL 服务器方式不同，在该方式中为单个线程中所有连接执行的工作都路由至同一作业。这是因为 XA 规范对于每个资源管理器实例都要求单独的准备、落实或回滚调用。

  注意: 以下注意事项只适用于具有作业作用域锁定的事务。
- 如果嵌入式 SQL 用来执行 XA 事务，则每个线程只对每个关系数据库建立一个连接。每当线程不与事务分支以活动方式关联时，通过其中一个线程连接请求的工作将使 RM 使用 TM 的 ax_reg() 出口程序以确定工作是启动、继续还是连接事务分支。

  如果该工作将启动事务分支，则它是通过该线程与相应关系数据库的连接执行的。

  如果该工作将连接事务分支，则通过与相应关系数据库的连接（该连接是在启动事务分支的线程中建立的）对它重新进行路由。注意系统不强制该连接的用户概要文件与连接线程的连接的用户概要文件相同。TM 将负责确保这不会影响安全问题。通常 TM 对于所有连接都使用相同的用户概要文件。将此用户概要文件授权给所有由 TM 管理的数据。关于访问此数据的进一步安全性由 TM 或 AP 管理，而不是使用标准 iSeries 安全技术管理。
- 如果该工作将继续事务分支，则使用的连接将取决于暂挂的事务分支关联是通过启动事务分支还是通过连接事务分支建立的。

  将通过同一连接执行后续的工作，直到使用 db2xa_end() API 来暂挂或结束与该事务分支的线程关联为止。

CLI 注意事项

- 如果 CLI 用来执行 XA 事务，则可能当使用 db2xa_open() API 之后在同一线程中建立多个连接。只要那些其它线程首先使用具有相同 Xainfo 参数值的 db2xa_open() API，就可以在其它线程中使用这些连接来执行 XA 事务。

  注意: 以下注意事项只适用于具有作业作用域锁定的事务。
- 如果 CLI 用来执行 XA 事务，则用来启动事务分支的连接必须用于该事务分支上的所有工作。如果另一线程将连接事务分支，则用来启动事务分支的连接的连接句柄必须传递给连接线程以便它可以通过同一连接来执行工作。同样，如果线程将继续事务分支，则将使用相同的连接。

  注意: 以下适用于具有事务作用域锁定和作业作用域锁定的事务。

  因为 CLI 连接句柄不能用于其它作业中，所以该连接功能只限用于运行在同一作业（该作业在使用 CLI 时启动事务分支）中的线程。

远程关系数据库注意事项

注意: 远程关系数据库的这些注意事项只适用于具有作业作用域锁定的事务。

- 仅当关系数据库驻留在支持“分布式工作单元”（DUW）DRDA(R) 连接的系统上，XA 与远程关系数据库的连接才受支持。这包括通过 SNA LU6.2 对话运行 DRDA 的系统。还包括当使用 TCP/IP 连接运行 DRDA 时使用 V5R1 的系统。
- db2xa_open() API 必须用于连接线程中才能使用 XA 连接功能。必须在启动事务分支的线程和连接线程中的 db2xa_open() API 上指定相同的关系数据库名和 RMID。如果尝试连接时，事务分支是活动的，则连接线程将阻塞。连接线程将保持阻塞，直到活动线程暂挂或结束其与事务分支的关联为止。

恢复注意事项

- 如果有必要强制事务分支落实或回滚（当它在已准备状态时），则可以使用为所有落实定义提供的手工试探落实和回滚支持。有关详细信息，请参阅何时强制落实和回滚以及何时取消再同步。

事务分支注意事项

- 关于 XA 事务分支的信息显示为落实控制信息的一部分，这些信息由 iSeries 导航器、处理作业（WRKJOB）、显示作业（DSPJOB）以及使用落实定义（WRKCMTDFN）命令显示。TM 名称、事务分支状态、事务标识和分支限定符都会显示。可以通过使用命令 `WRKCMTDFN JOB(*ALL) STATUS(*XOPEN)` 或通过在 iSeries 导航器中显示**全局事务**来显示与所有当前活动的 XA 事务相关的落实定义。

注意：以下项仅适用于具有作业作用域锁定的事务。

- 如果使用 db2xa_end() API 暂挂或结束线程和现有事务分支之间的关联，则该线程可能启动新的事务分支。如果用来启动新事务分支的连接在较早时候用来启动另一事务分支，并且 db2xa_end() API 已经结束或暂挂了线程与该事务分支的关联，则可能会启动新的 SQL 服务器作业。仅当 db2xa_commit() 或 db2xa_rollback() API 尚未完成第一个事务分支时，才需要新的 SQL 服务器作业。在此情况下，将把另一完成消息 SQL7908 发送至作业日志以标识新的 SQL 服务器作业，就好像建立连接时标识连接的初始 SQL 服务器作业一样。将把新事务分支的所有 SQL 请求路由至新的 SQL 服务器作业。当 db2xa_commit() 或 db2xa_rollback() API 完成了事务分支时，将回收新的 SQL 服务器作业并将其返回至预启动作业池。
- 当发生下列情况时，系统将把事务分支标记为“只能回滚”：
  - 线程在它仍与事务分支相关联时结束。
  - db2xa_close() API 用于具有与事务分支的活动关联的线程中。
- 当发生下列任何情况时，如果任何线程仍与事务分支相关联，系统将回滚该事务分支:
  - 结束了与该事务分支相关的连接。
  - 结束了启动事务分支的作业。
  - 系统失败。
- 在某种情况下，系统将回滚事务分支而不管是否仍存在相关联的线程。当正将连接工作路由至其中的 SQL 服务器作业结束时，将发生此情况。当针对该作业使用“结束作业”（ENDJOB）CL 命令时，才可能发生此情况。

注意：以下项仅适用于具有作业作用域锁定的事务。

- 当发生下列任何情况时，如果没有线程具有与该线程的活动关联，则事务分支将不受影响。TM 可以从任何线程中落实或回滚事务分支，该线程已使用具有与线程（启动事务分支的线程）中指定的相同 Xainfo 参数值的 db2xa_open() API。
  - 结束了与该事务分支相关的连接。
  - 执行事务分支的工作，但是不再具有与其的活动关联的线程或作业使用 db2xa_close() API。
  - 执行事务分支的工作，但是不再具有与其的活动关联的线程或作业使用 db2xa_close() API。
  - 系统失败。在此情况下，仅当事务分支处于准备状态时，它才不会受到影响。如果该事务分支处于空闲状态，则系统将回滚它。

落实控制的 SQL 服务器方式和线程作用域事务

具有作业作用域锁定的落实定义通常将作用域限定为激活组。如果作业是多线程的，则作业中的所有线程都可以访问落实定义并且对特定事务执行的更改可以跨多个线程扩散。即，其程序运行在相同激活组中的所有线程都参与单个事务。

在某些情况下，我们希望事务性工作的作用域限定为线程，而不是激活组。换句话说，每个线程将具有它自己的落实定义并且每个落实定义的事务性工作将独立于在其它线程中执行的工作。

通过使用更改作业（QWTCHGJB）API 来更改作业以使用 SQL 服务器方式运行，这样 DB2(R) UDB iSeries(TM) 版才支持这一点。当以 SQL 服务器方式请求 SQL 连接时，将把 SQL 连接路由至单独的作业。还将为该连接执行的所有后续 SQL 操作路由至该作业。当建立了连接后，将把完成消息 SQL7908 发送至 SQL 服务器方式作业的作业记录中以指示 SQL 请求正路由至哪个作业。落实定义由在此消息中指示的作业所拥有。如果发生错误，则可能有必要查看两个作业的作业记录以了解问题的出处，因为在执行 SQL 语句的作业中并没有正在执行真正的工作。

当以 SQL 服务器方式运行时，只可以使用 SQL 接口来在落实控制下执行工作。可以使用嵌入式 SQL 或调用级别接口（CLI）。将把单个线程中通过嵌入式 SQL 执行的所有连接路由至相同的后端作业。这将允许单个落实请求落实所有连接的工作，就好像它位于没有以 SQL 服务器方式运行的作业中一样。通过 CLI 执行的每个连接都将路由至单独的作业。CLI 要求为每个连接执行的工作单独落实或回滚。

当以 SQL 服务器方式运行时，不能在落实控制下执行下列操作:

- 以非 SQL 接口的接口执行的记录更改
- 对 DDM 文件的更改
- 对 API 落实资源的更改

不能在以 SQL 服务器方式运行的作业中直接启动落实控制。有关 SQL 服务器方式的更多信息，请参阅“数据库”主题中的下列页:

- 以 SQL 服务器方式运行 DB2 CLI
- 以 SQL 服务器方式启动 DB2 CLI
- 以服务器方式运行 DB2 CLI 的限制

启动落实控制

要启动落实控制，使用 STRCMTCTL（启动落实控制）命令。

注意: 落实控制不需要由 SQL 应用程序启动。当 SQL 隔离级别不是 *NONE 时，SQL 在连接时隐式启动落实控制。

当您使用 STRCMTCTL 命令时，可以指定以下内容:

在随后的主题中说明了这些参数。

落实锁定级别
在 STRCMTCTL 命令上使用 LCKLVL 参数指定锁定级别。对于因为落实定义而在落实控制下打开和放置的数据库文件而言，您所指定的级别将成为它们缺省的记录锁定级别。有关更多信息，请参阅落实锁定级别。

落实通知对象
使用 NTFY 参数来指定通知对象。通知对象是消息队列、数据区域或数据库文件，如果某特定落实定义未正常结束，则通知对象将包含标识此落实定义成功完成的最后一项事务的信息。有关更多信息，请参阅落实通知对象

落实作用域参数
使用 CMTSCOPE 参数来指定落实作用域。当启动落实控制时，系统将创建落实定义。落实作用域参数标识落实定义的作用域。缺省值是将落实定义的作用域限定为执行启动落实控制请求的程序的激活组。落实定义的作用域也可以是作业。

缺省日志参数

启动落实控制时可以指定缺省日志。您可能由于下列原因会使用缺省日志:

- 想要捕获事务日志项。这些项可以帮助您分析哪些资源与事务相关联的历史记录。它们不会用于应用和除去已记录的更改。省略日志项（OMTJRNE）参数确定系统是否写入事务项。
- 想要提高在路由步骤中关闭文件然后再次打开它们的作业的性能。如果关闭分配给不是缺省日志的日志的所有文件，则将从路由步骤中除去关于该日志的所有系统信息。如果稍后打开分配给该日志的文件，则必须再次创建关于该日志的所有信息。系统将保留关于使用落实定义的缺省日志的信息，而不管分配给该日志的任何资源是否活动。

落实文本参数

当显示有关为作业启动的落实定义的信息时，使用 TEXT 参数来标识与落实定义相关联的特定文本。如果未指定文本，则系统将提供缺省文本描述。

省略日志项参数

如果指定缺省日志以提高性能，则可以使用 OMTJRNE 参数来防止系统写入事务日志项。让系统写入事务项将明显增加日志接收器的大小并且在落实和回滚操作期间性能会降低。

当正在设置和测试落实控制环境或新的应用程序时，事务项将很有用。

在下列情形下，不管 OMTJRNE 参数值如何，都将向缺省日志写入事务项:

- 在落实或回滚操作期间发生系统错误。
- 对参与事务的资源进行了手工更改，并且该更改导致启发式混合状态。有关启发式混合状态的描述，请参阅两阶段落实控制的事务的状态。此类型的手工更改称为启发式决策。

可以使用关于哪些资源参与事务的信息来确定在这些情况下执行怎样的操作。

可使用日志项信息查找程序列出特定于事务（落实控制）日志项的数据。

如下主题中有关于启动落实控制的更多信息。

- 落实通知对象
- 落实锁定级别

落实通知对象

通知对象是消息队列、数据区或数据库文件，它包含标识为特定落实定义最后成功完成的事务的信息（如果该落实定义未正常结束的话）。用来标识落实定义的上一成功事务的信息由**落实标识**提供，该标识将落实操作与一组特定的可落实资源更改相关联。

仅当落实定义没有正常结束时，才将落实定义的上一成功事务的落实标识放入通知对象中。此信息可以用来帮助确定应用程序的处理在何处结束，以便可以再次启动应用程序。

对于独立磁盘池，通知对象与落实定义必须存在于同一个独立磁盘池或独立磁盘池组中。如果将落实定义移至另一个独立磁盘池或独立磁盘池组，则通知对象也必须驻留在这一其它独立磁盘池或独立磁盘池组上。如果落实定义异常结束，则会更新这一其它独立磁盘池或独立磁盘池组上的通知对象。如果在这一其它独立磁盘池或独立磁盘池组上找不到此通知对象，则更新会失败，且出现消息 CPF8358。

如果记录的资源参与当前事务并且以落实标识执行落实操作，则将把落实标识放入落实日志项（日志码和 C CM 的项类型）中，该落实日志项将该特定事务标识为正在落实的事务。将把包含落实标识的落实日志项发送至与参与事务的资源相关联的每个日志。

下表显示如何指定落实标识和它的最大大小。如果落实标识超出了它的最大大小，则在将其写入通知对象时将截断它。

语言	操作	落实标识中的最大字符数
CL	COMMIT 命令	3000[1]
ILE RPG*	COMIT 操作码	4000[1]
PLI。	PLICOMMIT 子例程	4000[1]
ILE C*	_Rcommit 函数	4000[1]
ILE COBOL*	COMMIT 动词	不支持
SQL	COMMIT 语句	不支持
注意: [1]如果通知对象是数据区，则最大大小是 2000 个字符。		

当以落实标识更新通知对象时，将按如下所示更新它:

数据库文件
如果将数据库文件用作通知对象，则将把落实标识添加至文件的末尾。将把任何现有的记录保留在文件中。因为几个用户或作业可以同时更改记录，所以文件中的每个落实标识都包含唯一的信息以将数据与失败的作业和落实定义相关联。可记录执行此功能的文件。

数据区
如果数据区用作通知对象，当落实标识置于该数据区中时，将替换该数据区的全部内容。如果多个用户或作业正在使用同一程序，则只有来自未正常结束的最后落实定义的落实标识将会位于该数据区中。因此，单个数据区通知对象有可能不生成用于再次启动应用程序的正确信息。要解决此问题，对于每个工作站用户或作业的每个落实定义使用单独的数据区。

消息队列
如果将消息队列用作通知对象，则将把消息 CPI8399 发送至消息队列。将落实标识放入消息 CPI8399 的二级文本中。关于将数据库文件用于通知对象，每个落实标识的内容唯一地标识作业的特定落实定义，以便可以再次启动应用程序。

有关使用通知对象的示例，请参阅示例: 使用通知对象来启动应用程序。

落实锁定级别

对于因为落实定义而在落实控制下打开和放置的数据库文件而言，您在启动落实控制（STRCMTCTL）命令上为 LCKLVL 参数指定的值将成为它们缺省的记录锁定级别。当打开本地数据库文件时，不能覆盖缺省记录锁定级别。但是，由 SQL 访问的数据库文件在对其发出第一个 SQL 语句时使用生效的当前 SQL 隔离级别。落实控制的注意事项和限制说明了对象和记录级别更改的注意事项。

根据您的需要、允许的等待时间和最常使用的释放过程来指定锁定级别。

下列描述只适用于在落实控制下打开的文件:

***CHG 锁定级别**
如果要保护已更改的记录不被同时运行的其它作业更改，则使用此值。对于在落实控制下打开的文件，将在事务持续期间保持锁定。对于不是在落实控制下打开的文件，保持记录锁定仅限于从读取记录到完成更新操作这段时间。

***CS 锁定级别**
使用此值来保护已更改的和已检索的记录不被同时运行的其它作业更改。未更改的已检索记录只在释放它之前或检索了另一记录之前受保护。

*CS 锁定级别确保其它作业不能读取此作业已读取的记录以进行更新。此外，该程序不能读取另一作业中以记录锁定类型 *UPDATE 锁定的记录以进行更新，直到该作业访问另一记录为止。

***ALL 锁定级别**
使用此值来保护处于落实控制下的已更改记录和已检索记录不被同时运行在落实控制下的其它作业更改。已检索或更改的记录将受保护，直到下一次执行落实或回滚操作为止。

*ALL 锁定级别确保其它作业不能访问此作业已读取的记录以进行更新。这与正常锁定协议是不同的。当将锁定级别指定为 *ALL 时，如果一条记录由另一作业中的记录锁定类型 *UPDATE 锁定时，不能访问未读取以进行更新的记录。

下表显示在落实控制下和不在落实控制下的文件的记录锁定的持续时间。

请求	LCKLVL 参数	锁定持续时间	锁定类型
只读	无落实控制	无锁定	无
	*CHG	无锁定	无
	*CS	从读取到下一次读取、落实或回滚	*READ
	*ALL	从读取到落实或回滚	*READ
读取以进行更新，然后更新或删除[1]	无落实控制	从读取到更新或删除	*UPDATE
	*CHG	从读取到更新或删除	*UPDATE
		然后从更新或删除到下一个落实或回滚[2]	*UPDATE
	*CS	从读取到更新或删除	*UPDATE
		然后从更新或删除到下一个落实或回滚[2]	*UPDATE
	*ALL	从读取到更新或删除	*UPDATE
		然后从更新或删除到下一个落实或回滚[2]	
读取以进行更新然后释放[1]	无落实控制	从读取到释放	*UPDATE
	*CHG	从读取到释放	*UPDATE
	*CS	从读取到释放、落实或回滚	*UPDATE
		然后从释放到下一个读取、落实或回滚	*UPDATE
	*ALL	从读取到释放、落实或回滚	*UPDATE
		然后从释放到下一个落实或回滚	
添加	无落实控制	无锁定	无
	*CHG	从添加到落实或回滚	*UPDATE
	*CS	从添加到落实或回滚	*UPDATE
	*ALL	从添加到落实或回滚	*UPDATE

请求	LCKLVL 参数	锁定持续时间	锁定类型
直接写入	无落实控制	直接写入的持续时间	*UPDATE
	*CHG	从直接写入到落实或回滚	*UPDATE
	*CS	从直接写入到落实或回滚	*UPDATE
	*ALL	从直接写入到落实或回滚	*UPDATE

注意:

[1]如果在读取以进行更新操作之后，但是在更新、删除或释放记录之前执行落实或回滚操作，则将在落实或回滚操作期间解锁该记录。落实或回滚一完成，就不再对记录进行保护。

[2]如果删除了记录，但是尚未对事务发出落实或回滚，则删除的记录不会保持锁定。如果相同或不同的作业尝试按键读取已删除的记录，则该作业将接收到“记录未找到”的指示。但是，如果该文件的唯一键控访问路径存在，则直到落实事务，才会允许另一作业插入或更新具有与已删除记录相同的唯一键值的记录。

当锁定级别为 *CS 或 *ALL 时，将在未读取以进行更新的记录上获取记录锁定类型 *READ。此锁定类型将阻止其它作业读取记录以进行更新，但是不会阻止只读操作正在访问的记录。

在更新、删除、添加或读取以进行更新的记录上获取记录锁定类型 *UPDATE。此锁定类型将阻止其它作业读取记录以进行更新，并阻止运行在落实控制下具有记录锁定级别 *CS 或 *ALL 的作业访问记录，即使是进行只读操作。

未正在使用落实控制的程序可以读取由另一作业锁定的记录，但是不能读取这些记录以进行更新，而不管为 LCKLVL 参数指定的值。

当为激活组或作业启动落实控制时，为落实定义指定的锁定级别只适用于与该特定落实定义相关联的打开记录。

注意:

*CS 和 *ALL 锁定级别值防止您检索当前具有来自另一作业的暂挂更改的记录。但是，*CS 和 *ALL 锁定级别值不会保护您使用运行在某一激活组（当前具有来自运行在同一作业内另一激活组中的的暂挂更改的激活组）中的程序检索记录。

在同一作业内，只要使用同一落实定义再次访问记录，程序就可以更改已经在当前事务内更改的记录。当使用作业级别落实定义时，可从运行于正在使用作业级别落实定义的任何激活组内的程序执行对已更改记录的访问。

结束落实控制

可以使用结束落实控制（ENDCMTCTL）命令结束作业级别或激活组级别的落实定义的落实控制。发出 ENDCMTCTL 命令向系统指示将要结束执行请求的程序正在使用的落实定义。ENDCMTCTL 命令只结束作业的一个落实定义，作业的所有其它落实定义保持不变。

如果激活组级别落实定义已结束，则除非已经对该作业启动了作业级别的落实定义，否则在该激活组中运行的程序不再能够在落实控制下进行更改。如果作业级别落实定义是活动的，则运行于刚结束落实控制的激活组中的程序可立即使用它。

如果作业级别落实定义已结束，则运行于作业中正在使用作业级别落实定义的任何程序都不再能够在落实控制下进行更改，除非首先使用 STRCMTCTL 命令再次启动落实控制。

在发出 ENDCMTCTL 命令之前，对于要结束的落实定义必须满足下列条件:

- 必须先关闭在落实控制下对要结束的落实定义打开的所有文件。当结束作业级别落实定义时，这包括在落实控制下由运行于任何激活组（正在使用作业级别落实定义的激活组）中的任何程序打开的所有文件。
- 必须首先使用 QTNRMVCR API 除去要结束的落实定义的所有 API 落实资源。当结束作业级别落实定义时，这包括由运行于任何激活组（正在使用作业级别落实定义的激活组）中的任何程序添加的所有 API 落实资源。
- 必须断开与要结束的落实定义相关联的远程数据库的连接。
- 必须使用正确的同步级别正常结束与落实定义相关联的所有受保护对话。

如果正在结束交互式作业中的落实控制并且与该落实定义相关联的一个或多个可落实资源具有暂挂更改，则会将查询消息 CPA8350 发送至用户，询问是落实暂挂更改、回滚暂挂更改还是取消 ENDCMTCTL 请求。

如果正在结束批处理作业中的落实控制，并且与要结束的落实定义相关联的一个或多个已关闭文件具有暂挂更改，则会回滚该更改并且发送消息:

- 如果只注册了本地资源，则将发送 CPF8356 消息
- 如果只注册了远程资源，则将发送 CPF835C 消息
- 如果同时注册了本地资源和远程资源，则将发送 CPF83E4 消息

如果为正结束的落实定义定义了通知对象，则可能更新它。

当将 API 注册为上一代理程序的激活组正在结束时，会调用该 API 的出口程序以接收落实或回滚决定。在这种情况下，即使该激活组正在正常结束，仍可从 API 出口程序返回回滚请求。因此，可能不执行隐式落实操作。

当成功结束落实定义之后，也就执行了所有必需的恢复（如果有的话）。没有对与刚结束的落实定义相关联的落实资源执行附加恢复。

在结束了落实定义之后，就可以对运行于激活组中的程序再次启动作业级别或激活组级别的落实定义。仅当尚未对作业启动作业级别的落实定义时才能启动它。

虽然在激活组中运行的程序可以多次启动和结束落实定义，但是重复启动和结束操作所需的系统资源量可能会导致作业性能和整个系统性能降低。因此，如果稍后调用的程序将使用落实定义，则建议使落实定义保持活动。

有关系统如何更新通知对象的更多信息，请参阅对通知对象的更新。

系统启动的落实控制结束

系统可以结束落实控制、执行隐式落实或回滚操作。有时，系统启动的落实控制结束是正常的。其它时间，落实控制以异常的系统或作业结束而结束。

以下页描述在哪些情况下落实控制以系统的结束而隐式结束，以及采取哪些操作（如果存在任何可以执行的操作的话）:

- 激活组结束期间的落实控制
- 隐式落实和回滚操作
- 正常路由步骤结束期间的落实控制
- 异常的系统或作业结束期间的落实控制
- 在落实控制结束后对通知对象的更新

- 在初始程序装入期间的落实控制恢复

激活组结束期间的落实控制

当激活组结束时系统自动结束激活组级别落实定义。如果激活组级别落实定义的暂挂更改存在并且激活组正在正常结束，则激活组在结束之前，系统将对落实定义执行隐式落实操作。否则，如果激活组正在异常结束，或者如果关闭在落实控制（其作用域限定为激活组）下打开的任何文件时系统遇到错误，则在结束激活组之前会对激活组级别的落实定义执行隐式回滚操作。

注意:

在 *JOB 或 *DFTACTGRP 落实定义的激活组结束处理期间*永远不会*执行隐式落实或回滚操作。这是因为 *JOB 和 *DFTACTGRP 落实定义永远不会因为激活组结束而结束。其实，这些落实定义是使用 ENDCMTCTL 命令显式结束或在作业结束时由系统结束的。

当激活组结束时，系统自动关闭其作用域限定为激活组的任何文件。这包括在落实控制下打开的其作用域限定为激活组的任何数据库文件。在可能对激活组级别落实定义执行的任何隐式落实操作之前会关闭这些文件。因此，在执行任何隐式落实操作之前，会首先将驻留在 I/O 缓冲区中的任何记录强制放入数据库。

作为可能执行的隐式落实或回滚操作的一部分，对与激活组级别落实定义相关联的每个 API 落实资源调用 API 落实和回滚出口程序。出口程序必须在 5 分钟内完成它的处理。在调用了 API 落实和回滚出口程序之后，系统将自动除去 API 落实资源。

如果对由于结束激活组而正结束的落实定义执行隐式回滚操作，则可能会更新通知对象（如果为该落实定义定义了通知对象的话）。有关系统更新通知对象的更多信息，请参阅对通知对象的更新。

隐式落实和回滚操作

通常，从使用支持落实控制的某个可用编程语言的应用程序启动落实或回滚操作。这些类型的落实和回滚操作称为**显式落实和回滚请求**。然而，在某些实例中，系统对落实定义启动落实或回滚操作。系统启动的落实和回滚操作称为**隐式落实和回滚请求**。

下面两个表显示当发生与具有暂挂更改的落实定义有关的事件时系统执行的操作。如果存在下列任何情况，则落实定义具有暂挂更改:

- 更新了任何可落实资源。
- 由于读取文件更改了文件位置，所以已读取在落实控制下打开的数据库文件。
- 落实定义具有 API 资源。由于对 API 资源执行的更改是由用户程序完成的，所以系统必须假定所有 API 资源都具有暂挂更改。

C CM（落实操作）日志项和 C RB（回滚操作）日志项指示操作是显式的还是隐式的。

下表显示当作业正常或异常结束时系统根据以下情况执行的操作:

- 事务的状态。
- 落实定义的“结束时执行的操作”作业值。
- API 资源是否是上一代理程序。

状态	上一代理程序 API	如果选择 Endjob[1] 选项时执行的操作	落实或回滚操作
RST	不适用	不适用	如果落实定义不与 X/Open 全局事务相关联，则执行隐式回滚。 如果落实定义与 X/Open 全局事务相关联，则发生下列情况: • 如果事务分支状态不是活动的（S1），则不执行操作并且事务分支保留在同一状态。 • 如果事务分支状态是活动的（S1），则执行隐式回滚。
PIP	不适用	不适用	如果落实定义不与 X/Open 全局事务相关联，则执行隐式回滚。 如果落实定义与 X/Open 全局事务相关联，则事务分支处于“空闲”（S2）状态并且它保持为“空闲”（S2）状态。
PRP	不适用	等待	如果落实定义不与 X/Open[2] 全局事务相关联，则发生下列情况: • 启动再同步以接收来自落实操作的启动程序的决定。 • 执行返回的落实或回滚的决定。认为它是显式操作。

状态	上一代理程序 API	如果选择 Endjob[1] 选项时执行的操作	落实或回滚操作
PRP	不适用	C	如果落实定义不与 X/Open[2] 全局事务相关联，则执行隐式落实操作。
		R	如果落实定义不与 X/Open 全局事务相关联，则执行隐式回滚操作。 如果落实定义与 X/Open 全局事务相关联，则发生下列情况: • 如果启动事务的作业结束，则在 XA TM 落实或回滚该事务之前，它将保持处于准备状态。在这种情况下，XA 事务分支状态将保持在“准备”（S3）状态。 • 如果结束正在将事务的工作路由至的 SQL 服务器作业，则会隐式执行强制回滚。在这种情况下，XA 事务分支状态将更改为“启发式完成”（S5）状态。
CIP	不适用	不适用	执行显式落实操作。
LAP	否	等待	1. 将对上一代理程序的再同步用于检索落实或回滚的决定。
			2. 执行返回的落实或回滚的决定。认为它是显式操作。
LAP	是	等待	1. 调用上一代理程序以检索落实或回滚决定。
			2. 执行落实或回滚操作。认为它是显式操作。
LAP	不适用	C	执行隐式落实操作。
		R	执行隐式回滚操作。
CMT	不适用	不适用	此落实定义以及任何下游位置的落实操作已经完成。落实操作已完成。
VRO	不适用	不适用	对本地和远程代理程序执行只读表决。所有下游代理程序也必须已执行只读表决。不需要操作。
RBR	不适用	不适用	需要回滚操作。执行显式回滚操作。

状态	上一代理程序 API	如果选择 Endjob[1] 选项时执行的操作	落实或回滚操作
注意: [1]可以使用更改落实选项（QTNCHGCO）API 更改结束作业时的操作选项。 [2]如果落实定义与 X/Open 全局事务相关联，则发生下列情况: • 如果启动事务的作业结束，则在 XA TM 落实或回滚该事务之前，它将保持处于准备状态。在这种情况下，XA 事务分支状态将保持在“准备”（S3）状态。 • 仅对于事务作用域锁定，如果结束正在将事务的工作路由至的 SQL 服务器作业，则会隐式执行强制回滚。在这种情况下，XA 事务分支状态将更改为“启发式完成”（S5）状态。			

下表显示当结束激活组时系统执行的操作并且仅适用于具有作业作用域锁定的事务。系统操作基于以下内容:

- 事务的状态。（当激活组结束时通常会对其进行复位（RST））。
- 激活组如何结束 – 正常或异常。
- API 资源是否是上一代理程序。

注意: 如果将 API 资源注册为上一代理程序，则这将把对落实或回滚决定的控制权赋予上一代理程序。决定的结果被视为显式操作。

状态	上一代理程序 API	结束类型	落实或回滚操作
RST	否	正常	执行隐式落实操作。如果存在受保护对话，则落实定义将成为落实操作的根启动程序。
RST	否	异常	执行隐式回滚。
RST	是	正常	调用 API 出口程序。落实或回滚操作由 API 确定。
RST	是	异常	调用 API 出口程序。落实或回滚操作由 API 确定。

正常路由步骤结束期间的落实控制

当正常结束路由步骤时，系统将结束作业的所有落实定义。

注意: 下列项只适用于具有作业作用域锁定的落实定义。

路由步骤按下列某方式正常结束:

- 批处理作业的正常结束。
- 交互式作业的正常注销。
- 重新路由作业（RRTJOB）、转移作业（TFRJOB）或转移批处理作业（TFRBCHJOB）命令将结束当前路由步骤并启动新的路由步骤。

任何其它结束路由步骤的情况被认为是异常的，并且将由作业记录中的作业完成消息 CPF1164 中的非零完成代码识别。

在路由步骤结束期间并且在结束落实定义之前，如果落实定义具有暂挂更改，则系统将执行隐式回滚操作。这包括调用与落实定义相关联的每个 API 落实资源的 API 落实和回滚出口程序。出口程序必须在 5 分钟内完成它的处理。在调用了 API 落实和回滚出口程序之后，系统将自动除去 API 落实资源。

如果为落实定义定义了通知对象，则可以更新它。请参阅对通知对象的更新以获取关于系统更新通知对象的更多信息。

异常系统或作业结束期间的落实控制

本主题仅适用于具有作业作用域锁定的落实定义。当作业异常结束时系统会结束该作业的所有落实定义。这些落实定义在结束作业处理期间结束。如果系统异常结束，则系统会结束在系统异常结束时已启动并正被所有活动作业使用的所有落实定义。这些落实定义是作为系统异常结束之后下一次 IPL 期间执行的数据库恢复处理的一部分结束的。

注意:
恢复落实定义是指由于电源故障、硬件故障或操作系统或许可内码中的故障引起的系统或作业的异常结束。一定不能使用“异常结束作业”（ENDJOBABN）命令来强制异常结束作业。异常结束会导致您正在结束以部分落实或回滚的作业的活动事务的暂挂更改。下一次 IPL 可能会尝试恢复使用 ENDJOBABN 命令结束的作业的任何部分事务。

系统在 IPL 期间对使用 ENDJOBABN 命令结束的作业执行的落实控制恢复的结果是不确定的。产生这种不确定性的原因是作业异常结束时释放了落实资源的所有锁定。由部分事务引起的任何暂挂更改可用于其它作业。然后，这些暂挂更改会导致其它应用程序对数据库作出额外的错误更改。类似地，稍后执行的任何后续的 IPL 恢复对在异常结束作业之后应用程序所作的更改有不利影响。例如，在 IPL 恢复期间，可能会删除 SQL 表作为对暂挂创建表的回滚操作。然而，在异常结束作业之后，其它应用程序可能已将一些行插入到表中。

系统对在异常的作业结束期间或异常的系统结束之后的下一次 IPL 期间正在结束的落实定义执行下列操作:

- 在结束落实定义之前，如果落实定义具有暂挂更改，则系统执行隐式回滚操作，除非在落实操作中间中断了落实定义的处理。如果在落实操作中间结束，则可回滚、再同步或落实事务，这取决于事务的状态。请参阅隐式落实和回滚操作。执行隐式回滚操作或完成落实操作处理过程包括为每个与落实定义相关联的 API 落实资源调用 API 落实及回滚出口程序。在调用了 API 落实和回滚出口程序之后，系统将自动除去 API 落实资源。

  注意:
  当事务处于不确定状态（事务状态为 LAP 或 PRP）时结束作业会导致数据库中存在不一致情况（可能在一个或多个系统上落实更改并且在其它系统上回滚更改）。
  - 如果*结束作业时的操作*落实选项是 COMMIT，则如果结束作业，就落实在此系统上的更改，而不必考虑是否落实或回滚参与事务的其它系统上的更改。
  - 如果*结束作业时的操作*落实选项是 ROLLBACK，则如果结束作业，就回滚在此系统上的更改，而不必考虑是否落实或回滚参与事务的其它系统上的更改。
  - 如果*结束作业时的操作*落实选项是 WAIT，则对拥有落实或回滚决定权的系统完成再同步之前，作业将不会结束。要使作业在再同步完成之前结束，必须作出启发式决策并且必须取消再同步。

  建议**不**要在长时间运行回滚期间异常结束作业或系统。这样做会导致在作业结束时（或者如果结束了系统在下一次 IPL 期间）发生另一个回滚。后续的回滚将重复初始回滚执行的工作并且要花更长的时间来运行。
- 如果为落实定义定义了通知对象，则可更新它。有关系统更新通知对象的更多信息，请参阅对通知对象的更新。

如果进程在落实控制结束之前结束并且受保护对话仍然活动，则可能需要落实或回滚落实定义。操作基于落实定义的“状态”选项和“结束作业时的操作”选项。

异常结束之后初始程序装入期间的落实控制恢复

当在系统异常结束之后执行初始程序装入（IPL）时，系统尝试恢复系统结束时所有活动的落实定义。同样，当您使独立磁盘池联机时，系统尝试恢复与该独立磁盘池有关的当独立磁盘池脱机或异常结束时所有活动的落实定义。恢复由在 IPL 期间由系统启动的数据库服务器作业执行。数据库服务器作业由系统启动来处理不能或一定不能由其它作业执行的工作。

数据库服务器作业命名为 QDBSRVnn，其中 nn 是一个两位数。数据库服务器作业数取决于系统的大小。同样，独立磁盘池或独立磁盘池组的数据库服务器作业的名称是 QDBSxxxVnn，其中 xxx 是独立磁盘池编号，nn 是一个两位数。例如，QDBS035V02 可以是独立磁盘池 35 的数据库服务器作业的名称。

两阶段落实控制的事务的状态显示系统（根据故障发生时事务的状态）执行的操作。对于 PRP 和 LAP 这两种状态，系统操作是不确定的。

注意:

- 下列项只适用于具有作业作用域锁定的落实定义。
- 事务管理器使用 XA API 而不是本主题中描述的再同步过程恢复与 XA 事务（锁定范围限于作业或事务）相关联的落实定义。

系统在它执行与参与事务的其它位置再同步之前不能确定执行什么操作。此再同步是在 IPL 或联机操作完成之后执行的。

系统使用数据库服务器作业来执行此再同步。需要恢复的落实定义与数据库服务器作业相关联。在 IPL 期间，系统获取所有记录锁定以及在系统结束之前落实定义持有的其它对象锁定。这些锁定在再同步完成之前以及可落实或回滚资源之前对于保护本地落实资源是必需的。

消息发送至数据库服务器作业的作业记录以指示与远程位置的再同步状态。如果事务处于不确定状态，则在可以落实或回滚本地资源之前必须完成与拥有事务决定权的位置的再同步。

当对事务作出决定时，以下消息可能会发送至数据库服务器作业的作业记录。

CPI8351
正在回滚 &1 暂挂更改

CPC8355
作业 &19/&18/&17 的落实定义 &8 的 IPL 后恢复已完成。

CPD835F
作业 &19/&18/&17 的落实定义 &8 的 IPL 恢复失败。

可能还会发送与恢复有关的其它消息。这些消息发送至历史（QHST）记录。如果发生错误，消息还会发送至 QSYSOPR 消息队列。

通过使用 iSeries(TM) 导航器、通过显示数据库服务器作业的作业记录或通过使用“使用落实定义”（WRKCMTDFN）命令来确定恢复的进度。虽然“iSeries 导航器”和“使用落实定义”屏幕允许您强制系统落实或回滚，但是您只能将它用作上次恢复。如果您预料参与事务的所有位置最终将返回至操作，则必须允许系统使它们再同步。这确保数据库的完整性。

对通知对象的更新

对于通知对象，认为以下项是未落实的更改:

- 在落实控制下进行的记录更新。
- 在落实控制下删除的记录。
- 在落实控制下对本地 DDL 对象进行的对象级别更改。
- 对在落实控制下打开的数据库文件所执行的读操作。这是因为当执行回滚操作时，将把文件位置带回至上一落实边界。如果在落实控制下执行读操作，则将更改文件位置，因此对于落实定义存在未落实的更改。
- 具有下列已添加的其中一个资源的落实定义通常认为具有未落实的更改:
  - API 落实资源
  - 远程 Distributed Relational Database Architecture(TM)（DRDA(R)*）资源
  - “分布式数据库管理体系结构”（DDM）资源
  - LU 6.2 资源

  这是因为系统不知道何时对与这些类型的资源相关联的对象进行真正的更改。可落实资源的类型具有关于如何添加和使用这些类型的资源的更多信息。

系统对通知对象进行更新并且根据落实定义可以结束的下列方式:

- 如果作业正常结束并且不存在未落实的更改，则系统不会将上一成功落实操作的落实标识放入通知对象中。
- 如果在激活组结束时，对激活组级别落实定义执行了隐式落实操作，则系统不会将上一成功落实操作的落实标识放入通知对象中。

注意: 从不会对 *DFTACTGRP 或 *JOB 落实定义执行隐式落实操作

- 如果在落实定义的第一个成功落实操作前，系统、作业或激活组异常结束，则系统不会更新通知对象，因为没有最后落实标识。要区分此条件和正常程序完成，程序必须在完成落实定义的第一个成功落实操作之前，以特定项更新通知对象。
- 如果在至少一个成功落实操作后发生异常作业结束或异常系统结束，则系统将把该落实操作的落实标识放入通知对象中。如果上一成功落实操作没有指定落实标识，则不会更新通知对象。对于异常作业结束，将为每个落实定义（对于作业是活动的落实定义）执行此通知对象处理。对于异常系统结束，将为每个落实定义（对于系统上的所有作业是活动的落实定义）执行此通知对象处理。
- 如果发生下列所有情况，则系统将以该落实定义的最后成功落实操作的落实标识来更新通知对象:
  - 非缺省激活组结束。
  - 对激活组级别的落实定义执行隐式回滚操作。
  - 对于该落实定义至少已经执行了一个成功的落实操作。

  如果上一成功落实操作没有指定落实标识，则不会更新通知对象。如果激活组正在异常结束或者当关闭在落实控制（其作用域限定为该激活组）下打开的文件时发生错误，则将对激活组级别落实定义执行隐式回滚操作。有关将数据库文件的作用域限定为激活组以及激活组可以如何结束的更多信息，请参阅正在使用的 ILE 语言的参考书。
- 如果在作业正常结束时存在未落实的更改，并且至少执行了一个成功的落实操作，则将把上一成功落实操作的落实标识放入通知对象中并且将回滚未落实的更改。如果上一成功落实操作没有指定落实标识，则不会更新通知对象。将对在作业结束时对作业为活动的落实定义执行此通知对象处理。有关在作业正常结束期间执行的功能的更多信息，请参阅主题正常路由步骤结束期间的落实控制。

- 如果在运行 ENDCMTCTL 命令时存在未落实的更改，则仅当上一成功落实操作指定了落实标识时，才会更新通知对象:
  - 对于批处理作业，将回滚未落实的更改并且将把上一成功落实操作的落实标识放入通知对象中。
  - 对于交互式作业，如果对查询消息 CPA8350 的响应将回滚更改，则将回滚未落实的更改并且将把上一成功落实操作的落实标识放入通知对象中。
  - 对于交互式作业，如果对查询消息 CPA8350 的响应将落实更改，则系统将提示您输入要使用的落实标识并且将落实更改。将把在提示屏幕上输入的落实标识放入通知对象中。
  - 对于交互式作业，如果对查询消息 CPA8350 的响应将取消 ENDCMTCTL 请求，则将保留暂挂的更改并且不会更新通知对象。

管理事务和落实控制

下面是可以执行以管理落实控制的任务。

显示落实控制信息
此信息具有可以执行的任务以显示系统上关于所有事务的信息以及显示关于与这些事务相关联的作业的信息。

优化落实控制的性能
此信息具有可以执行的任务以最小化落实控制对系统性能的影响。

显示落实控制信息

可以使用 iSeries(TM) 导航器来显示有关系统上所有事务（逻辑工作单元）的信息。还可以查看有关与事务相关联的作业（如果有的话）的信息。

注意: 这些显示操作不显示 SQL 应用程序的隔离级别。

要显示信息，按如下所示继续:

1. 在 **iSeries 导航器**窗口中，展开想要使用的服务器。
2. 展开**数据库**。
3. 展开想要使用的系统。
4. 展开**事务**。

注意: 要查看与 X/Open 全局事务相关联的事务，展开**全局事务**。要查看 DB2(R) UDB 管理的事务，展开**数据库事务**。

5. 展开**全局事务**或**数据库事务**。

此屏幕显示以下内容:

- 工作单元标识
- 工作单元状态
- 作业
- 用户
- 编号
- 正在进行的再同步
- 落实定义

联机帮助提供有关所有状态屏幕以及每个屏幕上的字段的信息。

还可以使用“iSeries 导航器”来显示以下信息:

- 显示事务的已锁定对象
- 显示与事务相关联的作业
- 显示事务的资源状态
- 显示事务属性

显示事务的已锁定对象

只能显示具有事务作用域锁定的全局事务的已锁定对象。

要显示事务的已锁定对象:

1. 在 **iSeries(TM) 导航器**窗口中，展开想要使用的服务器。
2. 展开**数据库**。
3. 展开想要使用的系统。
4. 展开**事务**。
5. 展开**全局事务**。
6. 右键单击想要使用的事务并选择**已锁定对象**。

显示与事务相关联的作业

要显示与事务相关联的作业:

1. 在 **iSeries(TM) 导航器**窗口中，展开想要使用的服务器。
2. 展开**数据库**。
3. 展开想要使用的系统。
4. 展开**事务**。
5. 展开**全局事务**或**数据库事务**。
6. 右键单击想要使用的事务并选择**作业**。

对于具有作业作用域锁定的数据库事务和全局事务，将显示与事务相关联的作业的列表。

对于具有限于事务的锁定的全局事务，将显示此事务对象所附属的作业列表或正等待此事务对象附属其上的作业列表。

显示事务的资源状态

要显示事务的资源状态:

1. 在 **iSeries(TM) 导航器**窗口中，展开想要使用的服务器。
2. 展开**数据库**。
3. 展开想要使用的系统。
4. 展开**事务**。
5. 展开**全局事务**或**数据库事务**。
6. 右键单击想要使用的事务并选择**资源状态**。

显示事务属性

要显示事务属性:

1. 在 **iSeries(TM) 导航器**窗口中，展开想要使用的服务器。
2. 展开**数据库**。
3. 展开想要使用的系统。
4. 展开**事务**。
5. 展开**全局事务**或**数据库事务**。
6. 右键单击想要使用的事务并选择**属性**。

优化落实控制的性能

使用落实控制需要可以影响系统性能的资源。几个因素影响关于落实控制的系统性能。以下是不影响性能的因素、降低性能的因素和提高性能的因素。

不影响性能的因素

打开文件

如果打开文件而不指定落实打开选项，则不会使用附加系统资源，即使已经启动了落实定义。有关指定落实打开选项的更多信息，请参阅适当的高级语言参考手册。

降低性能的因素

日志记录

记录文件需要系统资源。但是，在大多数情况下，日志记录在具有落实控制时的性能比没有落实控制时要好。如果只指定后映像，则落实控制将在落实控制生效时将它更改为前映像和后映像。这通常是出于空间而不是性能方面的考虑。有关日志记录的更多信息，请参阅日志管理主题。

落实操作

如果在事务期间对已记录的资源进行了任何更改，则事务的每次落实将把两个项添加至与这些资源相关的每个日志中。对于大量的小事务，添加的项的数目可以显著增加。可能想要将日志接收器放入独立于日志的磁盘池中。

回滚操作

因为落实控制必须回滚记录在数据库中的暂挂更改，所以每当发生回滚时都需要附加的系统资源。另外，如果记录更改正在暂挂，则回滚操作将使附加项添加至日志中。

启动落实控制（STRCMTCTL）和结束落实控制（ENDCMTCTL）命令

每次分别使用 STRCMTCTL 和 ENDCMTCTL 命令启动和结束落实定义时，系统都会产生额外开销。避免对每个事务使用 STRCMTCTL 和 ENDCMTCTL 命令。只在必要时使用它们。可在交互式作业的开始建立落实定义并在作业期间使用它。

对落实控制事务使用多个日志

对于两阶段落实，可以将在落实控制下打开的文件记录至多个日志。但是，使用多个日志将需要附加的系统资源来管理落实定义。使用多个日志也会使恢复更加复杂。

记录锁定

记录锁定可以影响其它应用程序。在特定作业内锁定的记录数将增加用于该作业的整体系统资源。需要访问相同记录的应用程序必须等待事务结束。

请求 SEQONLY(*YES)

如果请求 SEQONLY(*YES) 选项（通过隐式使用 OVRDBF 命令或应用程序尝试使用 SEQONLY(*YES)）并且在具有 LCKLVL(*ALL) 的落实控制下打开文件仅用于输入，则将把该选项更改为 SEQONLY(*NO)。此选项会影响输入文件的性能，因为不会将记录分块。

当活动时保存处理活动时，请求对数据库文件进行记录级别更改

如果落实定义处于落实边界并且活动时保存操作正运行在另一作业中，则可能会延迟在落实控制下对数据库文件进行记录级别更改的请求。当将文件与某些保存请求对象记录至相同日志时，有可能发生此情况。

注意: 当作业由于活动时保存检查点处理而正在被保持时，“使用活动作业”（WRKACTJOB 命令）屏幕上的“状态”列显示 CMTW（落实等待）。

当活动时保存处理活动时，落实或回滚更改

当活动时保存操作正运行在另一作业中时，可能会延迟落实边界处的落实或回滚操作。当先前已将 API 落实资源添加至落实定义时会发生此情况，除非使用 QTNADDCR API 添加 API 资源并且*允许正常保存处理*字段的值为 Y。

因为在落实或回滚请求期间将保持作业，并且因为每次只可以对单个落实定义执行一个落实或回滚请求，所以具有已添加的 API 落实资源的多个落实定义的作业会阻止活动时保存操作的完成。

注意: 如果以部分事务功能部件使用新的保存，则可以不结束落实定义而保存对象。

当活动时保存处理活动时，请求对象级别的更改

如果落实定义处于落实边界并且活动时保存操作正运行在另一作业中，则可能会延迟在落实控制下进行的对象级别更改的请求。当针对对象所在的库正在运行活动时保存操作时而进行对象级别的更改时，会发生此情况。例如，当活动时保存操作正在针对库 MYSQLLIB 运行时，可能会延迟在落实控制下对库 MYSQLLIB 中的表 MYTBL 的创建 SQL 表操作。

注意: 如果等待时间超过 60 秒，则发送查询消息 CPA8351 以询问用户是继续等待还是取消操作。

使用 QTNADDCR API 添加 API 资源

如果落实定义处于落实边界并且活动时保存操作正运行在另一作业中，则可能会延迟使用 QTNADDCR API 添加 API 落实资源的请求。

注意:

1. 如果等待时间超过 60 秒，则发送查询消息 CPA8351 以询问用户是继续等待还是取消操作。
2. 如果*允许正常保存处理*字段的值是 Y，则这不会适用于使用 QTNADDCR API 添加的 API 资源。

提高性能的因素

使用缺省日志

在落实定义活动时，如果在落实控制下关闭和重新打开所有文件，则使用缺省日志可以帮助提高性能。但是，使用具有 OMTJRNE(*NONE) 的缺省日志将降低落实和回滚操作的性能。

选择上一代理程序

因为在落实操作期间需要系统和上一代理程序之间有更少的交互作用，所以当选择上一代理程序时将提高性能。但是，如果在落实操作期间发生通信故障，则落实操作不会完成，直到再同步完成为止，而不

管对结果选项的等待时间值如何。这种故障并不多见，但是此选项允许应用程序编写者考虑使用户无限期等待再同步完成的负面影响（在故障的确发生时）。将成功落实操作期间由上一代理优化所提供的性能提高与此负面影响相权衡。通常，此考虑对于交互式作业比对于批处理作业更有意义。

缺省值是上一代理程序允许由系统选择，但是用户可以使用 QTNCHGCO API 修改此值。

不使用等待结果选项
当远程资源在落实控制下时，将在“等待结果”选项设置为 N（否）并且所有远程系统都支持假定的异常终止时，提高性能。当建立与远程系统的第一个连接时，系统为 DRDA(R) 和 DDM 应用程序将“等待结果”选项设置为 N。APPC 应用程序必须显式设置“等待结果”选项，或者将使用缺省值 Y。

选择“确认以忽略”选项
当选择“确认以忽略”选项时，将提高性能。有关此选项的进一步信息，请参阅两阶段落实的落实定义：对“确认以忽略”进行说明。

选择“只读表决”选项
当选择“只读表决”选项时，将提高性能。有关此选项的进一步信息，请参阅两阶段落实的落实定义：允许只读表决。

可以执行以提高性能的任务为:

- 最小化锁定
- 管理事务大小

最小化锁定

最小化记录锁定的典型方法是释放记录锁定。（如果指定了 LCKLVL(*ALL)，则此方法不适用）。例如，单个文件维护应用程序通常执行下列任务:

- 显示要更改的记录标识的提示。
- 检索请求的记录。
- 显示记录。
- 允许工作站用户进行更改。
- 更新记录。

在大多数情况下，记录已锁定，不允许请求的记录通过更新访问它。对于正在等待该记录的另一作业，可能会超出记录等待时间。为了避免在工作站用户正在考虑更改时锁定记录，在从数据库检索记录后（在记录屏幕显示前）释放该记录。然后需要在更新前再次访问该记录。如果在释放记录和再次访问记录之间更改了记录，则必须通知工作站用户。通过保存初始记录的一个或多个字段然后在检索该记录后将它们与同一记录中的字段相比较，程序可以确定是否更改了该记录，如下所示:

- 在记录中使用更新计数字段并仅在更新前向该字段添加 1。程序将保存初始值并在再次检索该记录时将该值与字段中的值相比较。如果发生了更改，则将通知工作站用户并且将再次显示记录。只有在发生更新时，才会更改更新计数字段。当工作站用户正在考虑更改时将释放记录。如果使用此技术，则必须将它用于更新文件的每个程序中。
- 保存整个数据记录的内容并将在下一次检索记录时，将其与该记录相比较。

在上面的两种情况中，操作的顺序阻止在 RPG（其中，在主记录和显示文件中使用相同的字段名）中简单使用外部描述数据。不能使用相同的字段名（在 RPG 中），因为当再次检索记录时，将覆盖工作站用户的更改。

可以通过将记录数据移至数据结构来解决此问题，或者如果使用 DDS 关键字 RTNDTA，则可以继续使用外部描述数据。RTNDTA 关键字允许您的程序重新读取屏幕上的数据并且操作系统不必将数据从屏幕移动到程序中。这将允许程序执行下列操作:

1. 提示记录标识。
2. 从数据库中检索请求的记录。
3. 释放记录。
4. 保存用来确定是否已更改记录的字段。
5. 显示记录并等待工作站用户响应。

如果工作站用户更改屏幕上的记录，则程序将使用以下顺序:

1. 再次从数据库中检索记录。
2. 比较保存的字段以确定是否已更改数据库记录。如果更改了数据库记录，则程序将释放记录并在显示记录时发送一条消息。
3. 通过运行带有 RTNDTA 关键字的读操作来从屏幕中检索记录并更新数据库记录中的记录。
4. 继续下一个逻辑提示，因为如果工作站用户取消请求的话，将没有要释放的附加记录。

LCKLVL(*CHG) 和 LCKLVL(*CS) 可在此情况下使用。如果使用 LCKLVL(*ALL)，则必须通过使用落实或回滚操作来释放记录锁定。

有关锁定的更多信息，请参阅检测死锁。

管理事务大小

对于此讨论，事务是交互式的。（落实控制也可用于批处理应用程序，通常可以将批处理应用程序认为是一系列事务。许多相同的注意事项也适用于批处理应用程序，将在批处理应用程序的落实控制中讨论它。）

对于每个与事务相关联的日志，您可以在事务期间最多锁定 500 000 000 条记录。可以通过使用“查询选项文件”（QAQQINI）来减少此限制。使用“更改查询属性”（CHGQRYA）命令的 QRYOPTLIB 参数来指定要使用的作业的“查询选项文件”。使用“查询选项文件”中的 COMMITMENT_CONTROL_LOCK_LEVEL 值作为作业的锁定限制。

当选择记录的锁定级别时，应考虑事务的大小。使用大小来确定事务结束前将记录锁定多长时间。您必须决定落实控制的落实或回滚操作是仅限于使用一次 Enter 键，还是在该事务中可以多次使用 Enter 键。

注意: 事务越短，就能越早地继续并完成等待启动活动时保存检查点处理的作业。

例如，对于订单输入应用程序，客户有可能在单个订单内订购多个商品，这需要记录订单详细信息并针对订单中每一项进行库存主记录更新。如果将事务定义为整个订单并且每次使用 Enter 键都将订购一个商品，则将在整个订单的持续期间锁定订单中涉及的所有记录。因此，经常使用的记录（如库存主记录）可能长时间被锁定，防止其它工作进行。如果使用子文件以单个 Enter 键输入所有商品，则将最小化整个订单的锁定的持续时间。

通常，必需最小化锁定的数目和持续时间，以便几个工作站用户可以访问相同的数据而不必长时间等待。当用户在屏幕上输入数据时不要保持锁定便可以做到这一点。一些应用程序可能不需要多个工作站用户访问相同的数据。例如，在每个客户有许多开放商品记录的现金过账应用程序中，通常做法是锁定并延迟所有记录，直到工作站用户完成现金过账获得收条为止。

如果工作站用户对一个事务多次按 Enter 键，则可能在许多段中执行事务。例如:

- 第一段是工作站用户在其中请求信息的查询。
- 第二段是对工作站用户想要完成整个事务的确认。
- 第三段是对受影响的记录的检索和更新。

此方法允许将记录锁定以限制为单次使用 Enter 键。

通常，基于显示信息做出决定的应用程序中使用这种先执行查询的方法。例如，在机票预定应用程序中，客户在决定搭乘哪趟班机之前有可能想了解有哪些班次、转接班机以及座位安排情况。一旦客户做出决定，就会输入事务。如果事务失败（该航班已满），则可以使用回滚功能并且将输入不同的请求。如果从第一个查询就锁定记录，直到作出决定为止，则另一个预定职员程序将始终等待，直到其它事务完成为止。

方案和示例: 落实控制

以下是落实控制的方案和示例。该方案以较高的级别显示 JKL 玩具公司如何实现落实控制以跟踪他们本地数据库上的事务。

这些示例提供落实控制的样本代码。“练习题目”是实现落实控制的 RPG 程序。它包括一个逻辑流，显示方法的每个步骤正在发生的内容。

以下三个示例说明如何使用落实控制在发生异常系统结束后启动应用程序。

方案

- 方案: 落实控制

示例

- 落实控制的练习题目
- 落实控制的练习题目的逻辑流
- 示例: 使用事务记录文件来启动应用程序
- 示例: 使用通知对象来启动应用程序
- 示例: 使用标准处理程序来启动应用程序

注意: 请阅读代码示例免责以获取重要的法律信息。

方案: 落实控制

JKL 玩具公司使用落实控制来保护制造和库存的数据库记录。此方案详细地说明 JKL 玩具公司从库存部门提货至制造部门过程中如何使用落实控制。

有关 JKL 玩具公司的网络环境的描述，请参阅方案: 日志管理。随后的方案显示落实控制在他们的生产服务器 JKLPROD 上如何工作。

此方案用两个示例说明使用落实控制的优点。第一个示例显示公司的库存程序（程序 A）在不使用落实控制的情况下可能会如何工作以及可能会发生的问题。第二个示例显示程序在使用落实控制的情况下如何工作。

JKL 玩具公司在他们的服务器 JKLPROD 上使用库存应用程序（程序 A）。程序 A 使用两条记录。一条记录跟踪存储在库房中的商品。另一条记录跟踪从库房中取出并且用于生产的商品。

不使用落实控制的程序 A

假定以下应用程序不使用落实控制。系统锁定要读取以进行更新的记录。下列步骤描述了应用程序如何跟踪二极管（将二极管从库房中取出并将它传输至支出帐户）：

- 程序 A 锁定并检索库房记录。（如果另一个程序锁定了记录，则此操作可能需要等待。）
- 程序 A 锁定并检索生产记录。（此操作可能也需要等待。）现在，程序 A 锁定了两条记录，没有其它程序可以更改这两条记录。
- 程序 A 更新库房记录。此操作导致该记录被释放，使它现在可由任何其它程序读取以进行更新。
- 程序 A 更新生产记录，此操作导致该记录被释放，使它现在可由任何其它程序读取以进行更新。

若不使用落实控制，需要解决一个问题才能使此程序在所有情况下都能正常工作。例如，如果由于作业或系统故障而使程序 A 不能更新两条记录，就会出现问题。在这种情况下，两个文件就会不一致 – 从库房记录中除去了二极管，却没有将它们添加到生产记录中。使用落实控制允许您确保完成事务中所涉及的所有更改，或者如果中断了事务的处理，则确保将文件返回到它们的原始状态。

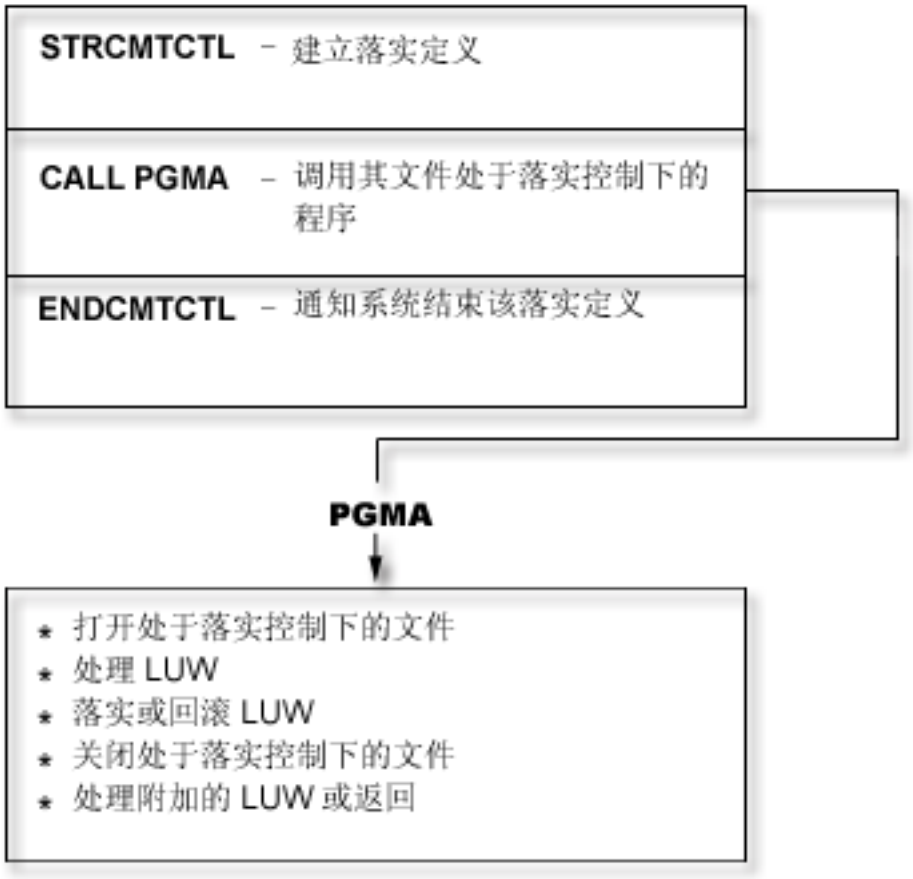
使用落实控制的程序 A

如果使用落实控制，则前面的示例更改如下:

1. 启动落实控制。
2. 程序 A 锁定并检索库房记录。（如果另一个程序锁定了该记录，则此操作可能需要等待。）
3. 程序 A 锁定并检索生产记录。（此操作可能也需要等待。）现在，程序 A 锁定了两条记录，没有其它程序可以更改这两条记录。
4. 程序 A 更新库房记录并且落实控制保持对该记录的锁定。
5. 程序 A 更新生产记录并且落实控制保持对该记录的锁定。
6. 程序 A 落实事务。对库房记录和生产记录的更改在文件中是永久更改。更改记录在日志中，日志假定更改将出现在磁盘上。落实控制释放对两条记录的锁定。现在，任何其它程序就能够读取这些记录以进行更新。

由于在落实事务之前落实控制保持两条记录的锁定，所以不可能发生更新了一条记录而不更新另一条记录的情况。如果在落实事务之前发生路由步骤或系统故障，则系统除去（回滚）已进行的更改以便将文件更新为上次落实事务时的状态。

对于文件将处于落实控制下的每个路由步骤，会发生下图显示的各个步骤:



在落实控制下执行的操作将记录到日志。启动落实控制日志项出现在落实控制下的第一个文件打开项之后。这是因为第一个文件打开项确定哪个日志用于落实控制。然后使用第一个打开操作中的日志项来检查后续的打开操作以确保所有文件都在使用同一个日志。

当发生作业故障或系统故障时，会将落实控制下的资源更新为落实边界。如果启动了事务但在路由步骤结束之前未完成事务，则系统回滚该事务并且在路由步骤结束之后该事务不会出现在文件中。如果在事务完成之前系统异常结束，则系统回滚该事务并且在许可内码的后续成功初始程序装入（IPL）之后该事务不会出现在文件中。不管回滚在什么时候发生，都会将颠倒的项放在日志中。

例如，假定 JKL 公司仓库中有 100 个二极管。从仓库中取出 20 个用于制造，那么新的剩余数量是 80。数据库同时更新前映像（100）和后映像（80）日志项。

假定在记录各项之后但在到达落实点或回滚点之前系统异常结束。在 IPL 之后，系统读取日志项并更新相应的数据库记录。此更新产生颠倒更新的两个日志项：第一个项是前映像（80），第二个项是后映像（100）。

当在异常结束之后成功完成 IPL 时，系统除去（或回滚）未落实的任何数据库更改。在前面的示例中，系统从库房记录中除去更改，原因是该事务的日志中没有落实操作。在这种情况下，就将库房记录的前映像放入文件中。日志包含回滚更改以及发生了回滚操作的指示。

落实控制的练习题目

此练习题目将帮助您了解落实控制和它的要求。下列步骤假定您已熟悉 OS/400(R) 许可程序和数据文件实用程序（DFU），并且假定您已阅读了此主题。逻辑流将帮助您进一步了解落实控制的这一练习程序。

注意: 请阅读代码示例免责以获取重要的法律信息。

在开始此问题前，执行下列操作:

- 为此练习题目创建特殊的库。在指示信息中，将该库称为 CMTLIB。替换您在其中看到 CMTLIB 的库的名称。
- 创建源文件和作业描述。

执行下列步骤:

1. 创建名为 ITMP（商品主文件）的物理文件。此文件的数据描述规范（DDS）为:

```
10    A    R ITMR
20    A      ITEM     2
30    A      ONHAND   5   0
40    A    K ITEM
```

2. 创建名为 TRNP（事务文件）的物理文件。此文件用作事务记录文件。此文件的 DDS 是:

```
10    A    R TRNR
20    A      QTY      5  0
30    A      ITEM     2
40    A      USER     10
```

3. 创建名为 TRNL（事务逻辑）的逻辑文件。此文件用于帮助再次启动应用程序。*USER* 字段是类型 LIFO 序列。此文件的 DDS 是:

```
10                        LIFO
20    A     R TRNR        PFILE (TRNP)
30    A     K USER
```

4. 输入 STRDFU 命令，并为 ITMP 文件创建名为 ITMU 的 DFU 应用程序。接受 DFU 在应用程序定义期间提供的缺省值。
5. 输入命令 CHGDTA ITMU 并输入 ITMP 文件的下列记录:

商品	现有数量
AA	450
BB	375
CC	4000

6. 使用 F3 结束该程序。此项提供该程序将对其执行操作的一些数据。
7. 创建 “CL 程序项进程”（ITMPCSC），如下所示:

```
PGM
DCL &USER *CHAR LEN(10)
RTVJOBA USER(&USER)
CALL ITMPCS PARM(&USER)
ENDPGM
```

   这是调用 ITMPCS 程序的控制程序。它检索用户名并将其传递给处理程序。此应用程序假定使用唯一的用户名。

8. 从 DDS 创建名为 ITMPCSD 的显示文件，如下所示。

   有两种格式，第一种用于基本提示屏幕，第二种允许操作员复查最后输入的事务。ITMPCS 程序使用此显示文件。

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A          R PROMPT
  2.00      A                                      CA03(93 'End of program')
  3.00      A                                      CA04(94 'Review last')
  4.00      A                                      SETOFF(64 'No rcd to rvw')
  5.00      A                                  1  2'INVENTORY TRANSACTIONS'
```

```
 6.00      A                                      3  2'Quantity'
 7.00      A            QTY           5  0I    +1
 8.00      A  61                                   ERRMSG('Invalid +
 9.00      A                                       quantity' 61)
10.00      A                                     +5'ITEM'
11.00      A            ITEM          2    I   +1
12.00      A  62                                   ERRMSG('Invalid +
13.00      A                                       Item number' 62)
14.00      A  63                                   ERRMSG('Rollback +
15.00      A                                       occurred' 63)
16.00      A  64                                24  2'CF4 was pressed and +
17.00      A                                       there are no +
18.00      A                                       transactions for +
19.00      A                                       this user'
20.00      A                                       DSPATR(HI)
21.00      A                                   23  2'CF4 Review last +
22.00      A                                       transaction'
23.00      A          R REVW
24.00      A                                    1  2'INVENTORY TRANSACTIONS'
25.00      A                                     +5'REVIEW LAST TRANSACTION'
26.00      A                                    3  2'Quantity'
27.00      A            QTY           5  0     +1EDTCDE(Z)
28.00      A                                     +5'Item'
29.00      A            ITEM          2        +1
```

9. 请研究落实控制的练习程序的逻辑流中提供的逻辑流。
10. 输入 STRSEU 命令并输入源，如下所示:

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

 1.00     FITMP    UF  E           K        DISK
 2.00     F*                                        KCOMIT
 3.00     FTRNP    O   E                    DISK
 4.00     F*                                        KCOMIT
 5.00     FTRNL    IF  E           K        DISK
 6.00     F            TRNR                         KRENAMETRNR1
 7.00     FITMPCSD  CF E                    WORKSTN
 8.00     C* Enter parameter with User name for -TRNP- file
 9.00     C           *ENTRY    PLIST
10.00     C                     PARM           USER   10
11.00     C           LOOP      TAG
12.00     C                     EXFMTPROMPT
13.00     C* Check for CF3 for end of program
14.00     C   93                DO                          End of Pgm
15.00     C                     SETON                    LR
16.00     C                     RETRN
17.00     C                     END
18.00     C* Check for CF4 for review last transaction
19.00     C   94                DO                          Review last
20.00     C* Check for existence of a record for this user in -TRNL- file
21.00     C           USER      CHAINTRNR1               64    Not found
22.00     C   64                GOTO LOOP
23.00     C                     EXFMTREVW
24.00     C                     GOTO LOOP
25.00     C                     END
26.00     C* Access Item record
27.00     C           ITEM      CHAINITMR                62    Not found
28.00     C* Handle -not found- Condition
29.00     C   62                GOTO LOOP
30.00     C* Does sufficient quantity exist
31.00     C           ONHAND    SUB  QTY       TEST    50   61  Minus
32.00     C* Handle insufficient quantity
33.00     C    61               DO
34.00     C* Release Item record which was locked by the CHAIN for update
35.00     C                     EXCPTRLSITM
36.00     C                     GOTO LOOP
37.00     C                     END
```

```
38.00      C* Change ONHAND and update the Item record
39.00      C                    Z-ADDTEST      ONHAND
40.00      C                    UPDATITMR
41.00      C* Test for Special Simulation Conditions
42.00      C           ITEM     IFEQ 'CC'
43.00      C*      Simulate program need for rollback
44.00      C           QTY      IFEQ 100
45.00      C                    SETON                     63    Simult Rlbck
46.00      C*                   ROLBK
47.00      C                    GOTO LOOP
48.00      C                    END
49.00      C*      Simulate an abnormal program cancellation by Div by zero
50.00      C*        Operator Should respond -C- to inquiry message
51.00      C           QTY      IFEQ 101
52.00      C                    Z-ADD0         ZERO    30
53.00      C           TESTZ    DIV  ZERO      TESTZ   30       Msg occurs
54.00      C                    END
55.00      C*      Simulate an abnormal job cancellation by DSPLY.
56.00      C*        Operator Should System Request to another job
57.00      C*         and cancel this one with OPTION(*IMMED)
58.00      C           QTY      IFEQ 102
59.00      C           'CC=102' DSPLY                           Msg occurs
60.00      C                    END
61 00      C                    END                           ITEM=CC
62.00      C* Write the -TRNP- file
63 00      C                    WRITETRNR
64.00      C* Commit the update to -ITMP- and write to -TRNP-
65.00      C*                   COMIT
66.00      C                    GOTO LOOP
67.00      OITMR    E               RLSITM
```

11. 输入 CRTRPGPGM 命令从在先前步骤中输入的源创建程序 ITMPCS。
12. 输入命令 CALL ITMPCSC，按 Enter 键并按 F4。一个消息将说明没有此操作员的项。
13. 输入以下数据以查看程序是否正常运作:

数量	商品
3	AA
4	BB

14. 按 F4。此复查屏幕将显示最后输入的商品 BB。输入以下数据:

数量	商品
5	FF（产生了无效商品编号消息。）
9000	BB（产生了数量不足错误消息。）
100	CC（产生了回滚消息。）
102	CC（必须发生 RPG DSPLY 操作。按 Enter 键。）
101	CC（程序必须显示查询消息，说明已发生或结束被零除的情况（这取决于作业属性 INQMSGRPY 的设置）。如果显示查询消息，则输入 C 以取消 RPG 程序，然后输入 C 以取消后续查询上的 CL 程序。这将模拟意外的错误状态。）

15. 输入“显示数据”命令 DSPDTA ITMP。

    查看是否正确更新了记录 AA 和 BB。这些值必须为 AA = 447、BB = 371 和 CC = 3697。请注意，CC 中发生数量减少，但是未编写事务记录。
16. 为落实控制创建日志接收器。使用“创建日志接收器”（CRTJRNRCV）命令来在 CMRLIB 库中创建名为 RCVR1 的日志接收器。指定至少为 5000KB 的阈值。如果系统具有足够的空间，则建议使用较大的阈值，这样可以最大化生成新的日志接收器之间的时间以最小化太频繁更改日志对性能的影响。

17. 创建落实控制的日志。使用“创建日志”（CRTJRN）命令来在 CMTLIB 库中创建名为 JRNTEST 的日志。因为此日志只用于落实控制，所以指定 MNGRCV(*SYSTEM) DLTRCV(*YES)。对于 JRNRCV 参数，指定在步骤 16 中创建的日志接收器。
18. 使用具有参数 FILE(CMTLIB/ITMP CMTLIB/TRNP) JRN(CMTLIB/JRNTEST) 的“启动日志物理文件”（STRJRNPF）命令以记录要用于落实控制的文件。

    IMAGES 参数使用缺省值 *AFTER，这意味着只有记录的后映像更改显示在日志中。文件 ITMP 和 TRNP 现在已经启动日志记录。

    通常，在启动日志记录后保存文件。不能将记录的更改应用于不与日志项具有相同 JID 的已恢复文件。因为此练习题目不需要您应用已记录的更改，所以可以跳过保存已记录的文件。
19. 输入命令 CALL ITMPCSC 并输入下列事务:

数量	商品
5	AA
6	BB

    按 F3 来结束程序。
20. 输入“显示日志”命令: DSPJRN CMTLIB/JRNTEST。

    请注意显示在日志中的项。与程序执行的顺序相同的项顺序（R UP = 更新 ITMP，其后为 R PT = 添加至 TRNP 的记录）出现在日志中。这是因为将通过物理文件 TRNP 定义逻辑文件并且系统将覆盖 RPG 缺省值。如果不存在逻辑文件，则将使用 RPG 假定 SEQONLY(*YES)，并且将显示 PT 项的块，原因是记录将保留在 RPG 缓冲区中直到该块充满为止。
21. 更改 CL 程序 ITMPCSC，如下所示（显示有星号的语句是新语句）:

```
     PGM
     DCL &USER *CHAR LEN(10)
     RTVJOBA USER(&USER)
*    STRCMTCTL LCKLVL(*CHG)
     CALL ITMPCS PARM(&USER)
*    MONMSG MSGID(RPG9001) EXEC(ROLLBACK)
*    ENDCMTCTL
     ENDPGM
```

    STRCMTCTL 命令设置落实控制环境。LCKLVL 字指定读取记录以进行更新，但是未更新的记录可在事务期间释放。MONMSG 命令处理任何 RPG 脱离消息，并在 RPG 程序异常结时执行 ROLLBACK。ENDCMTCTL 命令结束落实控制环境。
22. 删除现有 ITMPCSC 程序并再次创建它。
23. 更改 RPG 程序以除去语句 2.00、4.00、46.00 和 65.00 处的注释符号。源现在已准备好与落实控制一起使用。
24. 删除现有的 ITMPCS 程序并再次创建它。程序现在已准备好在落实控制下操作。
25. 输入命令 CALL ITMPCSC 和下列事务:

数量	商品
7	AA
8	BB

26. 使用“系统请求”并请求显示当前作业的选项。当出现“显示作业”屏幕时，选择选项 16 以请求显示落实控制状态。

    请注意屏幕上的值。因为两个落实语句运行在程序中，所以必须有两个落实。

27. 按 F9 以查看落实控制下文件的列表以及每个文件的活动数。
28. 返回至程序并按 F3 来结束它。
29. 输入 DSPJRN CMTLIB/JRNTEST 并注意文件的项和落实控制的特殊日志项:

C BC	发生了 STRCMTCTL 命令。
C SC	启动落实周期。每当事务中的第一个数据库操作导致插入、更新或删除记录（作为落实控制的一部分）时，都将发生此操作。
C CM	已发生落实操作。
C EC	发生了 ENDCMTCTL 命令。

即使最初已为日志请求 IMAGES(*AFTER)，落实控制前映像和后映像（R UB 和 R UP 类型）也会自动发生。

30. 输入命令 CALL ITMPCSC 和下列事务:

数量	商品
12	AA
100	CC（这是模拟应用程序需要使用回滚的条件。将回滚由 RPG 语句 40.00 更新的 ITMP 文件中的 CC 记录。）

31. 按 F4 以确定最后输入的事务。

最后落实的事务是商品 AA 的项。

32. 使用“系统请求”并请求“显示当前作业”选项。当出现“显示作业”屏幕时，请求显示落实控制状态。

请注意屏幕上的值以及回滚如何更改它们。

33. 返回至程序。
34. 返回至基本提示屏幕并通过按 F3 来结束程序。
35. 输入命令 DSPJRN CMTLIB/JRNTEST。

请注意出现在日志中用于回滚项（C RB 项）的附加项。当回滚 ITMP 记录时，将把三个项放入日志中。这是因为对落实控制下数据库文件的任何更改都将产生前（R BR）和后（R UR）项。

36. 显示具有日志码 R 的项，这些项类型是 UB、UP、BR 和 UR。使用选项 5 以显示全部项。因为数量字段是压缩十进制形式的，所以使用 F11 来请求十六进制显示。请注意下列内容:
    - UB 记录中 ITMP 记录的“当前的”值
    - UP 记录如何减少“当前的”值
    - BR 记录如何与 UP 记录相同
    - UR 记录如何返回最初为 UB 记录显示的值

    最后的项是回滚结束的 RB 项。
37. 输入命令 CALL ITMPCSC，按 Enter 键并按 F4。请注意最后输入的事务。
38. 输入下列事务:

数量	商品
13	AA

数量	商品
101	CC（这是模拟导致程序结束的意外错误状态的状态。通过将字段除以 0 而进行模拟。程序将显示查询消息或结束（这取决于作业属性 INQMSGRPY 的设置）。如果出现查询消息，则输入 C 以结束程序。因为已更改 CL 程序来监视 RPG 程序错误，所以本应该发生的第二个查询没有发生。）

39. 输入命令 DSPJRN CMTLIB/JRNTEST。

    发生了相同类型的回滚处理，但是此次回滚是由 CL 程序（不是 RPG 程序）中 MONMSG 命令的 EXEC 参数导致的。显示这两个 RB 项以查看是由哪个程序引起的。
40. 输入命令 WRKJOB 并写下全限定作业名供以后使用。
41. 输入命令 CALL ITMPCSC 并输入以下事务:

数量	商品
14	AA
102	CC（必须对外部消息队列发生 RPG DSPLY 操作。使用“系统请求”键并选择系统请求菜单上的选项 1 以传输至第二个作业。）

42. 注册至第二个作业并重新建立环境。
43. 输入命令 ENDJOB 并指定较早标识的全限定作业名和 OPTION(*IMMED)。这将模拟异常作业或系统结束。
44. 等待大约 30 秒，输入命令 CALL ITMPCSC 并按 F4。

    请注意最后落实的事务。它必须是较早输入的商品 AA。
45. 返回至基本提示屏幕并通过按 F3 来结束程序。
46. 输入命令 DSPJRN CMTLIB/JRNTEST。

    发生了相同类型的回滚处理，但是此次回滚是由系统导致而不是由某个程序导致的。RB 项是由程序 QWTPITPP（工作管理异常结束程序）编写的。

您现在已经使用了落实控制的基本功能。可以在您的应用程序上继续进行落实控制或尝试一些其它功能，如:

- 使用通知对象
- 锁定只以 LCKLVL(*ALL) 读取的记录
- 锁定同一文件中具有 LCKLVL(*ALL) 的多条记录

示例: 使用事务记录文件来启动应用程序

此示例提供如何使用事务记录文件在异常结束后启动应用程序的样本代码和指示信息。

注意: 请阅读代码示例免责以获取重要的法律信息。

当未使用通知对象时，**事务记录文件**用来在系统或作业发生故障后再次启动应用程序。事务记录文件通常用于交互式应用程序以总结事务的影响。

例如，在订单输入应用程序中，对每个订购的商品通常将记录写入事务记录文件中。该记录包含订购的商品、数量和价格。在可支付帐户应用程序中，对于将接收费用的每个帐号都将把一条记录写入事务记录文件中。此记录通常包含诸如帐号、收取的金额和供应商的信息。

在事务记录文件已存在于其中的许多应用程序中，工作站用户可以请求关于最后输入的事务的信息。通过将落实控制添加至事务记录文件已存在于其中的应用程序，您可以完成下列任务:

- 确保将数据库文件更新至落实边界。
- 再次简化对事务的启动。

如果将事务记录文件用于在落实控制下再次启动应用程序，则您必须能够唯一地标识工作站用户。如果在系统上使用唯一的用户概要文件名称，则可以将该概要文件名称置于事务日志记录的字段中。可将此字段用作文件的键。

以下示例假定订单库存文件正用于执行事务并且假定事务记录文件已经存在。该程序将执行下列操作:

1. 提示工作站用户输入数量和商品编号。
2. 更新生产主文件（PRDMSTP）中的数量。
3. 向事务记录文件（ISSLOGL）写入记录。

如果现有库存数量不足，则程序将拒绝该事务。因为商品编号、描述、数量、用户名和日期将被写入事务记录文件中，所以工作站用户可以询问程序在何处中断了该数据项。

物理文件 PRDMSTP 的 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R PRDMSTR                   TEXT('Master record')
  2.00      A            PRODCT         3          COLHDG('Product' 'Number')
  3.00      A            DESCRP        20          COLHDG('Description')
  4.00      A            ONHAND         5  0       COLHDG('On Hand' 'Amount')
  5.00      A                                      EDTCDE(Z)
  6.00      A          K PRODCT
```

ISSLOGP 使用的物理文件 ISSLOGP 的 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R ISSLOGR                   TEXT('Product log record')
  2.00      A            PRODCT         3          COLHDG('Product' 'Number')
  3.00      A            DESCRP        20          COLHDG('Description')
  4.00      A            QTY            3  0       COLHDG('Quantity')
  5.00      A                                      EDTCDE(Z)
  6.00      A            USER          10          COLHDG('User' 'Name')
  7.00      A            DATE           6  0       EDTCDE(Y)
  8.00      A                                      COLHDG('Date')
```

逻辑文件 ISSLOGL 的 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A                                      LIFO
  2.00      A          R ISSLOGR                   PFILE(ISSLOGP)
  3.00      A          K USER
```

在程序中使用的显示文件 PRDISSD 的 DDS

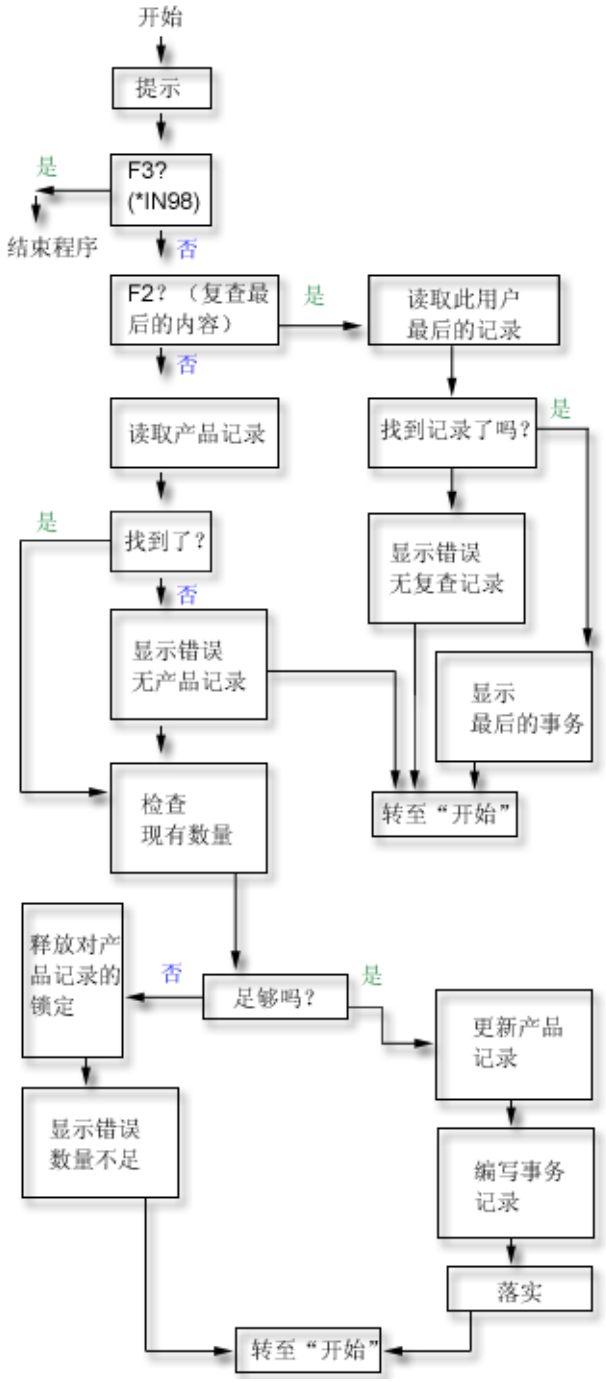
```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

 1.00       A                                      REF(ISSLOGP)
  2.00      A          R PROMPT
  3.00      A                                      CA03(98 'End of program')
 4.00       A                                      CA02(97 'Where am I')
 5.00       A                                   1 20'ISSUES PROCESSING'
  6.00      A                                   3  2'Quantity'
 7.00       A            QTY       R        I    +1
 8.00       A  62                                  ERRMSG('Not enough +
```

```
9.00       A                                        Qty' 62)
10.00      A                                      +6'Product'
11.00      A            PRODCT    R        I     +1
12.00      A  61                                    ERRMSG('No Product +
13.00      A                                        record found' 62)
14.00      A  55                                 15  2'No Previous record exists'
15.00      A                                     24  2'CF2 Last transaction'
16.00      A          R RESTART
17.00      A                                      1 20'LAST TRANSACTION +
18.00      A                                        INFORMATION'
19.00      A                                      5  2'Product'
20.00      A            PRODCT    R             +1
21.00      A                                      7  2'Description'
22.00      A            DESCRP    R             +1
23.00      A                                      9  2'Qty'
24.00      A            QTY       R             +1REFFLD(QTY)
```

在**程序流**中概括了此过程。

程序流



链接至描述

在更新了 PRDMSTP 文件并且将记录写入事务记录文件之后，指定 RPG COMMIT 操作码。因为每次对操作员的提示都表示新事务的边界，所以认为该事务是单个输入事务。

在调用用户名时将其传递至程序。事务记录文件的访问路径是以后进先出（LIFO）的顺序定义的，所以程序可以容易地访问最后输入的记录。

通过使用与标识在何处停止数据项的相同函数，工作站用户可以在系统或作业发生故障后再次启动程序。不需要向程序添加任何附加的代码。如果当前正在使用事务记录文件，但是并没有用其来查找您所处的位置，则将用户名添加至事务记录文件（假定用户名是唯一的）并在程序中使用此方法。

下面显示使用的 RPG 程序。以箭头（==>）标记落实控制所需的语句。

RPG 程序

```
 SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... .. 7 ..
 =>1.00      FPRDMSTP UP  E           K        DISK        KCOMIT
 =>2.00      FISSLOGL IF  E           K        DISK        KCOMIT
   3.00      PRDISSD  CP  E                    WORKSTN
   4.00                   *ENTRY    PLIST
   5.00                             PARM           USER    10
   6.00      C*
   7.00      C*  Initialize fields used in Trans Log Rcd
   8.00      C*
   9.00      C                     MOVE UDATE     DATE
  10.00      C*
  11.00      C*  Basic processing loop
 12.00      C*
  13.00      C           LOOP      TAG
  14.00      C                     EXFMTPROMPT
  15.00      C   98                GOTO END                       End of pgm
  16.00      C   97                DO                             Where am I
  17.00      C                     EXSR WHERE
  18.00      C                     GOTO LOOP
  19.00      C                     END
  20.00      C           PRODCT    CHAINPRDMSTR              61     Not found
  21.00      C   61                GOTO LOOP
  22.00      C           ONHAND    SUB  QTY      TEST    50    62   Less than
  23.00      C   62                DO                             Not enough
  24.00      C                     EXCPTRLSMST                    Release lock
  25.00      C                     GOTO LOOP
  26.00      C                     END
  27.00      C*
  28.00      C*  Update master record and output the Transaction Log Record
  29.00      C*
  30.00      C                     Z-ADDTEST     ONHAND
  31.00      C                     UPDATPRDMSTR
  32.00      C                     WRITEISSLOGR
=>33.00      C                     COMIT
  34.00      C                     GOTO LOOP
  35.00      C*
  36.00      C*  End of program processing
 37.00      C*
  38.00      C           END       TAG
  39.00      C                     SETON                      LR
  40.00      C*
  41.00      C* WHERE subroutine for "Where am I" requests
 42.00      C*
  43.00      C           WHERE     BEGSR
  44.00      C           USER      CHAINISSLOGL              55    Not found
  45.00      C  N55                EXFMTRESTART
  46.00      C                     ENDSR
  47.00      OPRDMSTR E                RLSMST
```

用来调用 RPG 程序 PRDISS 的 CL 程序

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00            PGM
  2.00            DCL          &USER *CHAR LEN(10)
  3.00            STRCMTCTL    LCKLVL(*CHG)
  4.00            RTVJOBA      USER(&USER)
  5.00            CALL         PRDISS PARM(&USER)
  6.00            MONMSG       MSGID(RPG9001) EXEC(ROLLBACK)
  7.00            ENDCMTCTL
  8.00            ENDPGM
```

要在此程序中使用落实控制，通常要指定锁定级别 *CHG。该记录由更改锁定，直到运行落实操作为止。注意，如果库存数量不足，则将显式释放该记录。（如果未在程序中显式释放记录，则当从文件中读取下一条记录以进行更新时释放它。）

在此示例中，使用锁定级别 *ALL 没有更多的优点。如果使用 *ALL，当数量不足时必须使用回滚或落实操作来释放该记录。

先前的代码是调用 RPG 程序 PRDISS 的 CL 程序。请注意 STRCMTCTL/ENDCMTCTL 命令的使用。将检索（RTVJOBA 命令）唯一的用户名并将其传递给程序。在示例: 使用标准处理程序来启动应用程序中描述了使用 MONMSG 命令来导致回滚的过程。

示例: 使用通知对象来启动应用程序

当程序在异常结束后启动时，它可以查找通知对象中的项。如果项存在，则该程序可以再次启动事务。当再次启动了事务后，该程序将清除通知对象以防止它再次启动同一事务。

以下是您可以使用通知对象的方式:

- 如果将落实标识放入数据库文件中，则查询此文件以确定在何处再次启动每个应用程序或工作站作业。
- 如果将落实标识放入特定工作站的消息队列中，将可以在工作站用户注册时向他们发送一条消息以通知他们上次落实的事务。
- 如果将落实标识放入具有键或用户名的数据库文件中，则该程序可以在启动时读取此文件。如果记录存在于该文件中，则再次启动该程序。程序可以将消息发送至工作站用户以标识上次落实的事务。程序将执行任何恢复。如果记录存在于数据库文件中，则程序将删除程序末尾的的记录。
- 对于批处理应用程序，可以将落实标识放入包含总量、交换机设置和再次启动应用程序所需的其它状态信息的数据区中。当启动应用程序时，它将访问数据区并验证存储在那里的值。如果应用程序正常结束，则将为下一次运行设置数据区。
- 对于批处理应用程序，可以将落实标识发送至消息队列。启动应用程序时运行的程序可以从该队列中检索消息并再次启动程序。

取决于您的应用程序的需要，再次启动应用程序存在几个技巧。在选择技巧时，请考虑以下内容:

- 当一个程序同时存在多个用户时，不能将单个数据区用作通知对象，因为在异常的系统结束后，每个用户的落实标识将在数据区中彼此覆盖。
- 关于删除通知对象中信息的设计必须能够处理在使用该信息后立刻发生故障的情形:
  - 如果立刻删除信息，则如果在处理中断事务前发生另一故障，该信息不会存在。
  - 直到成功处理中断事务后，才能删除通知对象中的信息。在此情况下，如果通知对象为数据库文件或消息队列，则通知对象中将存在多个项。
  - 如果存在多个项，则程序必须访问最后的记录。
- 通知对象不能用来向工作站用户提供上次落实的事务，因为只有在发生系统或作业故障时或如果在作业的正常结束时存在未落实的更改时，才会更新通知对象。
- 如果向工作站用户显示信息，则该信息必须是有意义的。要完成这一点，可以要求程序将通知对象中保留的代码转换为帮助用户再次启动的信息。
- 如果工作站用户需要再次启动的信息，则必须显示该信息。需要程序中的附加逻辑以防止再次显示不再有意义的信息。
- 如果通知对象为数据库文件，则单个通知对象和标准处理程序可以提供再次启动功能。该标准处理程序由需要再次启动能力的程序调用以最小化对每个个别程序的更改。

关于使用通知对象的示例代码，请参阅以下内容:

注意: 请阅读代码示例免责以获取重要的法律信息。

- 每个程序的唯一通知对象
- 所有程序的单个通知对象

示例: 每个程序的唯一通知对象

此主题提供使用唯一通知对象来重新启动每个程序的样本代码和指示信息。

对每个作业使用单个唯一的通知对象允许使用外部描述的落实标识，即使同一程序可能有多个用户。在下列示例中，将数据库文件用作通知对象并且只由此程序使用它。

该程序具有两个数据库文件（PRDMSTP 和 PRDLOCP），必须为库存收据更新这两个数据库文件。该程序使用的显示文件的名称是 PRDRCTD。将数据库文件 PRDRCTP 用作通知对象。将此通知对象对程序定义为文件并且还将该通知对象用作通知函数的数据结构的定义。

请参阅 物理文件 PRDMSTP 的 DDS 以查看物理文件 PRDMSTP 的 DDS。

注意: 请阅读代码示例免责以获取重要的法律信息。

物理文件 PRDLOCP 的 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00      A          R PRDLOCR                   TEXT('Location record')
  2.00      A            PRODCT         3          COLHDG('Product' 'Number')
  3.00      A            LOCATN         6          COLHDG('Location')
  4.00      A            LOCAMT         5  0       COLHDG('Location' 'Amount')
  5.00      A                                      EDTCDE(Z)
  6.00      A          K PRODCT
  7.00      A          K LOCATN
```

显示文件 PRDRCTD 的 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A                                      REF(PRDMSTP)
  2.00      A          R PROMPT
  3.00      A                                      CA03(98 'End of program')
  4.00      A                                      SETOFF(71 'RESTART')
  5.00      A                                   1  20'PRODUCT RECEIPTS'
  6.00      A                                    3  2'Quantity'
  7.00      A            QTY           3  0I     +1
  8.00      A                                     +6'Product'
  9.00      A            PRODCT   R        I     +1
 10.00      A  61                                  ERRMSG('No record +
 11.00      A                                      found in the +
 12.00      A                                      master file' 62)
 13.00      A                                     +6'Location'
 14.00      A            LOCATN   R        I     +1REFFLD(LOCATN PRDLOCP)
 15.00      A  62                                  ERRMSG('No record +
 16.00      A                                      found in the +
 17.00      A                                      location file' 62)
 18.00      A                                     9   2'Last Transaction'
 19.00      A  71                                 +6'This is restart +
 20.00      A                                        information'
 21.00      A                                      DSPATR(HI BL)
 22.00      A                                   12   2'Quantity'
 23.00      A                                   12  12'Product'
 24.00      A                                   12  23'Location'
 25.00      A                                   12  35'Description'
 26.00      A            LSTPRD   R          14  15REFFLD(PRODCT)
 27.00      A            LSTLOC   R          14  26REFFLD(LOCATN *SRC)
```

```
 28.00      A            LSTQTY    R          14   5REFFLD(QTY *SRC)
 29.00      A                                       EDTCDE(Z)
 30.00      A            LSTDSC    R          14  35REFFLD(DESCRP)
```

通知对象和外部描述数据结构（PRDRCTP）的 DDS

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      A                                       LIFO
  2.00      A                                       REF(PRDMSTP)
  3.00      A          R PRDRCTR
  4.00      A            USER          10
  5.00      A            PRODCT    R
  6.00      A            DESCRP    R
  7.00      A            QTY            3  0
  8.00      A            LOCATN    R               REFFLD(LOCATN PRDLOCP)
  9.00      A          K USER
```

程序按如下所示处理通知对象:

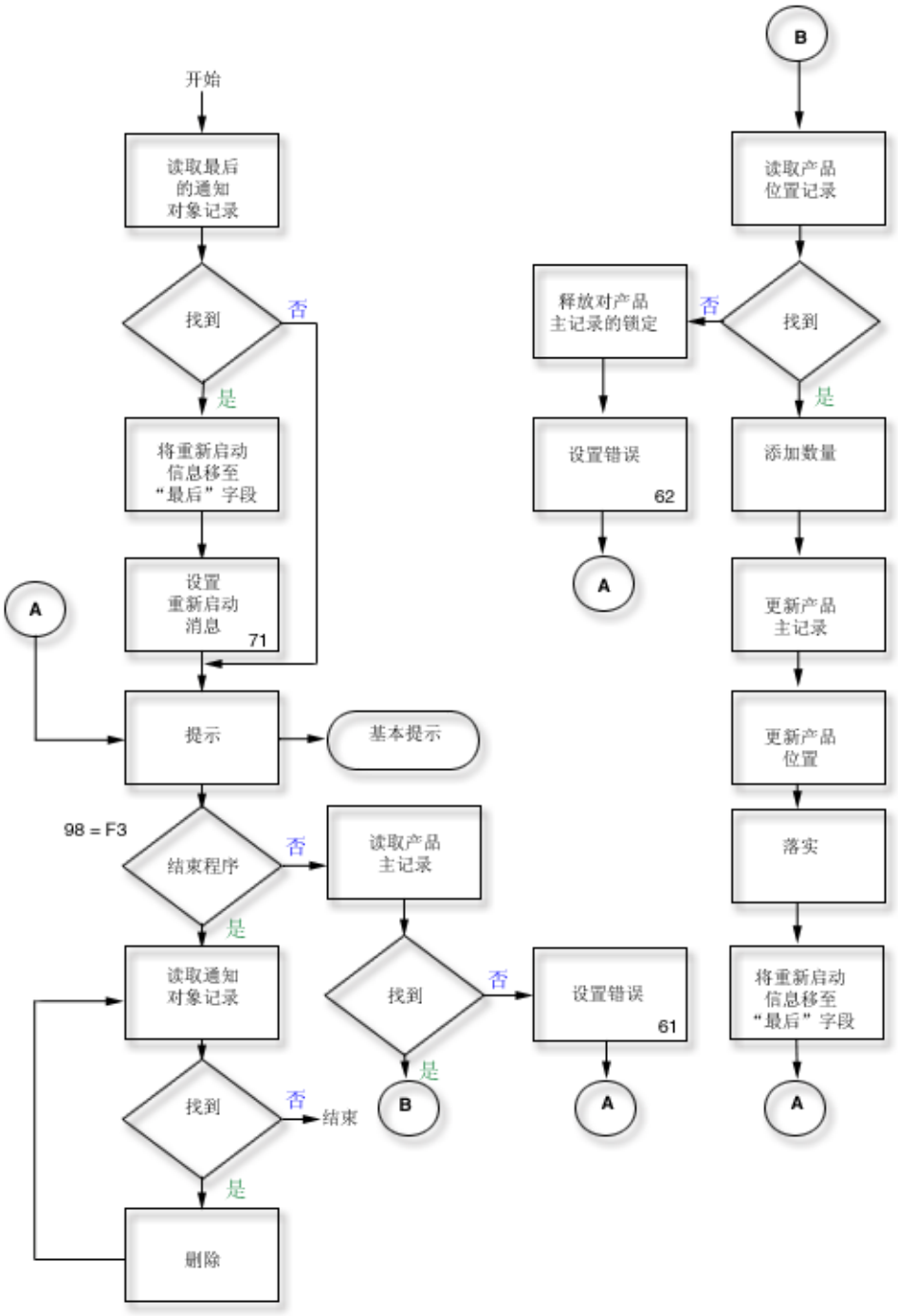
- 刚开始，程序随机处理通知对象并显示一条记录（如果该记录对于特定键存在）:
  - 如果存在多条记录，因为 PRDRCTP 文件处于 LIFO 序列，所以使用此键的最后一条记录。
  - 如果不存在记录，则不会中断事务，所以没有必要再次启动它。
  - 如果程序在第一次成功落实操作前失败，则该程序认为没有必要再次启动。
- 清除通知对象的例程在程序结束时发生:
  - 如果存在多个故障，则该例程可以处理通知对象中的多条记录的删除。
  - 尽管系统将落实标识放入数据库文件中，但是必须将落实标识指定为 RPG 程序中的变量。
  - 因为 RPG 允许外部描述数据结构，所以可以很方便地使用数据结构来指定落实标识。在此示例中，数据结构使用数据库文件用作通知对象的同一外部描述。

系统处理此程序时将提示用户输入产品号、位置和数量:

- 必须更新两个文件:
  - 产品主文件（PRDMSTP）
  - 产品位置文件（PRDLOCP）
- 每个文件中的记录必须存在才能更新。
- 在成功输入每个事务后，程序将输入字段移至相应的最后字段中。这些最后的字段在每次提示时显示给操作员，作为最后输入内容的反馈。
- 如果再次启动的信息存在，则将把它移至这些最后的字段并且将在屏幕上显示一条特殊消息。

在下面的图形中概括了此进程。将用户名传递给程序以提供通知对象中的唯一记录。

程序流



链接至描述

以下为此示例的 RPG 源代码。在程序的开始和结束时将通知对象（文件 PRDRCTP）用作正常文件，并且在调用程序前它还被指定为 CL（STRCMTCTL 命令）中的通知对象。

RPG 源

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FPRDMSTP UF  E           K        DISK         KCOMIT
  2.00      FPRDLOCP UF  E           K        DISK         KCOMIT
  3.00      FPRDRCTD CF  E                    WORKSTN
  4.00      F*
```

```
 5.00      F* The following file is the specific notify object for this pgm.
 6.00      F*      It is accessed only in a restart situation and at the
 7.00      F*        end of the program to delete any records.  The records
 8.00      F*        are written to the notify object by Commitment Control.
 9.00      F*
10.00      FPRDRCTP UF  E           K        DISK
11.00      ICMTID     E DSPRDRCTP

12.00      C           *ENTRY    PLIST
13.00      C                     PARM           USER10 10
14.00      C                     MOVE USER10    USER
15.00      C*
16.00      C*  Check for restart information - get last rcd per user
17.00      C*    PRDRCTP file access path is in LIFO sequence
18.00      C*
19.00      C           USER      CHAINPRDRCTR             20     Not found
20.00      C  N20                DO                               Restart
21.00      C                     EXSR MOVLST                      Move to last
22.00      C                     SETON                    71     Restart
23.00      C                     END
24.00      C*
25.00      C*  Basic processing loop
26.00      C*
27.00      C           L00P      TAG
28.00      C                     EXFMTPROMPT
29.00      C   98                GOTO END                         End of pgm
30.00      C           PRODCT    CHAINPRDMSTR             61     Not found
31.00      C   61                GOTO L00P
32.00      C           KEY       KLIST
33.00      C                     KFLD           PRODCT
34.00      C                     KFLD           LOCATN
35.00      C           KEY       CHAINPRDLOCR             62     Not found
36.00      C   62                DO
37.00      C                     EXCPTRLSMST                      Release lck
38.00      C                     GOTO L00P
39.00      C                     END
40.00      C                     ADD  QTY       ONHAND           Add
41.00      C                     ADD  QTY       LOCAMT
42.00      C                     UPDATPRDMSTR                     Update
43.00      C                     UPDATPRDLOCR                     Update
44.00      C*

45.00      C*  Commit and move to previous fields
46.00      C*
47.00      C           CMTID     COMIT
48.00      C                     EXSR MOVLST                      Move to last
49.00      C                     GOTO L00P
50.00      C*
51.00      C*  End of program processing
52.00      C*
53.00      C           END       TAG
54.00      C                     SETON                    LR
55.00      C*56.00      C*  Delete any records in the notify object
57.00      C*
58.00      C           DLTLP     TAG
59.00      C           USER      CHAINPRDRCTR             20     Not found
60.00      C  N20                DO
61.00      C                     DELETPRDRCTR                     Delete
62.00      C                     GOTO DLTLP
63.00      C                     END
64.00      C*
65.00      C*  Move to -Last Used- fields for operator feedback
66.00      C*
67.00      C           MOVLST    BEGSR
68.00      C                     MOVE PRODCT    LSTPRD
69.00      C                     MOVE LOCATN    LSTLOC
```

```
70.00      C                   MOVE QTY       LSTQTY
71.00      C                   MOVE DESCRP    LSTDSC
72.00      C                   ENDSR
73.00      OPRDMSTR E              RLSMST
```

示例: 所有程序的单个通知对象

对所有程序使用单个通知对象是有益的，因为再次启动所需要的所有信息都在同一对象中并且可以在所有程序中使用对通知对象采用的标准方法。在此情况下，使用唯一的用户和程序标识的组合以确保当再次启动程序时程序可访问正确的信息。

因为再次启动所需的信息可能因程序的不同而有所不同，所以不要对落实标识使用外部描述的数据结构。如果使用单个通知对象，则先前的程序可以在程序内（不是在外部）描述数据结构。例如:

```
 1   10      USER
11   20      PGMNAM
21   23      PRODCT
24   29      LOCATN
30   49      DESC
50   51  0   QTY
52  220      DUMMY
```

在使用此通知对象的每个程序内，为落实标识指定的信息对于程序将是唯一的（用户和程序名不是唯一的）。通知对象必须足够大才能包含任何程序将放入落实标识中的最多信息。

示例: 每个程序的唯一通知对象提供关于使用通知对象的更多示例。

示例: 使用标准处理程序来启动应用程序

标准处理程序是使用一个数据库文件作为所有应用程序的通知对象以再次启动应用程序的一种方法。此方法假定对于使用标准程序的所有应用程序的用户，用户概要文件名称是唯一的。

对于此方法，将物理文件 NFYOBJP 用作通知对象并且将其定义为:

唯一用户概要文件名称	10 个字符
程序标识	10 个字符
用于再次启动的信息	字符字段 (它必须足够大才能包含最大量 的信息以再次启动用于再次启动 所需信息的程序。 此字段是应用程序必需的。 在该示例中，假定它的长度为 200。)

以 SHARE(*YES) 创建了文件。该文件中的前两个字段是该文件的关键字。(此文件也可以定义为 RPG 程序中的数据结构)。

注意: 请阅读代码示例免责以获取重要的法律信息。

以下内容提供标准处理程序的示例代码:

- 示例: 标准处理程序的代码
- 示例: 标准落实处理程序
- 示例: 使用标准处理程序来决定是否重新启动应用程序

示例: 标准处理程序的代码

以下是使用标准处理程序的示例。在以下代码示例中显示的应用程序按如下方式执行:

1. 应用程序接收参数中的用户名并将其与程序名一起使用以作为通知对象中的唯一标识。
2. 应用程序将请求代码 R 传递给标准落实处理程序，该程序确定记录是否存在于通知对象中。

3. 如果标准落实处理程序返回代码 1，则将找到一条记录并且应用程序将向用户显示需要再次启动的信息。
4. 应用程序继续正常处理。
5. 当事务完成时，将保存值以供参考，以便工作站用户可以看到为先前的事务采取了哪些操作。

通知对象不提供保存的信息，原因是只有发生作业或系统故障时，才会更新通知对象。

有关此程序的流程，请参阅处理流程。

注意: 请阅读代码示例免责以获取重要的法律信息。

应用程序示例

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FPRDMSTP UF  E           K        DISK         KCOMIT
  2.00      FPRDLOCP UF  E           K        DISK         KCOMIT
  3.00      FPRDRCTD CF  E                    WORKSTN
  4.00      F*
  5.00      F* The following is a compile time array which contains the
  6.00      F*    restart information used in the next example
  7.00      F*
  8.00      E                    RTXT   50  50  1               Restart text
  9.00      I*
 10.00      I* Data structure used for info passed to notify object
 11.00      I*
 12.00      ICMTID       DS
 13.00      I                                       1  10 USER
 14.00      I                                      11  20 PGMNAM
 15.00      I                                      21  23 PRODCT
 16.00      I                                      24  29 LOCATN
 17.00      I                                      30  49 DESCRP
 18.00      I                                    P 50  510QTY
 19.00      I                                      52 170 DUMMY
 20.00      I                                     171 220 RSTART
 21.00      C           *ENTRY    PLIST
 22.00      C                     PARM          USER10 10
 23.00      C*
 24.00      C*  Initialize fields used to communicate with std program
 25.00      C*
 26.00      C                     MOVE USER10    USER
 27.00      C                     MOVEL'PRDRC2'  PGMNAM
 28.00      C                     MOVE 'R'       RQSCOD           Read Rqs
 29.00      C                CALL 'STDCMT'
 30.00      C                     PARM           RQSCOD  1
 31.00      C                     PARM           RTNCOD  1
 32.00      C                     PARM           CMTID 220        Data struct
 33.00      C            RTNCOD   IFEQ '1'                       Restart
 34.00      C                     EXSR MOVLST                     Move to last
 35.00      C SETON                                        71    Restart
 36.00      C                     END
 37.00      C*
 38.00      C*  Initialize fields used in notify object
 39.00      C*
 40.00      C                     MOVEARTXT,1    RSTART           Move text
 41.00      C*
 42.00      C* Basic processing loop
 43.00      C*
 44.00      C            LOOP     TAG
 45.00      C                     EXFMTPROMPT
 46.00      C   98                GOTO END
 47.00      C            PRODCT   CHAINPRDMSTR              61    Not found
 48.00      C   61                GOTO LOOP
```

```
 49.00      C            KEY      KLIST
 50.00      C                     KFLD           PRODCT
 51.00      C                     KFLD           LOCATN
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..
 52.00      C           KEY       CHAINPRDLOCR              62    Not found
 53.00      C   62                DO
 54.00      C                     EXCPTRLSMST                     Release lck
 55.00      C                     GOTO LOOP
 56.00      C                     END
 57.00      C                     ADD  QTY       ONHAND           Add
 58.00      C                     ADD  QTY       LOCAMT
 59.00      C                     UPDATPRDMSTR                    Update
 60.00      C                     UPDATPRDLOCR                    Update
 61.00      C*
 62.00      C*  Commit and move to previous fields
 63.00      C*
 64.00      C           CMTID     COMIT
 65.00      C                     EXSR MOVLST                     Move to last
 66.00      C                     GOTO LOOP
 67.00      C*  End of program processing
 68.00      C*
 69.00      C           END       TAG
 70.00      C                     MOVE 'D'       RQSCOD           Dlt Rqs
 71.00      C                     CALL 'STDCMT'
 72.00      C                     PARM           RQSCOD
 73.00      C                     PARM           RTNCOD
 74.00      C                     PARM           CMTID
 75.00      C                     SETON                     LR
 76.00      C*
 77.00      C*  Move to -Last Used- fields for operator feedback
 78.00      C*
 79.00      C           MOVLST    BEGSR
 80.00      C                     MOVE PRODCT    LSTPRD
 81.00      C                     MOVE LOCATN    LSTLOC
 82.00      C                     MOVE DESCRP    LSTDSC
 83.00      C                     MOVE QTY       LSTQTY
 84.00      C                     ENDSR
 85.00     OPRDMSTR E                 RLSMST
 86.00 ** RTXT       Restart Text
 87.00 Inventory Menu - Receipts Option
```

示例: 标准落实处理程序的代码

标准落实（STDCMT）处理程序执行与所有应用程序使用的单个通知对象通信所需要的功能。当落实控制功能自动向通知对象写入项时，用户编写的标准程序必须处理通知对象。标准程序对该方法进行简化和标准化。

将编写程序以验证传递了参数并且按如下所示执行适当的操作:

O=打开
调用程序请求将通知对象文件在返回时保持打开。因为 RPG 程序隐式打开通知对象，所以程序一定不能关闭它。设置了指示符 98 以便程序以 LR 关闭返回以保留程序的工作区并使通知对象保持打开，以便可以再次调用它而不会有额外的开销。

C=关闭
调用程序已确定它不再需要通知对象并请求关闭。将指示符 98 设置为关闭以允许通知对象的完全关闭。

R=读取
调用程序请求读取并传递回具有匹配关键字段的记录。程序使用传递的关键字段来尝试从 NFYOBJP 中检索记录。如果对于相同的键存在重复的记录，则将返回最后的记录。将相应地设置返回码，并且如果记录存在的话，将把它传递回数据结构 CMTID 中。

W=编写
调用程序请求将记录写入通知对象以允许下次调用该调用程序时，再次启动它。程序将已传递的数据的内容作为记录写入 NFYOBJP 中。

D=删除
调用程序请求删除此匹配键的记录。通常在成功完成调用程序时执行此功能以除去关于再次启动的任何信息。程序尝试删除已传递关键字段的任何记录。如果不存在记录，则将传递回不同的返回码。

S=搜索
调用程序请求搜索特定用户的记录，而不管是哪个程序编写该记录的。此功能在注册时的程序中使用以指示需要再次启动。程序只使用用户名作为键来查看记录是否存在。将适当设置返回码，并且将读取并传递回此键（如果它存在的话）的最后一条记录的内容。

注意: 请阅读代码示例免责以获取重要的法律信息。

下面显示了标准落实处理程序 STDCMT。

标准落实处理程序

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7 ..

  1.00      FNFYOBJP UF  E           K        DISK                      A
  2.00      ICMTID       DS
  3.00      I                                      1  10 UNQUSR
  4.00      I                                     11  20 UNQPGM
  5.00      I                                     21 220 BIGFLD
  6.00      C            *ENTRY    PLIST
  7.00      C                      PARM           RQSCOD  1
  8.00      C                      PARM           RTNCOD  1
  9.00      C                      PARM           CMTID 220
 10.00      C             UNQUSR   CABEQ*BLANKS   BADEND         H1 Invalid
 11.00      C             UNQPGM   CABEQ*BLANKS   BADEND         H2 Invalid
 12.00      C*
 13.00      C*  'O' for Open
 14.00      C*
 15.00      C             RQSCOD   IFEQ 'O'                        Open
 16.00      C                      SETON                     98    End LR
 17.00      C                      GOTO END
 18.00      C                      END
 19.00      C*
 20.00      C*  'C' for Close
 21.00      C*
 22.00      C             RQSCOD   IFEQ 'C'                        Close
 23.00      C                      SETOF                     98
 24.00      C                      GOTO END
 25.00      C                      END
 26.00      C*
 27.00      C*  'R' for Read - Get last record for the key
 28.00      C*
 29.00      C             RQSCOD   IFEQ 'R'                        Read
 30.00      C             KEY      KLIST
 31.00      C                      KFLD           UNQUSR
 32.00      C                      KFLD           UNQPGM
 33.00      C             KEY      CHAINNFYOBJR              51    Not found
 34.00      C   51                 MOVE '0'       RTNCOD
 35.00      C   51                 GOTO END
 36.00      C                      MOVE '1'       RTNCOD           Found
 37.00      C             LOOP1    TAG
 38.00      C             KEY      READENFYOBJR                 20 EOF
 39.00      C   20                 GOTO END
 40.00      C                      GOTO LOOP1
 41.00      C                      END
 42.00      C*
```

```
43.00      C*  'W' FOR Write
44.00      C*
45.00      C            RQSCOD   IFEQ 'W'                          Write
46.00      C                     WRITENFYOBJR
47.00      C                     GOTO END
48.00      C                     END
49.00      C*
50.00      C*  'D' for Delete - Delete all records for the key
51.00      C*
52.00      C           RQSCOD    IFEQ 'D'                          Delete
53.00      C           KEY       CHAINNFYOBJR              51    Not found
54.00      C   51              MOVE '0'       RTNCOD
55.00      C   51              GOTO END
56.00      C                   MOVE '1'       RTNCOD             Found
57.00      C           LOOP2     TAG
58.00      C                     DELETNFYOBJR
59.00      C           KEY       READENFYOBJR                  20 EOF
60.00      C  N20              GOTO LOOP2
61.00      C                   GOTO END
62.00      C                   END
63.00      C*
64.00      C*  'S' for Search for the last record for this user
65.00      C*              (Ignore the -Program- portion of the key)
66.00      C*
67.00      C           RQSCOD    IFEQ 'S'                             Search
68.00      C           UNQUSR    SETLLNFYOBJR                      20 If equal
69.00      C  N20              MOVE '0'       RTNCOD
70.00      C  N20              GOTO END
71.00      C                   MOVE '1'       RTNCOD               Found
72.00      C           LOOP3     TAG
73.00      C           UNQUSR    READENFYOBJR                      20 EOF
74.00      C  N20              GOTO LOOP3
75.00      C                   GOTO END
76.00      C                   END
77.00      C*
78.00      C*  Invalid request code processing
79.00      C*
80.00      C                   SETON                        H2    Bad RQS code
81.00      C                   GOTO BADEND
82.00      C*
83.00      C*  End of program processing
84.00      C*
85.00      C           END       TAG
86.00      C  N98              SETON                        LR
87.00      C                   RETRN
88.00      C* BADEND tag is used then fall thru to RPG cycle error return
89.00      C           BADEND    TAG
```

示例: 使用标准处理程序来决定是否重新启动应用程序

本主题提供了示例 CL 代码，用于使用标准处理程序来决定在异常 IPL 之后是否重新启动应用程序。

初始程序可以调用标准落实处理程序来确定是否有必要再次启动。然后，工作站用户可以决定是否再次启动。

初始程序将请求代码 S（搜索）传递给标准程序（搜索用户的任何记录）。如果记录存在，则再次启动的信息将传递给初始程序并向工作站用户显示该信息。

通知对象中的落实标识包含初始程序可显示的信息，这些信息标识哪个程序需要再次启动。例如，可以保留落实标识的最后 50 个字符以包含此信息。在应用程序中，此信息可位于编译时数组中并可移至启动步骤中的数据结构。示例: 标准落实处理程序的代码显示如何将它包括在应用程序中。

以下是初始程序的示例，它确定记录是否存在于通知对象中。

注意: 请阅读代码示例免责以获取重要的法律信息。

初始程序示例

```
SEQNBR *... ... 1 ... ... 2 ... ... 3 ... ... 4 ... ... 5 ... ... 6 ... ... 7

  1.00            PGM
  2.00            DCLF       CMTINLD
  3.00            DCL        &RQSCOD *CHAR LEN(1) VALUE(S)  /* Search */
  4.00            DCL        &RTNCOD *CHAR LEN(1)
  5.00            DCL        &CMTID *CHAR LEN(220)
  6.00            DCL        &USER *CHAR LEN(10)
  7.00            DCL        &INFO *CHAR LEN(50)
  8.00            RTVJOBA    USER(&USER)
  9.00            CHGVAR     &CMTID (&USER *CAT XX)
 10.00                       /* The XX is required to prevent a blank Pgm nam */
 11.00            CALL       STDCMT PARM(&RQSCOD &RTNCOD &CMTID)
 12.00            IF         (&RTNCOD *EQ '1') DO /* RESTART REQD */
 13.00            CHGVAR     &INFO %SST(&CMTID 171 50)
 14.00            SNDRCVF    RCDFMT(RESTART)
 15.00            ENDDO
 16.00                       /*                                          */
 17.00                       /* Enter normal initial program statements  */
 18.00                       /*    or -TFRCTL- to first menu program      */
 19.00                       /*                                           */
 20.00            ENDPGM
```

对事务和落实控制进行故障诊断

下列数页提供一些任务，您对落实控制进行故障诊断时需要执行这些任务:

落实控制错误
此信息描述创建错误、列示落实控制错误以及提供处理错误的方法的条件。

检测死锁
此任务的目标是查找死锁状态。

在发生通信故障之后恢复事务
此任务的目标是处理与远程系统进行通信失败后在该系统上执行工作的事务。

何时强制落实和回滚以及何时取消再同步和回滚
此信息描述何时以及如何强制回滚或落实，以及何时取消再同步

结束长时间运行的回滚
此信息描述如何结束消耗大量处理器时间、锁定资源或占用存储器空间的长时间运行的回滚。

落实控制错误

当使用落实控制时，了解哪些状态会导致错误，哪些状态不会导致错误是非常重要的。通常，当落实控制功能使用不一致时就会发生错误，例如当使用落实定义的文件仍然打开时运行“结束落实控制”（ENDCMTCTL）命令。

落实处理期间出错

如果在落实操作期间发生通信故障或系统故障，则可能需要执行再同步以确保事务管理器使所有系统上涉及该事务的数据保持一致。再同步的行为以及它影响落实操作的方式取决于下列因素:

- 等待结果落实选项。有关详细信息，请参阅两阶段落实的落实定义: 不等待结果。
- 事务的状态。有关详细信息，请参阅两阶段落实控制的事务状态。

如果故障是灾难性的，以至于不能修复它，或者不能及时修复它，则涉及该事务的其它系统的系统操作员必须作出启发式决策。启发式决策落实或回滚在该事务期间在该系统上所作的更改。若在作出此类决定之后修复了故障，而再同步检测到该决定产生数据完整性问题，则将消息 CPD83D9 或 CPD83E9 发送至 QSYSOPR 消息队列。

下列信息提供了有关使用落实控制错误的更多详细信息:

- 错误状态
- 无错误状态
- 落实控制期间要监视的错误消息
- 在执行 CALL 命令之后要监视的错误消息
- 正常落实或回滚处理的故障

错误状态

如果发生错误，会发送您在程序中可以监视的脱离消息。以下是与落实控制有关的一些典型错误:

- 不插入 ENDCMTCTL 命令运行连续的 STRCMTCTL 命令。
- 在落实控制下打开了文件，但没有运行 STRCMTCTL 命令。

  这不是运行于激活组（要使用作业级别落实定义）中的程序的错误状态。作业级别落实定义只能由单个程序启动，但当由程序启动时，作业级别落实定义可由运行于任何激活组（未正在使用激活组级别落实定义）中的任何程序使用。运行于激活组（要使用激活组级别落实定义）中的程序必须首先使用 STRCMTCTL 命令启动激活组级别落实定义。
- 未记录在落实控制下对输出打开的文件。
- 共享文件的第一个打开操作将文件置于落实控制之下，但同一共享文件的后续打开操作不会这样做。
- 共享文件的第一个打开操作没有将文件置于落实控制之下，但同一共享文件的后续打开操作却将文件置于落实控制之下。
- 在单个事务中达到了作业的记录锁定限制。
- 程序发出读操作、落实操作并且更改同一条记录。由于落实操作已释放对该记录的锁定，所以在落实操作之后必须再次发出读操作。
- 对于一阶段位置，置于落实控制之下的资源不驻留在已处于落实定义的落实控制之下的资源所在的同一位置。
- 当发出 ENDCMTCTL 命令时存在未落实的更改。

  如果所有文件都已关闭、任何远程数据库已断开连接并且没有 API 落实资源仍与要结束的落实定义相关联，则这不是 ENDCMTCTL 命令的错误状态。
- 运行了落实、回滚或 ENDCMTCTL 命令，而未运行 STRCMTCTL 命令。

  这不是运行于激活组中的程序的错误状态，并且作业级别落实定义是活动的。作业级别落实定义只能由单个程序启动，但当由程序启动时，作业级别落实定义可由运行于任何激活组（未正在使用激活组级别落实定义）中的任何程序使用。运行于激活组中并且要使用激活组级别落实定义的程序必须首先使用 STRCMTCTL 命令启动激活组级别落实定义。
- 在落实定义的落实控制下仍然打开文件的情况下运行 ENDCMTCTL 命令。
- 执行保存操作的作业具有一个或多个不在落实边界的落实定义。
- 因为具有可落实资源的其它作业没有在为 SAVACTWAIT 参数指定的时间内到达落实边界，所以活动时保存操作结束。
- 由于正在将 API 可落实资源添加至单个作业的多个落实定义，所以活动时保存过程不能继续。

- 对于单个作业，存在超过 1023 个落实定义。
- 由于资源故障，丢失与远程位置的对话。这可能会导致回滚事务。
- 为进行更新而打开的一阶段资源存在于未启动落实操作的节点处。必须除去启动落实请求的资源或节点。
- 当事务处于必需回滚（RBR）状态时请求落实操作。必须执行回滚操作。
- API 出口程序发出落实请求或回滚请求。
- 触发器程序对在其下已调用触发器程序的落实定义发出落实请求或回滚请求。

触发器程序可启动独立的落实定义并对该定义发出落实或回滚请求。

无错误状态

您可能以为这些情况会导致错误消息。但是，落实控制允许这些情况发生。以下是不会在其中发生错误的落实控制的某些情形:

- 落实或回滚操作已运行并且没有资源处于落实控制之下。这允许您将落实或回滚操作包括在程序中而不必考虑是否有资源处于落实控制之下。这还允许您在进行任何可落实更改前指定落实标识。
- 落实或回滚操作已运行并且没有未落实的资源更改。这允许您将落实或回滚操作包括在程序中而不必考虑是否有未落实的资源更改。
- 将关闭落实控制之下的文件并且存在未落实的记录。此情况允许调用另一程序来执行落实或回滚操作。将发生此情况，而不管是否共享文件。此功能允许子程序执行数据库更改（是涉及多个程序的事务的一部分）。
- 作业正常或异常结束，并带有一个或多个落实定义的未落实更改。将回滚所有落实定义的更改。
- 激活组结束，并带有激活组级别落实定义的暂挂更改。如果激活组正在正常结束并且当关闭落实控制（作用域是正在结束的同一激活组）下打开的任何文件时没有错误发生，则系统将执行隐式落实。否则，将执行隐式回滚。
- 程序将再次访问尚未落实的已更改记录。这允许程序执行下列操作:
  - 添加记录并在指定落实操作前更新它。
  - 在指定落实操作前两次更新同一记录。
  - 添加记录并在指定落实操作前删除它。
  - 通过不同的逻辑文件（在落实控制之下）再次访问未落实的记录。
- 在 STRCMTCTL 命令上指定 LCKLVL（*CHG 或 *CS）并使用落实操作以只读方式打开文件。在此情况下，不会对请求进行锁定。这就好像落实控制没有生效，但是该文件的确出现在落实控制之下的文件的 WRKJOB 菜单选项上。
- 发出 STRCMTCTL 命令并且不在落实控制之下打开任何文件。在此情况下，对文件执行的任何记录级别的更改不会在落实控制下执行。

落实控制期间要监视的错误消息

几个不同的错误消息可由落实或回滚操作返回或发送至作业记录，这取决于消息的类型以及发生错误的时间。

这些消息会在以下过程中发生:

- 正常落实或回滚处理
- 在作业处理结束期间的落实或回滚处理
- 在激活组结束期间的落实或回滚处理

在激活组结束或作业处理结束期间，不能监视下列任何消息。另外，只能监视 CPFxxxx 消息。CPDxxxx 消息始终作为不能监视的诊断消息发送。当激活组结束期间结束激活组级别的落实定义时或作业结束期间结束任何落实定义时遇到的任何错误将作为诊断消息保留在作业记录中。

要查找的与落实控制有关的错误消息如下所示:

CPD8351
可能未落实更改。

CPD8352
未在远程位置 &3 处落实更改。

CPD8353
可能未落实对关系数据库 &1 的更改。

CPD8354
可能未落实对 DDM 文件 &1 的更改。

CPD8355
可能未落实对 DDL 对象 &1 的更改。

CPD8356
可能落实了回滚更改。

CPD8358
可能未回滚对关系数据库 &1 的更改。

CPD8359
可能未回滚对 DDM 文件 &1 的更改。

CPD835A
可能未回滚对 DDL 对象 &3 的更改。

CPD835C
未更新 &2 中的通知对象 &1。

CPD835D
DRDA(R) 资源不允许保持 SQL 游标。

CPF835F
落实或回滚操作失败。

CPD8360
已经释放了成员和 / 或文件。

CPD8361
落实期间 API 出口程序 &1 失败。

CPD8362
回滚期间 API 出口程序 &1 失败。

CPD8363
落实期间 API 出口程序 &1 在 &4 分钟之后结束。

CPD8364
回滚期间 API 出口程序 &1 在 &4 分钟之后结束。

CPD836F
落实控制操作期间发生了协议错误。

CPD83D1
API 资源 &4 不能是上一代理程序。

CPD83D2
资源与落实控制不兼容。

CPD83D7
回滚已更改的落实操作。

CPD83D9
发生了启发式混合状态。

CPF83DB
落实操作导致回滚。

CPD83DC
“发生问题时执行的操作”用于确定落实或回滚操作；原因 &2。

CPD83DD
对话已结束；原因 &1。

CPD83DE
返回信息无效。

CPD83EC
API 出口程序 &1 已表决回滚。

CPD83EF
对下一个逻辑工作单元启动了回滚操作。

CPF8350
找不到落实定义。

CPF8355
不允许 ENDCMTCTL。暂挂更改是活动的。

CPF8356
落实控制结束，未落实 &1 本地更改。

CPF8358
未更新 &2 中的通知对象 &1。

CPF8359
回滚操作失败。

CPF835A
取消了结束落实定义 &1。

CPF835B
当结束落实控制时发生错误。

CPF835C
落实控制结束，未落实远程更改。

CPF8363
落实操作失败。

CPF8364
落实控制参数值无效。原因码 &3。

CPF8367
不能执行落实控制操作。

CPF8369
不能将 API 落实资源置于落实控制之下；原因码 &1。

CPF83D0
不允许落实操作。

CPF83D2
落实完成 == 正在进行的再同步已返回。

CPF83D3
落实完成 == “启发式混合”已返回。

CPF83D4
未发送逻辑工作单元日志项。

CPF83E1
由于约束违例，落实操作失败。

CPF83E2
需要回滚操作。

CPF83E3
请求的嵌套级别不活动。

CPF83E4
落实控制结束，未落实资源。

CPF83E6
落实控制操作已完成，正在进行再同步。

CPF83E7
不允许 X/Open 全局事务的落实或回滚。

在执行 CALL 命令之后监视错误

当调用使用落实控制的程序时，监视意外错误并且执行回滚操作（如果错误发生的话）。例如，当程序遇到意外错误（如 RPG 除以零错误）时会存在未落实的记录。根据作业的查询消息应答（INQMSGRPY）参数的状态，程序发送查询消息或执行缺省操作。如果操作员响应或缺省操作结束该程序，则未落实的记录仍然存在并且等待落实或回滚操作。

如果调用另一个程序并且导致落实操作，则落实前一程序中部分完成的事务。

要防止落实部分完成的事务，在执行 CALL 命令之后监视脱离消息。例如，如果它是 RPG 程序，则使用以下编码:

```
CALL RPGA
MONMSG MSGID(RPG9001)
EXEC(ROLLBACK) /*Rollback if pgm is canceled*/
```

如果它是 COBOL 程序:

```
CALL COBOLA
MONMSG MSGID(CBE9001)
EXEC(ROLLBACK) /*Rollback if pgm is canceled*/
```

正常落实或回滚处理的故障

在落实或回滚处理期间错误可能在任何时间发生。下表将该处理分为四种情况。中间的列描述当系统在每个情况中遇到错误时所采取的操作。第三列建议您或您的应用程序必须执行哪些操作以对这些消息做出响应。这些建议与系统处理落实控制处理的方法一致。

情况	落实或回滚处理	建议的操作
记录级别 I/O **落实**失败	• 如果在准备阶段期间发生错误，则将回滚事务并且将发送消息 CPF83DB。 • 如果在落实阶段期间发生错误，则落实处理将继续尽可能多地落实剩余的资源。在落实处理结束时发送消息 CPF8363。	监视消息；按期望的方式处理
在落实期间 API 落实资源的对象级别或落实和回滚出口程序失败	• 如果在准备阶段期间发生错误，则将回滚事务并且将发送消息 CPF83DB。 • 如果在落实阶段期间发生错误，则处理将继续尽可能多地落实或回滚剩余资源。根据落实资源类型，将返回下列其中一条消息: – CPD8353 – CPD8354 – CPD8355 – CPD8361 在落实处理结束时发送消息 CPF8363。	监视消息；按期望的方式处理
记录级别 I/O **回滚**失败	1. 返回 CPD8356 2. 尝试继续处理以回滚对象级别或 API 落实资源 3. 在处理结束时返回 CPF8359	监视消息；按期望的方式处理
回滚期间 API 落实资源的对象级别或落实和回滚出口程序失败	1. 根据落实资源类型，将返回下列其中一条消息: • CPD8358 • CPD8359 • CPD835A • CPD8362 2. 继续处理 3. 在处理结束时返回 CPF8359	监视消息；按期望的方式处理

在作业结束期间落实或回滚处理

除了发送下列其中一条消息外，当作业正在结束时，先前表中描述的所有情况也适用:

- 如果只注册了本地资源，则将发送 CPF8356 消息
- 如果只注册了远程资源，则将发送 CPF835C 消息
- 如果同时注册了本地资源和远程资源，则将发送 CPF83E4 消息

此外，如果调用了 API 可落实资源的落实和回滚出口程序，则可能会显示特定于作业完成的两个消息中的一个。如果落实和回滚出口程序未在 5 分钟之内完成，则将取消程序并且将发送诊断消息 CPD8363（对于落实）或 CPD8364（对于回滚），并且将继续完成剩余的落实或回滚处理。

IPL 期间的落实或回滚处理

除了发送消息 CPF835F（不是消息 CPF8359 或 CPF8363），在落实定义的 IPL 恢复期间，先前表中描述的所有情况也适用。为特定落实定义发送的消息可能出现在某个 QDBSRVxx 作业的作业记录中或 QHST 记录中。在 QHST 记录中，消息 CPI8356 指示特定落实定义的 IPL 恢复的开始。消息 CPC8351 指示特定落实定义的 IPL 恢复的结束，并且在这两个消息之间找到了关于恢复该落实定义的任何其它消息。

如果调用了 API 可落实资源的落实和回滚出口程序，则可能会显示特定于落实定义的两个消息中的一个。如果落实和回滚出口程序未在 5 分钟之内完成，则将取消程序并且将发送诊断消息 CPD8363（对于落实）或 CPD8364（对于回滚），并且将继续完成剩余的落实或回滚处理。

检测死锁

当作业保持某对象（对象 A）上的锁定并且正在等待获取另一对象（对象 B）的锁定时，而这时另一作业或事务当前保持对象 B 上的锁定并且正在等待获取对象 A 的锁定时，就会发生死锁情况。

执行下列步骤以查找是否发生了死锁情况，并且如果发生的话，请修正它:

1. 在活动作业的列表中找到暂挂的作业。确定作业的状态以确定是否暂挂了作业。
2. 查看作业正在等待锁定的对象。对于具有事务作用域锁定的事务，请参阅显示事务的已锁定对象以了解各步骤。对于具有作业作用域锁定的事务，请参阅详细信息: 活动作业属性。
3. 对于作业正在等待锁定的所有对象，查看锁持有者（事务或作业）的列表并尝试查找对应于暂挂的作业所请求级别的冲突锁定。
4. 如果事务正保持冲突锁定，则显示与此事务相关联的作业并查看是否有某个作业正在等待锁定。
5. 确定此等待作业是否正在尝试锁定由初始暂挂作业锁定的其中一个对象。当查找要正在尝试锁定由初始暂挂作业锁定的某个对象的作业时，可以将有问题的对象标识为故障点。
6. 调查事务以确定操作的正确过程。
   a. 查看事务属性以了解哪个应用程序启动它，然后查看该应用程序代码。
   b. 或者通过查找事务属性中的“落实周期标识”然后在日志中搜索包含此标识的项来跟踪事务的操作，直到此点。为此，可以使用检索日志项（RTVJRNE）命令并指定 CMTCYCID 参数。
   c. 在获取了相关信息之后，用户可选择强制回滚或落实操作。

在发生通信故障之后恢复事务

在出现通信故障时，系统通常自动完成与任何远程系统的再同步。但是，如果故障是灾难性的以至于永远不能重新建立对远程系统的通信（例如，如果通信线路断开），则必须取消再同步并且自己恢复事务。事务还可能正持有需要释放的锁定。

1. 在 iSeries(TM) 导航器中，显示您正在使用的事务的落实控制信息。
2. 查找您感兴趣的正在尝试与远程系统再同步的事务。将该事务的**正在进行的再同步**字段设置为**是**。
3. 通过检查个别事务的资源状态来查找连接至远程系统的事务。
4. 在标识事务后，根据事务的状态，您可能必须强制落实或强制回滚。

5. 在调查事务属性后，决定是落实还是回滚。
   - 可以使用**工作单元标识**来查找其它系统上的事务的其它部分。
   - 还可以从事务的状态确定是落实还是回滚。例如，如果数据库事务正在通信故障期间执行两阶段落实并且故障后其状态为“已准备”或“上一代理暂挂”，则可以选择在事务上强制落实。
6. 在不确定的事务上强制落实或回滚后，停止在失败的连接上对标识的事务进行再同步。

有关恢复的更多信息，请参阅何时强制落实和回滚以及何时取消再同步。

何时强制执行落实和回滚以及何时取消再同步

强制落实、回滚或取消再同步的决定称为**启发式决策**。启发式决策是您强制系统落实或回滚事务时执行的操作。当作出启发式决策时，如果您的决定与事务中的其它位置的结果不一致，则事务的状态就变成启发式混合。当确定由参与事务的所有其它位置执行的操作以及再同步数据库记录时，您必须小心。

在作出启发式决策之前，尽可能多地收集有关事务方面的信息。显示与落实定义相关联的作业并记录所涉及的日志和文件。以后，如果需要显示日志项以及手工应用或除去已记录的更改，则可以使用此信息。

查找有关事务的信息的最好位置是查看该事务的启动程序所在的位置。然而，落实或回滚的决定权可能由 API 资源或上一代理程序拥有。

如果将 API 资源注册为上一代理程序资源，则是落实还是回滚的最终决定权由 API 资源拥有。您需要查看有关应用程序以及应用程序如何使用 API 资源来确定是落实还是回滚的信息。

如果事务选择了上一代理程序，则上一代理程序拥有落实或回滚的决定权。请查看上一代理程序的状态以获取有关事务的信息。

当由于不能修复的系统或通信故障而必须作出启发式决策或取消再同步时，可以使用以下步骤查找所有不确定的事务:

1. 在 iSeries(TM) 导航器中，展开想要使用的系统。
2. 展开**数据库**和系统的本地数据库。
3. 展开**事务**。
4. 展开**数据库事务**或**全局事务**。

在此屏幕中，您可以看到每个事务的落实定义、再同步状态、当前工作单元标识和当前工作单元状态。查找具有以下状态的事务:

- 具有**逻辑工作单元状态**为“已准备”或“上一代理暂挂”的事务。
- 显示**正在进行的再同步**状态为“是”的事务。

要使用此系统上正在参与事务的作业，右键单击该事务并选择**作业**。

当右键单击事务时，还可以选择**强制落实**、**强制回滚**或**取消再同步**。

在作出启发式决策或取消再同步之前，您可能想检查其它系统上与该事务相关联的作业的状态。检查远程系统上的作业可能会帮助您避免导致系统之间数据库不一致的决定。

1. 右键单击想要使用的事务。
2. 选择**资源状态**。
3. 在“资源状态”对话框中，对 SNA 连接选择**对话**选项卡；对 TCP/IP 连接选择**连接**。

每个对话资源表示正在参与事务的一个远程系统。在远程系统上，可以使用“iSeries 导航器”来查看与该事务相关联的事务。

工作单元标识的基本部分在所有系统上都是相同的。当在远程系统上显示落实控制信息时，查找与本地系统上相同的工作单元标识的基本部分。

例如，如果本地系统上的工作单元标识以 APPN.RCHASL97.X'112233445566 开头，则在远程系统上查找也以 APPN.RCHASL97.X'112233445566 开头的工作单元标识。

结束长时间运行的回滚

回滚操作除去自前一落实操作或回滚操作以来在事务中所作的所有更改。在回滚操作期间，系统还释放与事务有关的锁定。如果系统包含成千上万个事务，则系统可能要花几个小时才能完成回滚操作。这些长时间运行的回滚会消耗相当多的处理器时间、锁定资源或占用存储器空间。

在结束长时间运行的回滚之前，您需要知道正在回滚哪些落实定义以及各落实定义正处于哪种状态。正在回滚的落实定义的“状态”字段设置为 ROLLBACK IN PROGRESS。

使用“使用落实定义”（WRKCMTDFN）命令来遵循下列步骤检查回滚的状态:

1. 从基于字符的界面输入 WRKCMTDFN JOB(*ALL)。
2. 按 F11 键以显示“状态”字段。

如果您结束长时间运行的回滚，在事务期间更改的文件将保留部分事务。如果文件不能具有部分事务，则一定不能结束回滚。要查看在事务期间更改了哪些文件，从 WRKCMTDFN 列表中选择选项 5 以显示状态。按 F6 键以显示资源状态并选择“记录级别”。

必须具有“所有对象”（*ALLOBJ）特权才能结束长时间运行的回滚。要结束长时间运行的回滚，遵循下列步骤:

1. 从基于字符的界面输入 WRKCMTDFN JOB(*ALL)。
2. 对想要结束的落实定义输入选项 20（结束回滚）。

具有部分事务的文件在“显示文件描述”（DSPFD）命令的输出中将“若部分事务存在，则回滚结束”字段设置为 *YES。必须除去部分事务才能使用文件。可以通过删除文件并从先前保存版本恢复文件来除去部分事务。如果没有先前保存版本，则可以使用“更改记录的对象”（CHGJRNOBJ）命令来复位“部分事务存在”状态以便您可以打开文件。使用 CHGJRNOBJ 要求您编辑文件以使文件处于一致状态。仅当没有先前保存版本可用时才能使用 CHGJRNOBJ 命令。

禁用结束长时间运行的回滚功能

具有 *ALLOBJ 特权的用户缺省情况下可以结束回滚。如果想要限制具有 *ALLOBJ 特权的用户结束回滚，您可以通过创建数据区 QGPL/QTNNOENDRB 达到此目的。

有关数据区的更多信息，请参阅创建数据区（CRTDTAARA）命令。

落实控制的相关信息

下面是与“落实控制”主题相关的 iSeries(TM) 手册和 IBM(R) Redbooks(TM)（为 PDF 格式）、Web 站点和“信息中心”主题。可以查看或打印任何 PDF。

手册

- 请参阅 V5R1 补充手册 Web 站点上的 COBOL/400(R) User's Guide （5.8 MB）。
- 请参阅 V5R1 补充手册 Web 站点上的 RPG/400(R) User's Guide （2 MB）。

红皮书

- Connecting WebSphere(R) to DB2(R) UDB Server （5 MB）
- Advanced Functions and Administration on DB2 Universal Database(TM) for iSeries （5.4 MB）
- Stored Procedures and Triggers on DB2 Universal Database for iSeries （4.8 MB）
- Striving for Optimal Journal Performance on DB2 Universal Database for iSeries （3.1 MB）

Web 站点

The Open Group（www.opengroup.org）

其它信息

- 数据库编程
- SQL 编程概念
- XA API
- 日志管理

要将 PDF 保存在工作站上以查看或打印:

1. 右键单击浏览器中的 PDF（右键单击上面的链接）。
2. 单击**目标另存为...**
3. 浏览至想要在其中保存 PDF 的目录。
4. 单击**保存**。

如果需要 Adobe Acrobat Reader 来查看或打印这些 PDF，则可从 Adobe Web 站点（www.adobe.com/products/acrobat/readstep.html） 下载副本。

附录. 声明

本信息是为在美国提供的产品和服务编写的。

IBM(R) 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

```
IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY  10594-1785
U.S.A.
```

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan
```

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本出版物的新版本中。IBM 可以随时对本出版物中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

```
IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.
```

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

所有 IBM 的价格均是 IBM 当前的建议零售价，可随时更改而不另行通知。经销商的价格可与此不同。

本信息仅用于规划目的。在所述产品上市之前，可以更改此处的信息。

本信息包含日常业务经营中使用的数据和报告的示例。为了尽可能完整地说明这些示例，这些示例中包括个人、公司、品牌和产品的名称。所有这些人名或名称均系虚构，如有实际的企业名称和地址与此雷同，纯属巧合。

版权许可:

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。用户如果是为了按照 IBM 应用程序编程接口开发、使用、经销或分发应用程序，则可以任何形式复制、修改和分发这些样本程序，而无须向 IBM 付费。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

(C)（贵公司的名称）（年）。此部分代码是根据 IBM 公司的样本程序衍生出来的。(C) Copyright IBM Corp. （输入年份）。All rights reserved.

如果您正以软拷贝格式查看本信息，图片和彩色图例可能无法显示。

编程接口信息

本“落实控制”主题记录了一些编程接口，它们设计用于允许客户编写获取 i5/OS 的服务的程序。

商标

下列各项是 International Business Machines Corporation 在美国和／或其他国家或地区的商标:
Application System/400
AS/400
e (logo)
IBM
iSeries
Operating System/400
OS/400
400
Websphere
RPG/400
Redbooks

Integrated Language Environment
DRDA
Distributed Relational Database Architecture
DB2 Universal Database
DB2
COBOL/400

其他公司、产品和服务名称可能是其他公司的商标或服务标记。

用于下载和打印出版物的条款和条件

如果符合以下条款和条件并且由此您表示接受它们，则授予您使用您选择下载的出版物的准用权。

个人使用：只要保留所有的专有权声明，您就可以为个人、非商业使用复制这些出版物。未经 IBM 明确同意，您不可以分发、展示或制作这些出版物或其中任何部分的演绎作品。

商业使用：只要保留所有的专有权声明，您就可以仅在企业内复制、分发和展示这些出版物。未经 IBM 明确同意，您不可以制作此信息的演绎作品，或者在您的企业外部复制、分发或展示这些出版物或其中的任何部分。

除非本准用权中有明确授权，不得把其他准用权、许可或权利（无论是明示的还是暗含的）授予其中包含的这些出版物或任何信息、数据、软件或其他知识产权。

当使用这些出版物损害了 IBM 的利益，或者根据 IBM 的规定，未正确遵守上述指导说明时，则 IBM 保留自主决定撤销本文授予的准用权的权利。

您不可以下载、出口或再出口本信息，除非完全遵守所有适用的法律和法规，包括所有美国出口法律和法规。IBM 对这些出版物的内容不作任何保证。本信息“按现状”提供，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的关于适销和适用于某种特定用途的保证。

从此站点下载或打印出版物信息，即表明您同意这些条款和条件。

代码示例免责

IBM 授予您使用所有编程代码示例的非专有版权许可，您可以由此生成根据您的特定需要而定制的相似功能。

IBM 提供所有样本代码只是出于解释的目的。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

此处包含的所有程序“按现状”提供，不附有任何形式的保证。特此明确声明免除任何暗含的非侵权和关于适销和适用于某种特定用途的保证的责任。

中国印刷