企业内部业务流程管理

何谓业务流程管理？

整合所有事务：EJBs, CICS
事务处理，人员交互

处理异常业务

Geert Van de Putte
Tony Benedetti
Daniel Gagic
Peter Gersak
Krisztian Krutzler
Mark Perry

ibm.com／redbooks

国际技术支持组织

企业内部业务流程管理

2001 年 10 月

注意！使用此信息和其支持的产品之前，请务必先阅读第 411 页的“ 特别声明 ”中的一般信息。

第一版（2001年10月）

此版本适用于MQSeries工作流版本3和发行版3，程序号5697-FM3用于Window NT版本4和Windows 2000。

请将您的意见寄往：
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 对于您所提供的任何信息，有权利以任何它认为适当的方式使用或散发，而不必对您负任何责任。

前言

企业中的业务流程管理是 MQSeries Workflow 的核心。本红皮书阐述了实现业务流程管理模型的步骤。该解决方案整合了许多商业应用并为人与业务流程实例交互提供了基于浏览器的接口。

连接在一起的应用程序是使用多种技术实现的。一方面，解决方案集成了运行于 S/390 后台系统的 CICS 事务处理。另一方面，它还集成了运行在 Websphere 应用服务器上的 Enterprise JavaBeans。使用 MQSeries integrator 和 MQSseries Adapter offering 实现应用集成。

本红皮书中，业务流程模型由许多步骤构建并扩展。在每个实现阶段，读者都会关注当前实现的不足。业务流程逐步改进以处理更多异常情况，从而确保既符合服务水平协议又不影响边际利润。

最后，本红皮书讨论了怎样利用业务流程管理系统的审计信息进一步改进模型的实现。

此红皮书的编写小组

此红皮书是由国际技术支持组织 Raleigh 中心的来自世界各地的专家小组编写的。

Geert Van de Putte 是 Raleigh 中心国际技术支持组织的 IT 专家。他拥有五年的设计和实施基于 MQSeries 解决方案的经验。其加入 ITSO 之前，Geert 在比利时的 IBM 全球服务部工作。

Tony Benedetti 是加州康科德城美利坚银行的高级系统工程师。他拥有 20 年的 IT 从业经验，并且最近七年一直致力于设计、开发和实施成像，以及工作流解决方案。最近，他广泛地使用 MQSeries Workflow 和 WebSphere 进行工作。

Daniel Gagic 是澳大利亚墨尔本 IBM 软件部的 IT 解决方案设计师。他是应用和集成中间件专家，从事 IT 业（包括在 IBM 的两年时间）长达超过七年之久。其专业领域包括：MQSeries family 产品、WebSphere 应用服务器以及 B2B 整合。

Peter Gersak 是 IBM 斯洛文尼亚的 IT 专家。他已在 IBM 工作三年，并且最近两年他一直在欧洲的东南部的技术销售部从事 MQSeries family 中间件产品的相关工作。其专长领域包括：MQSeries、MQSeries Everyplace 和 MQSeries Integrator。他还是《MQSeries Integrator 的 IGS 安装服务操作指南》2.0 版本的合著者。

Krisztian Krutzler 是匈牙利的系统设计师和中间件专家。他拥有五年的 IT 从业经验，并且其从事企业应用集成达两年的时间。其特长领域包括：系统分析和设计、数据库设计、用 MQSeries family 产品进行应用集成解决方案的设计和开发、WebSphere 以及 DB2。

Mark Perry 是 IT 专家。他拥有十年在 Hursley MQ 共同体工作的经验，主要是在系统测试中担任小组领导，与此同时还在 Raleigh 的 MQ 服务部、NC 和 Poughkeepsie 工作过一段时间。

特别声明

此出版物旨在帮助 IT 设计师设计和创建业务流程管理解决方案。本出版物中的信息不作为由 MQSeries Workflow 和 MQSeries Integrator 提供的任何编程接口规范。关于哪些出版物被视为产品文档的详细信息，请参阅《用于 MQSeries Workflow 和 MQSeries Integrator 的 IBM 编程公告》的“出版物”部分。

IBM商标

以下术语是国际商业机器在美国和/或其它国家的商标：

e (logo)®	Redbooks
IBM ®	Redbooks Logo
MQSeries	OS/390
AIX	PC 300
CICS	S/390
DB2 Universal Database	SP
Everyplace	SupportPac
MQSeries	VisualAge
Netfinity	WebSphere

欢迎评论

您的评论对于我们非常重要！

希望我们的 IBM 红皮书对您大有帮助。请以下列方式之一向我们发送您对于此红皮书或其他红皮书的评论：

- 请使用以下网址中的联系我们项查看红皮书评论表格：
  ibm.com/redbooks

- 请通过互联网邮件将您的评论发送至以下地址：
  edbook@us.ibm.com

- 通过邮件将您的评论发送至第 ii 页上的地址。

第一部分

业务流程管理简介

1

业务流程管理概述

业务流程管理是产业新闻中频繁使用的术语之一。因为该术语只是最近才被引入 IT 行业，所以值得在此花些时间来阐述一下。实际上，业务流程管理并不是什么新鲜事物。企业中每个个体都在业务流程管理中扮演着重要角色。在软件行业中，此概念指的是业务流程以及与之相关的业务规则的正规化和自动化。IT 业中的业务流程管理提供了可使您获得更好的业务流程视图技术。该技术也使得业务规则更加明确并且更易于管理。从而业务经理可获得他们的所需信息以改进其业务流程。

1.1 什么是业务流程管理

在讨论或定义什么是业务流程管理或业务流程管理系统之前，我们先来讨论一下*业务流程*这一术语。

业务流程的一个定义可以是一组相互关联（由商业规则控制）的商业功能。那些商业规则对于某一企业来说是独特的，而且在某一确定的时间点上也是如此。

正如行业观察家所定义的那样，业务流程管理是理解、系统化、自动化，以及改进公司业务运作方式的一门艺术。

最近几年，广泛运用三层应用开发，至少其重要性已被认可。在三层环境中，划分了三个逻辑层次：表示逻辑层、业务逻辑层和数据访问逻辑层。该划分就每个层在不同设备上运行来说是完整的。我们来看基于浏览器应用的实例。由 HTML 驱动的浏览器负责表示层。业务逻辑封装在应用服务器中（服务件、Enterprise javabeans 等等）。从应用服务器上访问的数据可以由远程机器上的数据库服务器来管理。

在三层环境中，下一步就是业务流程管理。现在，从业务逻辑层中提取封装在业务逻辑层中的业务逻辑和商业规则，并且在基于工作流的环境中被表示，形象地阐述业务流程不同步骤。在每个节点，商业规则用于选择下一个节点和执行的业务逻辑。

结果，商业规则也就变得更加明确、可见和快速可变了。这使得公司对市场的变化能够做出更加迅速的反应。

业务流程管理也可以被看作是文件工作流和企业应用集成的紧密结合。历史工作流应用是以人为导向的。也就是说每个人都得面对他或她必须完成、批准或执行的电子文档。而在一个企业应用集成环境中，应用程序彼此连接并在无人干涉的情况下进行合作。典型地，应用程序用消息系统来交换信息。业务流程管理是如下两种技术的结合：以人为导向的工作流和应用集成的结合。

业务流程管理并不是什么新事物，但管理和改进业务流程执行技术的使用却是最近的事情。业务流程管理的技术实现被称为业务流程管理系统（BPMS）。

以下是 BPMS 所必需的功能：

- 迅速实现商业规则和商业目标改变的能力
  BPMS 必须提供实现改变的必要技术并确保业务经理能够对正在改变的业务状况作出迅速的反应。

- 测量这些改变的影响的能力
  能够做出迅速改变固然是重要的。但是这一改变有什么作用呢？很明显，BPMS 必须提供现在比改变以前更好的业务流程信息。该信息可能包括更迅速的执行或更便捷的操作。

- 将什么和如何、资源管理，以及流程独立的分离

- 以前后一致的方式定义、改变和实现业务流程。
  有能力近于实时地改变业务流程的实现是有价值的，但没有什么比企业的完整性更重要。

没有 BPMS，用户更关注的是管理业务流程而不是商业目标。BPMS 应该通过在适当的时间提供给用户适当的工具以实现其商业功能来减少管理方面的投入。

1.2 BPMS的技术组件

从上面给出的需求概要可以看出，业务流程管理系统显然并不是项简单的技术。我们可以将其分解成若干组件并分别加以讨论。

流程引擎

流程引擎是 BPMS 真正的核心部件。它实现业务流程同时管理活动的启用和终止或商业功能。流程引擎不应对业务活动如何实现施加影响。

资源管理器

资源管理器使实现商业功能或活动所必须的资源具有可用性。这些资源包括人力资源或者商业活动所必需工具和文档。

调度程序

由于受资源可用性的限制，商业功能就会经常受时间上的约束。因此，需要调度程序以使时间约束和资源可用性相匹配。

审计管理器

审计管理器是关键组件。其必须能够跟踪谁在什么时间做了什么。审计管理器的输出是理解业务流程并找到改进方法的关键。

安全管理器

该组件进行资格授权。对于给定的业务流程，授权给谁去管理呢？假设 BPMS 管理公司的核心资格，那么对其访问并使其完好无损是至关重要的。

1.3 为何如此重要？

如此关注业务流程管理有多个原因。首先，很显然，每个企业都面对快速变化的业务环境。发现这些变化的不应是运行商业应用程序的 IT 部门而应是公司的业务分析人员，他们会发现变化需求。业务分析人员需要实现更新的商业规则的工具，而这种更新并不需要重新设计整个 IT 系统，甚至无需 IT 部门的参与。这些业务流程管理工具将*为什么，以及怎样*分离，同时也将来自 IT 基础结构的业务流程及其度量分离。这就导致业务流程趋于独立，从而业务管理流程代替 IT 管理流程。

BPMS 不仅助于您应对商业环境变化，还可以有助于您应对 IT 层次的变化：更新应用程序、采购应用程序、更改应用程序。BPM 技术的应用使其影响更为清晰。

促使我们研究业务流程管理的另一方面原因是企业内部应用程序的新型用户的增加。互联网的流行已经开拓了客户以及商业合作伙伴与公司交互的新方式。WEB 驱动应用程序可极大地增加客户满意度。下一步就是要将您的商业合作伙伴带入您公司的 IT 环境。他们也期待着更多更灵活的交互方式，例如运用 B2B 整合和互联网技术。

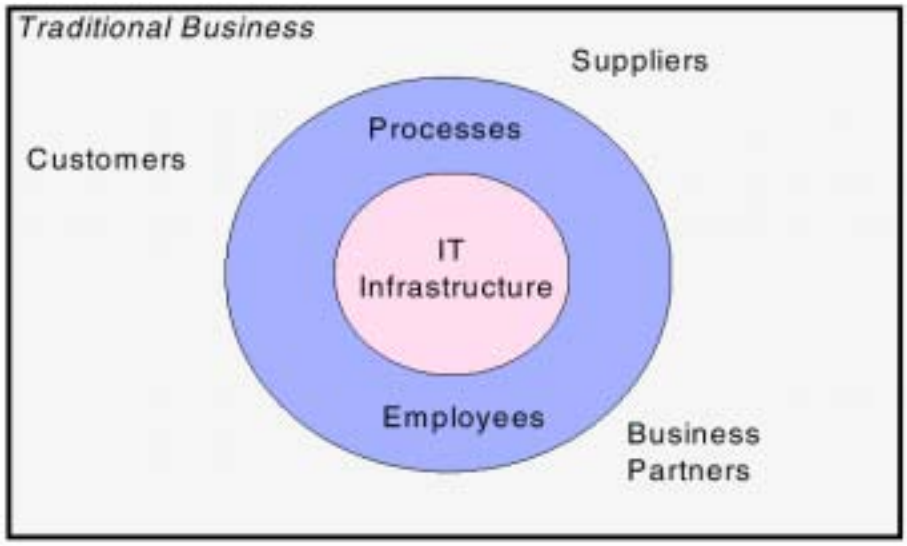

图 1-1 历史商业结构

为了应付这些新渠道并确保您公司通过所有渠道提供一致服务，您公司现存的 IT 基础结构就显得不是那么灵活了。它只是为公司内部使用而设计，而不是为您客户和商业合作伙伴设计。在历史商业中，公司外部是您职员，以及他们管理业务和个人流程的方法（参看图 1-1）。在电子商务中，如图 1-2 所示，公司外部也由 IT 基础结构组成。客户和商业合作伙伴通过 Web 直接和公司的 IT 基础结构交互。以前由面向客户的公司职员执行业务流程和商业规则，现在应与 IT 基础结构相结合。但是简单数据连通性和应用集成是不足以应付这些挑战的。需要数据连通性和应用集成来驱动业务流程，但它们缺乏和人员组合的集成。

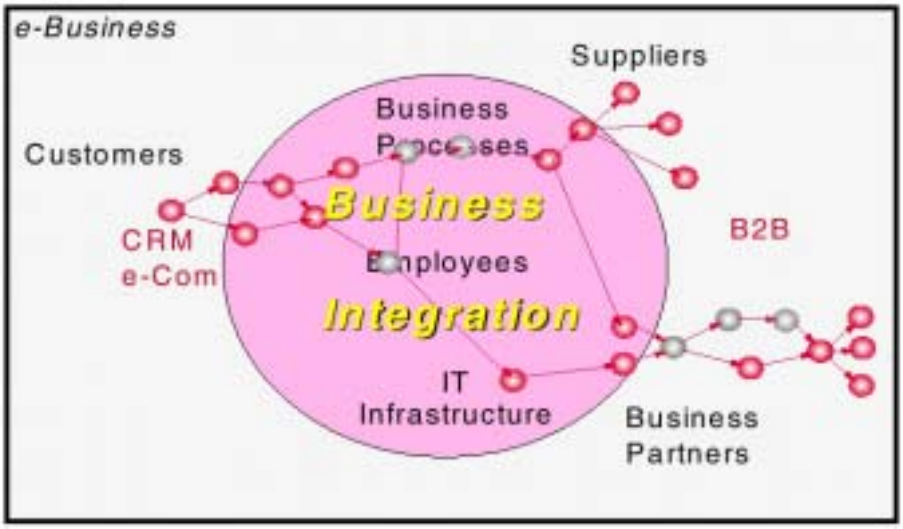

图 1-2 电子商务架构

最后，当今世界，公司的商业目标控制商业是其关键所在。业务流程的执行必须如实反映公司的商业目标。但是在进行控制之前，理解是很重要的。您需要对公司的业务有个整体观察，从业务流程一直到参与其中的资源和资本。应将业务流程管理杠杆作用知识植入您的应用程序，以及运转公司业务的人员头脑之中。在您公司的运作过程中，正规化的业务流程将会帮您获得新的洞察力。不同职员的角色和他们与供应商和客户的关系会变得更加清晰。一旦业务流程管理工具成为实际产品，它会产生有用的数据来帮助业务分析人员理解和测量业务周期。工作流环境中产生的数据允许他或她识别什么活动和子流程花费的时间最多或者使用的资源最多。换句话说，业务流程管理工具提供给您所需要的数据，从而使您能够决定在什么地方改进业务流程，成本是多少，以及可期待的利润是多少。

推动公司进行业务流程管理的根本力量是技术、全球化，以及客户力量。技术已经消除了时域和地域的界限，并且正在创造无时不在的全球市场。但是技术也已经赋予了消费者权力，只需单击一下鼠标，客户就可以离开您的公司。很显然，互联网经济已经改变了客户的期望。互联网的存在已不足以吸引客户。客户也已变得更加没有耐心和拥有更大的选择权力。当他或她在网站上订购物品时，客户显然想要知道当前订单的状况。他或她想知道产品什么时候发货，货物放在门口的什么地方等等。

这就是现在客户的权力，而不是下一个工作日的事情。客户、合作伙伴和供应商都被直接链接到了您系统之中。电子商务技术已经紧密结合到 IT 系统和业务流程当中了。

2

商业案例概述

在本章中，我们将阐述虚拟公司 BuyXYZ.com 的商业案例。我们还将阐述用于解决方案的一些技术需求。

2.1 订单处理应用程序

BuyXYZ.com 是一家在线零售商。客户通过 Web 输入他们的订单。订单处理系统在接到订单时，检验并确认客户 ID 和存货清单系统。如果产品有存货，将会把订单转发到会计应用程序和配送应用程序中进行处理。

如果产品没有存货，BuyXYZ.com 将联系产品供应商并且订购所需数量的产品。

当主供应商不能交付所需产品时，BuyXYZ.com 依赖于多个后备供应商为其供货。然而，我们必须平衡利润和客户服务。即使试图在所有情况下满足客户期望，我们仍要确保获取利润。因此，必须进行核算以确保后备供应商的索价不高于售价。

2.2 现有应用程序

正如本书第 3 页“ 第一章 业务流程管理概述 ”所述，业务流程管理被视为企业应用集成演变。在 BuyXYZ.com 案例中，连接多个现有应用程序以实现进程管理的全部目标。

BuyXYZ.com 现有应用程序包括 CICS 环境下的会计应用程序。该应用程序的入口是可由 MQSeries-CICS 网桥驱动的 CICS 程序。因此，会计应用程序的集成是通过发送 MQSeries 消息给由 MQSeries-CICS 网桥监听事务处理程序服务的队列来实现的。入口程序的名称在 MQSeries 消息中编码。驱动该集成的消息流将在本书第 226 页的 5.7 节“ 创建 BuyXYZ_Order_Entry_CICS 消息流 ”中讨论。

管理仓库和预订货物配送的应用程序作为企业 JavaBeans 的集合在 WebSphere 应用服务器中运行。业务流程模型对 EJB 的集成将在本书第 301 页的“ 第七章 工作流程中的企业 JavaBeans 调用 ”中讨论。

2.3 系统需求

解决方案将依靠多个技术组件实现。需要服务于 HTML 请求的 Web 服务器。需要存储客户数据和存货清单数据的数据库服务器。需要在不同组件之间提供路由和消息传送的消息代理。最后，还需要工作流服务器用于控制所有组件的执行。

所有这些组件都可能出现故障而导致整个系统在单个组件出现故障期间无法使用。对于 Web 服务器，可参考《Websphere 可扩展性：WLM 和群集使用 WebSphere 应用服务器》的高级版（编号：SG24-5105）中关于如何解决单点故障问题的信息。DB2 数据库的高可用性的更多详细信息包含在《用 DB2 UDB EEE 管理 VLDB》（编号：SG24-5105）中。贯穿本红皮书始终的解决方案阐述了一项技术。该项技术解决了消息代理和工作流服务器的高可用性问题。我们将配置两台消息代理和两台工作流服务器。消息代理集中所有出现的消息流，并且使用其私有代理数据库。这些组件的负载均衡和接管基于 MQSeries 群集技术实现。

然而，要注意到前面提到的系统可用性和负载均衡的解决方案并非完整的解决方案。如前所述，其是基于 MQSeries 群集技术的。这就意味着两个系统之间的 MQSeries 通信故障将导致消息发送到其他系统。然而，如果 MQSeries 通信正常，但是例如消息代理停止了，那么消息将仍会发送到空闲代理中。

关于高可用性和负载均衡的完整解决方案以后将在各个单独红皮书的主题中加以讨论。

2.4 实施步骤

我们将在多步骤的 MQSeries Workflow 中实现商业模式。首先，我们做个简单模型，所有作业都自动运行。将该模型应用到 MQSeries Workflow 正常正常正常运行时环境后，当供应商不能满足需求时，我们可演示可能发生什么。然后，为了包含后备供应商将改进模型。根本原因是 BuyXYZ.com 提出了满足与其客户达成服务水平的协议。商务规则的变化，从同一商品的一个供应商到多个供应商，能够快速地在 MQSeries Workflow 中实现，从而表明业务经理能够对商业情况的变化迅速做出反应。

然而，当后备供应商开始索要高价时，边际利润会受到威胁。为了避免这样，当边际利润受挤压时，工作流模型被更新以融入审批步骤。然后，销售经理做出判断决定公司是否因为情况特殊而减少边际利润。例如，销售人员正在与客户商谈一项大合同。如果不能履行当前低边际利润小订单，可能会影响客户的满意度和达成交易的愿望。因而，人工审批步骤允许销售经理以调整的方式处理异常。

最后一步，我们看到 MQSeries Workflow 中的可利用审计信息并且使用该信息生成报告。然后使用这些审计信息进一步改进业务流程。

第二部分

实现业务流程管理

解决方案

3

技术组件的配置

在本章中，我们将讨论 MQSeries Workflow 环境和 MQSeries Integrator 环境的配置。在配置中，MQSeries Workflow 服务器功能将由两台设备支持，它们共享第三台设备上的正常正常正常运行时数据库。为了达到验证的目的，我们将讨论具有基于浏览器接口实例应用程序的应用。

两个 MQSeries Integrator 代理分别配置在两台独立的设备上。MQSeries Integrator 配置管理器和用户名服务器功能由第三台设备支持。为了验证配置，我们将应用简单的消息流。

最后，详细定义 BuyXYZ.com 订单处理应用程序所需的资源。

3.1 目标环境

在第 11 页的第二章“ 业务案例概述 ”中已给出可满足需求的目标环境，并且该目标环境由六台设备组成。该配置提供了设备间的某种负载均衡。我们还选择了中央数据库仓库，它由用于 MQSeries Workflow 正常正常正常运行时数据库的 MQSeries Workflow 设备，以及运行 MQSeries Integrator（以支持配置和消息仓库数据库）的设备所共享。这样就为多设备环境中的大部分数据库提供了管理中心点。设备之一可作为 Web 服务器工作，同时支持 MQSeries WorkflowWeb 客户机。另两台设备可作为 MQSeries Workflow 服务器，其余两台设备运行 MQSeries Integrator 代理。

为了检验 Web 服务器、MQSseries 工作流服务器和数据库设备的安装，我们可使用 IBM 网站上的 Web Credit 实例程序，例如 SupportPac WA82：
Http://www-4.ibm.com/software/ts/mqseries/txpacs/wa82.html

为了检验 MQSeries Integrator 的安装，我们将创建和应用非常简单的信息流并通过它发送一些消息。如果您喜欢，可以使用如下链接中的 MQSeries Integrator 实用程序之一：
Http://www-4.ibm.com/software/ts/mqseries/txpacs/txpml.html

3.1.1 硬件和软件

本文档所述方案中使用的设备将在第 393 页的附录 A“ 硬件和软件配置 ”中加以描述。所有设备通过 TCP/IP 网络彼此连接，并且属于同一个 Windows NT 域。没有任何设备可以既是主域控制器又是备份域控制器。

配置中所用设备上安装的所有软件将在第 393 页的附录 A“ 硬件和软件配置 ”中详细描述。在服务器上开始安装和配置 MQSeries Workflow 和 MQSeries Integrator 之前，我们已经使用这些产品的程序文档安装了许多其它产品：

- Web 服务器
  - DB2 通用数据库版本 7
  - MQSeries 版本 5.2
  - WebSphere 高级版 版本 3.5.3

- MQSeries 作流服务器
  - DB2 通用数据库 版本 7
  - MQSeries 版本 5.2
- MQSeries Integrator 服务器
  - DB2 通用数据库 版本 7
  - MQSeries 版本 5.2
- 数据库服务器
  - DB2 通用数据库 版本 7
  - MQSeries 版本 5.2

3.1.2 拓扑图

图 3-1 描述了本文档所讨论和使用的设备拓扑图。

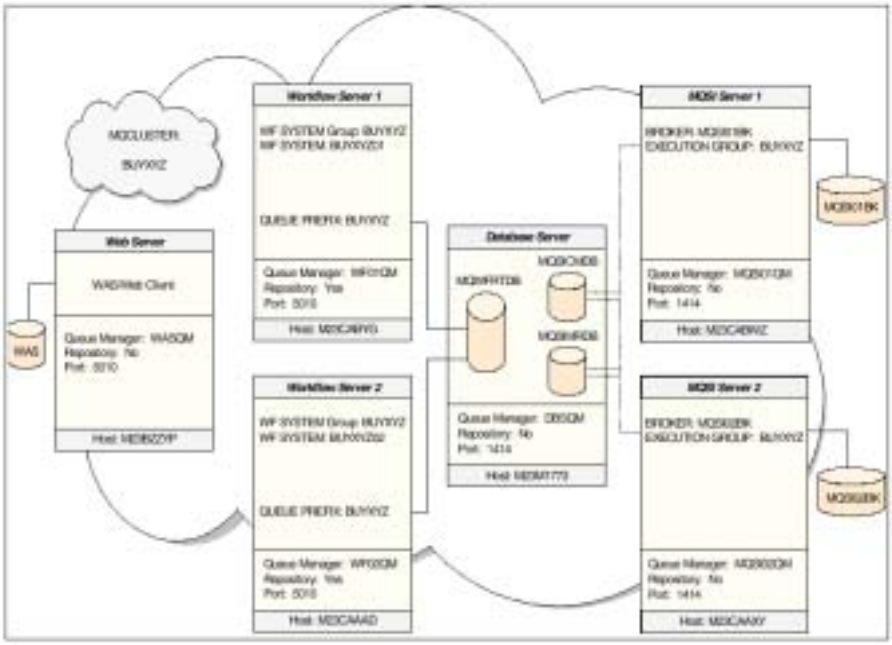

图3-1 拓朴视图

MQSeries Workflow 客户机通过 Web 服务器与工作流系统连接。作为 MQSeries Workflow Web 客户机的功能部件，服务件通过发送 MQSeries 消息与两台工作流服务器通信。服务件与在 Web 服务器上运行的队列管理器 WASQM 连接。利用 MQSeries 群集技术，该队列管理器将其负载分布到与两个工作流服务器相关联的两个队列管理器上。通过使用 MQSeries 群集技术，我们还可获得故障恢复机制。当其中一个工作流服务器连接中断，来自于 Web 客户机的所有工作流请求将路由到另一台活跃的工作流服务器上。在 MQSeries Workflow 手册中，您将看到该实现技术有时也被称为*客户机集中器*。

注意，这并非完善的故障恢复机制。如果一台服务器上的工作流子系统停止运行，而 MQSeries 连接仍是活跃的，请求仍会路由到这台服务器上。MQSeries 群集并不知道接收应用程序的实际情况，即 MQSeries Workflow 并不处理消息。

正如我们将在下一章中所讲到的，使用 MQSeries Integrator 中的消息流执行工作流中的某些活动。在此，我们再次尝试使用某种故障恢复机制来获得负载均衡系统。两个消息代理应用在两个系统的适当位置上，每个代理将支持所有消息流。这样，当 MQSeries Workflow 向 MQSeries Integrator 发送活动消息时，MQSeries 群集会将负载分布到两个消息代理上。如果与一个消息代理的通信中断，那么所有消息将会路由到没有中断的系统中。在此，仅在通信中断时，工作流服务器上的 MQSeries 知道了这一变化，从而再次成功实现了故障恢复机制。当其中一个消息代理停止工作而系统通信正常时，仍不会触发故障恢复机制。就此来说，该拓扑仍有其他故障点。仅有一个 Web 服务器与浏览器相连，并且 MQSeries Workflow 和 MQSeries Integrator 所依赖的数据库服务器并未通过备用设备加以备份。尽管 WebSphere 和 DB2 具有解决这些单点故障问题的解决方案。然而这并不在本红皮书所涉及的范围之内。关于此主题或其它主题的附加读物列在第 407 页的“ 相关出版物 ”一节中。

本章主要关注的是安装和配置 MQSeries Workflow 和 MQSeries Integrator 以实现带有故障恢复机制的负载均衡系统。

3.1.3 命名规则（Naming conventions）

在这个阶段决定命名规则是很重要的。您将需要在安装和配置拓扑期间为队列管理器、群集、数据库等命名时使用这些规则。在创建带有队列管理器的工作流系统时，MQSeries Workflow 将在规则范围内创建一些 MQSeries 对象，例如群集发送通道和群集接收通道。此时可能发生 MQSeries Workflow 的命名规则与您的标准所强加的命名规则不匹配的情况。

请记住某些产品不能成功使用“网络管理员”（Administrator）用户 ID 进行安装。因此如必要，为了确保安装成功，请使用管理员组成员中的另一 ID 进行安装。

3.2 MQSeries Workflow正常运行时数据库

同一系统组中的 MQSeries Workflow 服务器共享同一正常正常正常运行时数据库。因为希望创建包含多个工作流服务器的工作流环境，所以我们必须创建能够由所有工作流服务器共同使用的正常正常正常运行时数据库。

我们将在设备 M23M1773 上保存 MQSeries Workflow 正常正常正常运行时数据库。在本案例中，我们不需要安装全部 IBM MQSeries Workflow 产品，而只需安装其管理组件。这些组件包括生成正常正常正常运行时数据库的实用程序，以及使用以后可在其创建期间通过工作流服务器获得的信息来启动它的实用程序。参考本书第 35 页的 3.3 节“MQSeries Workflow 服务器”以获取关于如何连接工作流服务器和远程正常正常正常运行时数据库的信息。

对于所有已安装 MQSeries Workflow 软件的设备来说，在安装 MQSeries Workflow 之前，我们已经安装了 MQSeries 和 DB2。这并不是必需的，因为如果这些产品尚未安装，MQSeries Workflow 将会自动安装它们。但是预安装 MQSeries 和 DB2 将带给您更多的灵活性。为了在数据库服务器上安装必需的 MQSeries Workflow 组件，安装步骤如下。

在启动安装程序之后，首先要选择所需的语言，本案例中使用是英语。

图3-2 选择安装语言

现在选择安装位置和使用的目录名。为了保持路径名简短，我们将把默认目录名改为MQWF。

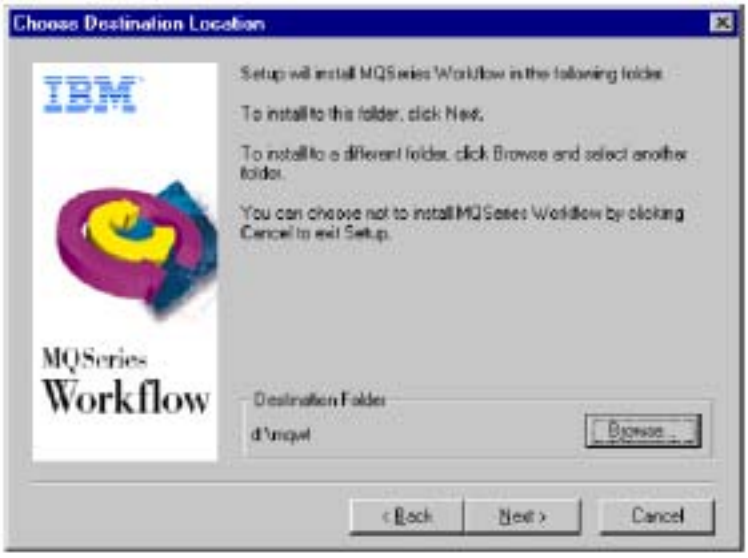

图3-3 给出安装目录

现在选择**行政管理组件（Administrative Components）**，因为它包含了创建数据库所需的所有组件。

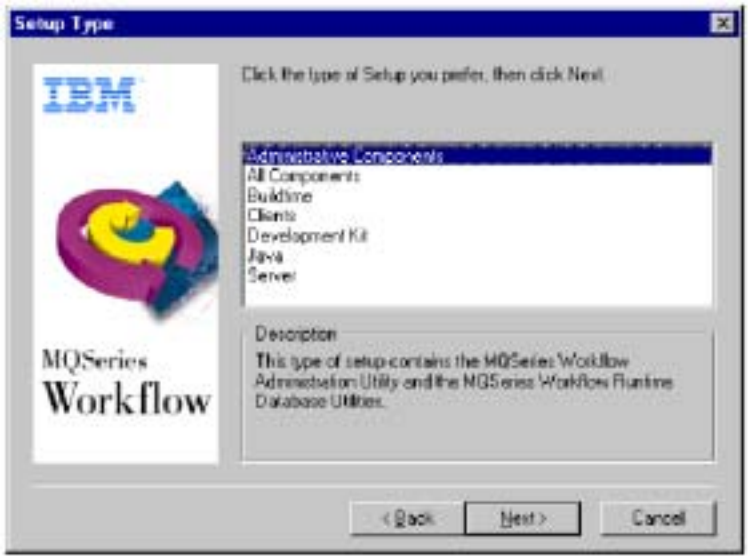

图3-4 选择安装类型

然后，选择**管理实用程序（Administration Utility）**和**正常运行时数据库实用程序（Runtime Database Utilities）**。

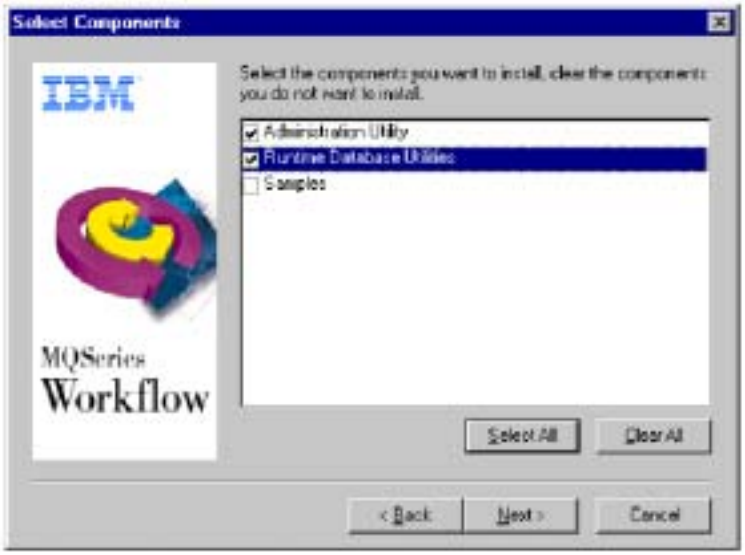

图3-5 选择安装 MQSeries Workflow 组件

在下一个窗口中给出程序文件夹名称。

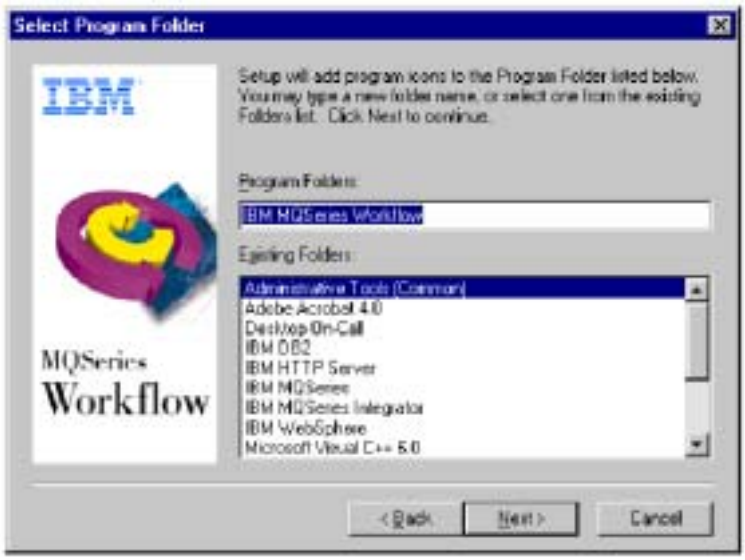

图3-6 给出程序文件夹名称

最后，单击**下一步（Next）**完成安装。

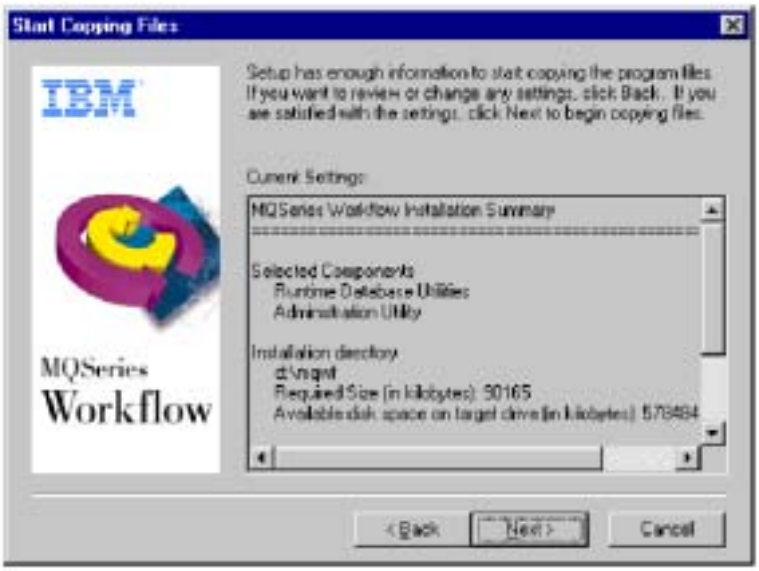

图3-7 安装摘要

在安装结束后，将显示 MQSeries Workflow 配置窗口。然而，我们建议您取消该窗口并使用开始菜单重新调用一个新的实例，以确保获得正确的参数值。

3.2.1 使用配置实用程序创建第一个配置

MQSeries Workflow 将使用称为配置 ID 的概念存储配置参数值。在某一时间点上，您可以在任一系统上定义多个配置 ID。

现在，我们需要通过运行在 MQSeries Workflow 服务器 01 上的 MQSeries Workflow 配置实用程序来创建正常正常正常运行时数据库。

我们将使用下列名称来配置环境：
配置名：XYZ01
队列管理器名：WF01QM
服务器名：BUYXYZ01
系统组：BUYXYZ
数据库名：MQWFRTDB

通过选择 **Start** -> **Programs** -> **IBM MQSeries Workflow** -> **MQSeries Workflow 配置实用程序**来启动图形化配置实用程序。

如图 3-8 所示。

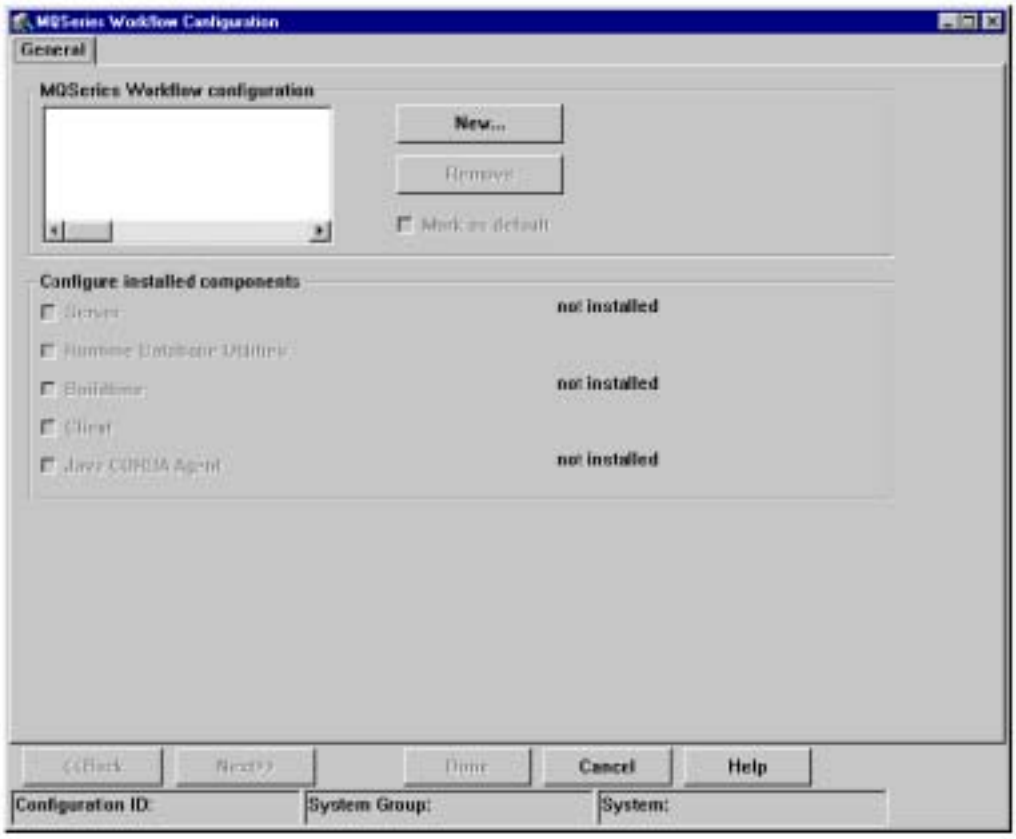

图 3-8 MQSeries Workflow 配置实用程序

首先，通过单击**新（New）**来告诉 MQSeries Workflow，我们想创建新的配置。这时会出现小窗口要求我们输入已选择的配置 ID。

输入该信息，本实例 XYZ01 中，单击**确定（OK）**继续。

在如图 3-9 所示的窗口中，选择刚刚定义的 MQSeries Workflow 配置 **XYZ01**，然后选择**正常运行时数据库实用程序（Runtime Database Utilities）**选项。

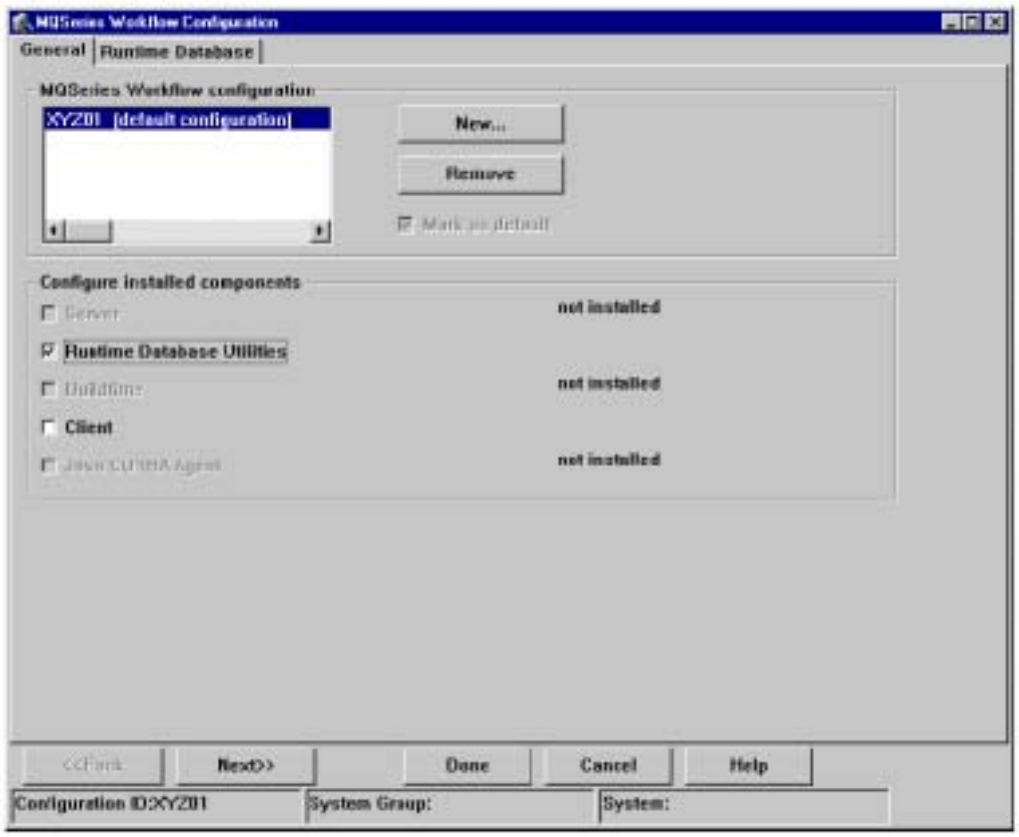

图 3-9 创建一个新的配置

单击**下一步（Next）**继续。

现在，我们将查看所有先前创建的数据库列表，如图 3-10 所示。因为我们正在创建新的数据库，所以选择 **DB2** 实例并单击**新（New）**。

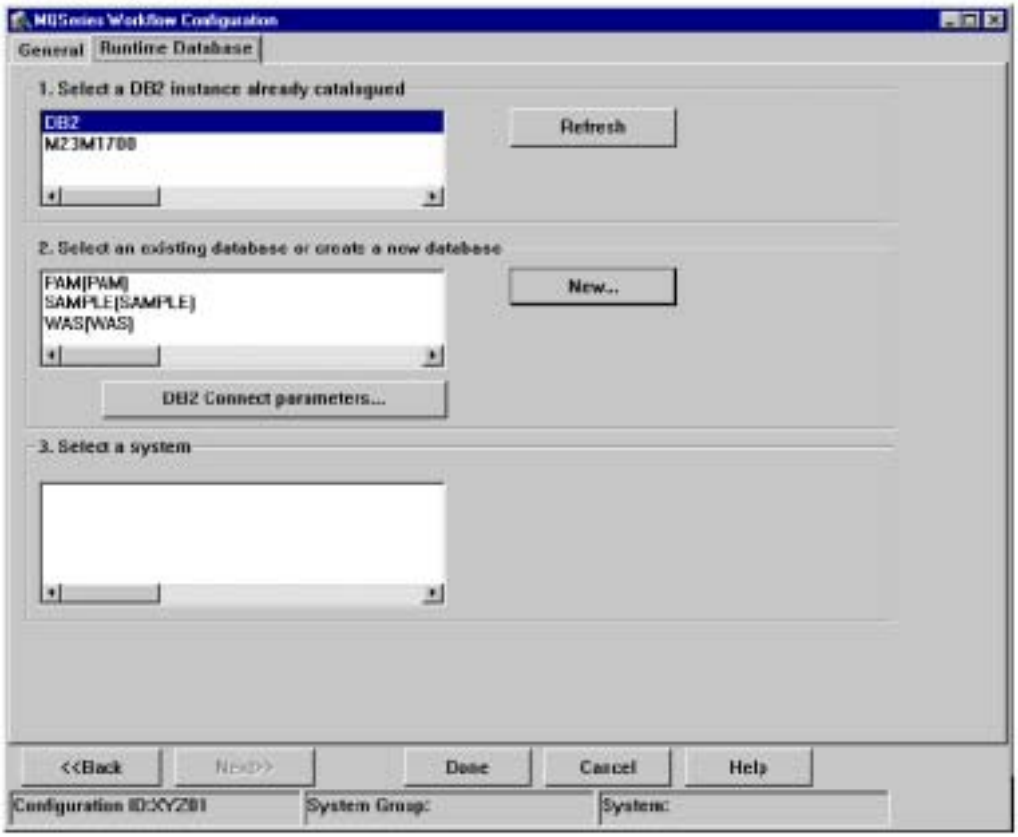

图 3-10 现有的 DB2 数据库和实例

使用如图 3-11 所示的已选名称填写所有字段并单击**确定(OK)**。为了获得最佳性能和可用性，您可以考虑使用不同磁盘作为 DB2 容器和 DB2 日志文件存储位置。

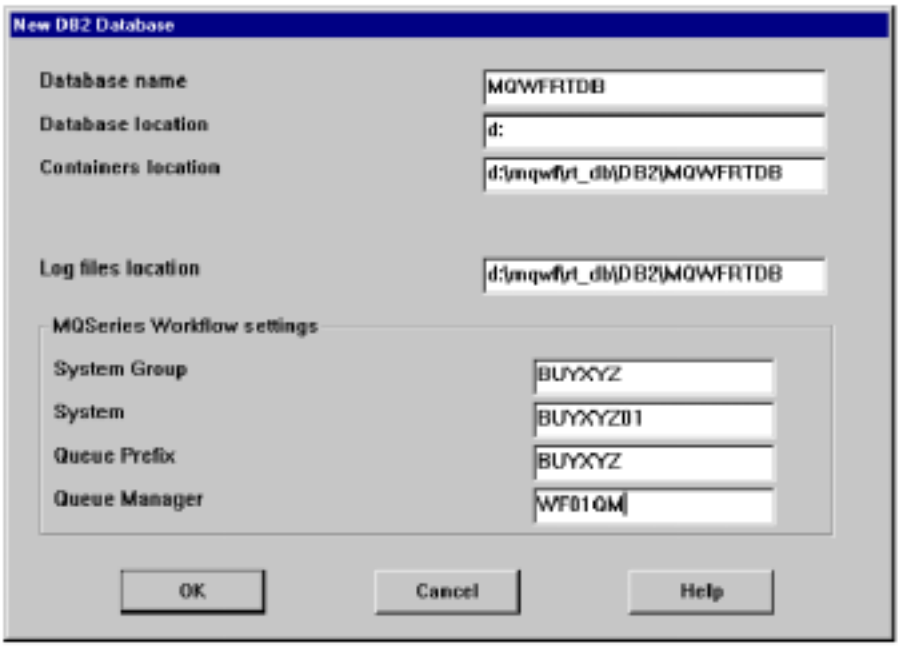

图 3-11 提供 MQSeries Workflow 配置参数

现在选择最新创建的系统，如图 3-12 所示。

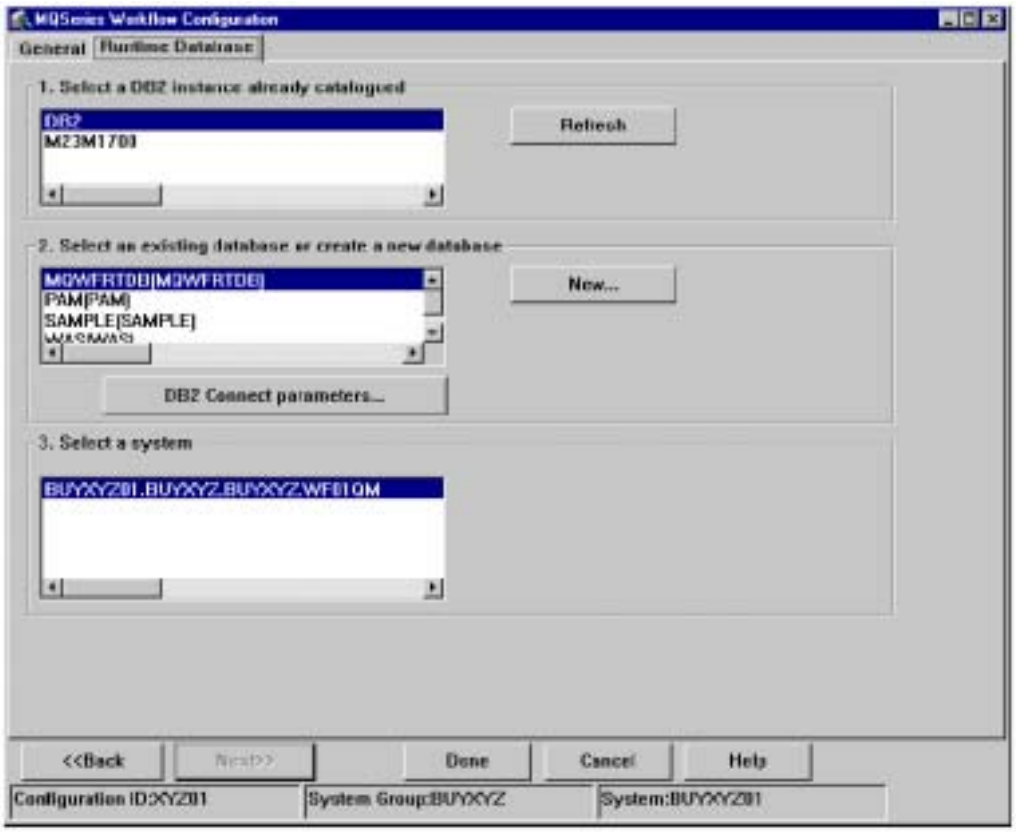

图 3-12 选择已创建的配置

现在将出现要求我们输入 DB2 用户 ID 和密码的窗口。
输入该信息并单击**确定(OK)**。

返回如图 3-12 所示窗口，单击**完成(Done)**。

图 3-13 中的窗口显示表明当前正在创建数据库的进程条。这可能会花费一段时间，时间的长短取决于设备的速度。

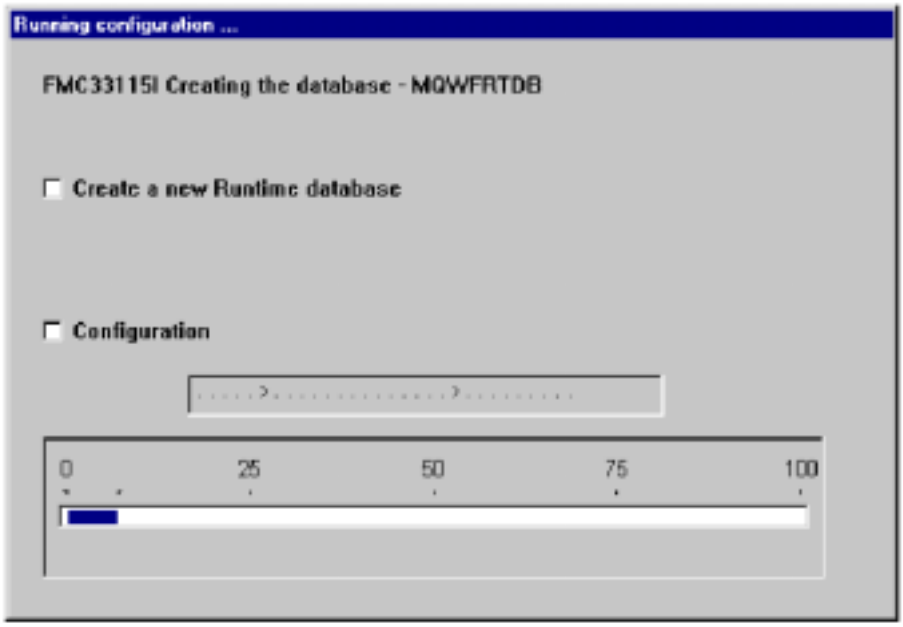

图 3-13 数据库创建进程

图 3-14 所示窗口表明数据库安装成功，现在我们可以单击**确定(OK)**继续。

图 3-14 配置已完成

现在，我们已经完成了 XYZ01 的配置。也就是说，我们已经创建了正常正常正常运行时数据库并在数据库服务器中记录了关于第一个工作流服务器的信息。注意，我们仍需定义实际的工作流服务器和队列管理器。

3.2.2 使用FMCZUTIL创建第二个配置

对于第二个配置来说，我们使用了基于文本的 DOS 实用程序 FMCZUTIL。该基于文本的实用程序将使为对象命名更加灵活性，并且必须在网上定义和发布这些对象。

我们将在第二个配置中使用以下名称：

配置名：XYZ02
队列管理器名：WF02QM
服务器名：BUYXYZ02
系统组名：BUYXYZ
数据库名：MQWFRTDB

这将创建带有 XYZ02 配置 ID 的第二个配置。这样不会删除现有配置。

下面的脚本详述了现在需要执行的事件序列：

```
C:\>fmczutil
FMC33201I Configuration Commands Menu (FMC33201I 配置命令菜单) :
l ...List
s ...Select
c ...Create
x ...Exit Configuration Commands Menu (退出配置命令菜单)
c
Configuration identifier  (配置标识符) :[FMC ]XYZ02

FMC33210I Select Category Menu (FMC33210I 选择目录菜单) :
i ...( )Runtime Database Utilities (正常正常正常运行时数据库实用程序)
c ...( )Client
a ...all
n ...none
x ...Exit Select Category Menu (退出选择目录菜单)
i
FMC33210I Select Category Menu (FMC33210I 退出选择目录菜单) :
i ...( X )Runtime Database Utilities (正常正常正常运行时数据库实用程序)
c ...( )Client
a ...all
n ...none
x ...Exit Select Category Menu (退出选择目录菜单)
x
-    Configuration of Runtime database ... (配置正常正常正常运行时数据库......)

u ...( )Use an existing Runtime database (使用现有运行时数据库)
n ...( X )Create a new Runtime database (创建新运行时数据库)
u
```

```
DB2 instance :[DB2 ]
DB2 database :[ ] MQWFRTDB
DB2 user ID of database administrator （数据库管理员的 DB2 用户 ID）:[vdputteg ] db2admin

DB2 user ID to access Runtime database（DB2 用户 ID 访问正常正常正常运行时数据库） :[vdputteg ]
db2admin

System group name（系统组名称） :[FMCGRP ]BUYXYZ
System name（系统名称） :[FMCSYS ]BUYXYZ02
Queue manager name（队列管理器名称） :[FMCQM ] WF02QM
Queue prefix（队列前缀）:[FMC ]BUYXYZ
c ...Create configuration profile for 'XYZ02'now（现在创建'XYZ02'配置文件）
s ...Save input to file（将输入保存到文件）
r ...Review/change input（查看/改变输入）
x ...Exit (input for configuration 'XYZ02'will be lost)（退出（配置'XYZ02'的输入即将关闭））
c
- FMC33680I The profile for the configuration 'XYZ02'was updated successfully（顺利更新'XYZ02'
配置的文件 FMC33680I）
- Do you want to load the FDL for system 'BUYXYZ02'into the Runtime database（您想要将系统
'BUYXYZ02'的 FDL 加载到正常正常正常运行时数据库吗？）
MQWFRTDB'now?
y ...Yes
n ...No
y
MQ Workflow user ID to import FDL （MQ 工作流 ID 输入 FDI）:[ADMIN ]
MQ Workflow password for user 'ADMIN'（用户'ADMIN'的 MQ 工作流密码）:[] *********

Enter password for user ID 'db2admin'（用户 ID 'db2admin'的输入密码）:[ ] ********
Confirm password for user ID 'db2admin'（用户 ID 'db2admin'的确认密码）:[ ] ********
- FMC33691I Password changed（更改 FMC33691I 密码）.
FMC33131I Loading reference FDL（FMC33131I 加载参考 FDL）.
FMC20500I Start parsing d（FMC20500I 开始分析 D 盘）:\mqwf \cfgs \xyz02 \fdl \fmcznews.fdl.
FMC25100I CREATE SERVER 'ADMINSVR.BUYXYZ02.BUYXYZ'finished（FMC25100I 创建服务器'ADMINSVR.BUYXYZ02.BUYXYZ'完成）.
FMC25100I CREATE SYSTEM 'BUYXYZ02'finished（FMC25100I 创建系统'BUYXYZ02'完成）.
FMC25100I CREATE SERVER 'EXECSVR.BUYXYZ02.BUYXYZ'finished （FMC25100I 创建服务器'EXECSVR.BUYXYZ02.BUYXYZ'完成）.
FMC25100I CREATE SERVER 'PESERVER.BUYXYZ02.BUYXYZ'finished. （FMC25100I 创建服务器'PESERVER.BUYXYZ02.BUYXYZ'完成）.

FMC25100I CREATE QUEUE_MANAGER 'WF02QM'finished. （FMC25100I 创建 QUEUE_MANAGER 'WF02QM'完成）.

FMC20510I Finished parsing d（FMC20510I 开始分析 D 盘）:\mqwf \cfgs \xyz02 \fdl \fmcznews.fdl.

FMC33201I Configuration Commands Menu（FMC33201I 配置命令菜单）:
l ...List
s ...Select
c ...Create
d ...Change default configuration（改变默认值配置）
x ...Exit Configuration Commands Menu（退出配置命令菜单）
x
C:\>
```

该脚本中最重要的步骤已在第 32 页中用红笔圈出。在默认情况下，该实用程序将引导您去创建新的正常正常正常运行时数据库。使用本书第 25 页的 3.2.1 “ 使用配置实用程序创建第一个配置 ” 中创建的正常正常正常运行时数据库选择 **u**。

现在，我们已经完成了 XYZ01 和 XYZ02 的配置。因此，在启动 MQSeries Workflow 配置实用程序时，如图 3-15 所示，我们将会看到这两个新建的配置。

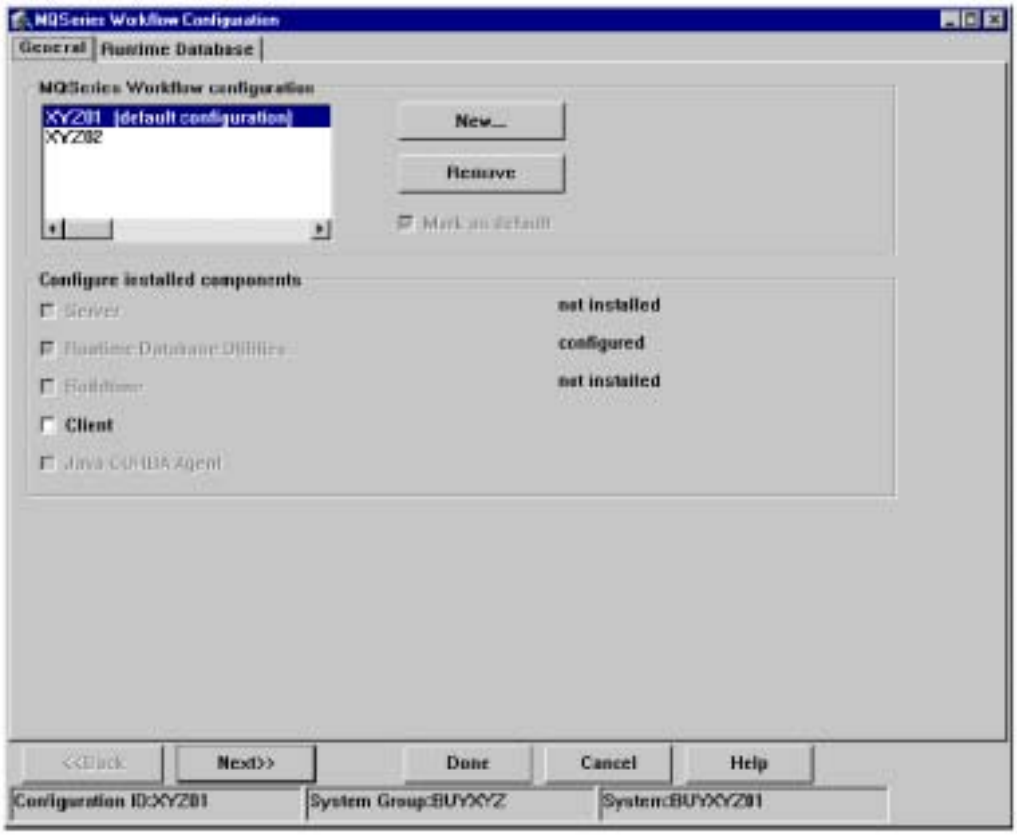

图 3-15 准备使用的两个配置

现在，MQSeries Workflow 在数据库设备上的配置部分已经完成。

3.3 MQSeries Workflow服务器

现在，我们必须在设备 M23CABYG 和 M23CAAAD 上配置 MQSeries Workflow 服务器。这两台设备上都已安装了 MQSeries 和 DB2。

3.3.1 DB2配置

首先，我们必须配置先前已创建的远程数据库 MQWFRTDB 上的 ODBC 连接。因此，需要通过选择 **Start -> Programs -> DB2 for Windows NT-> Client Configuration Assistant** 来调用 DB2 客户机配置助手。

在如图 3-16 所示的初始窗口中单击**添加(Add)**。

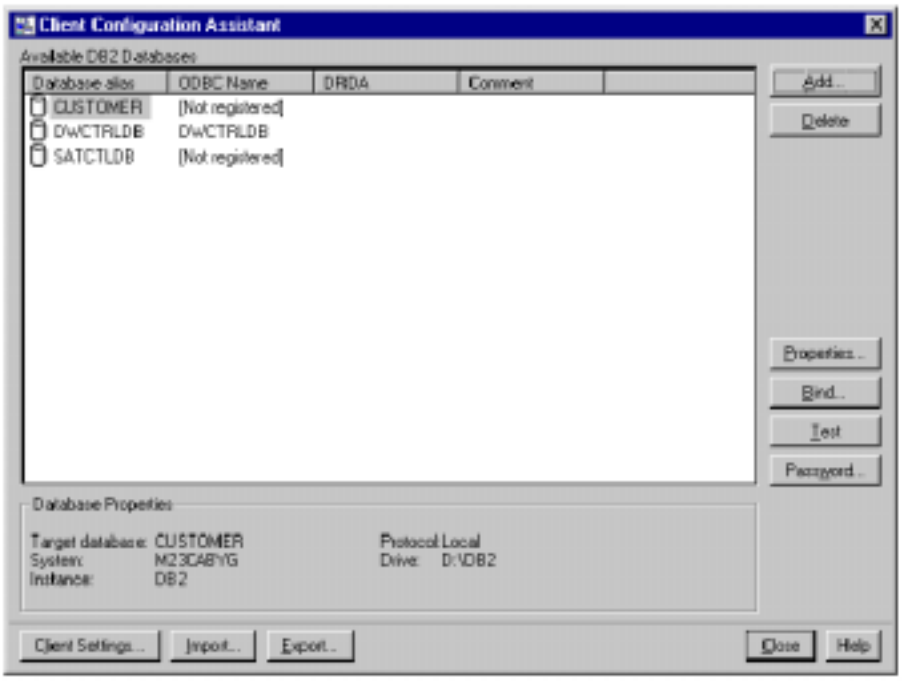

图3-16ODBC 配置－主窗口

在如图 3-17 所示的窗口中选择**手工配置连接到数据库(Manually configure a connection to a database)**，然后单击**下一步(Next)**。

图 3-17 添加数据库向导——步骤 1

如图 3-18 所示，选择 **TPC/IP** 协议，然后单击下一步**(Next)**。

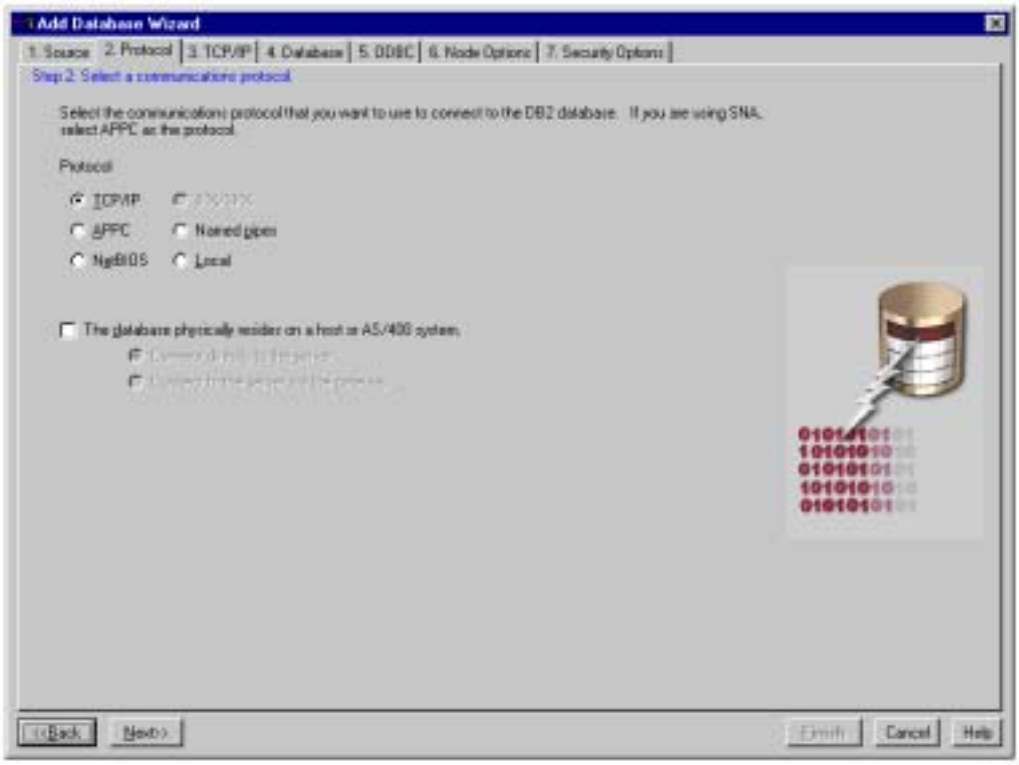

图3-18 添加数据库向导——步骤2

在图 3-19 所示的窗口中，输入目标计算机名称（本案例中，数据库定位在设备 M23M1773 上）。然后输入默认端口号 50000 并单击**下一步(Next)**。

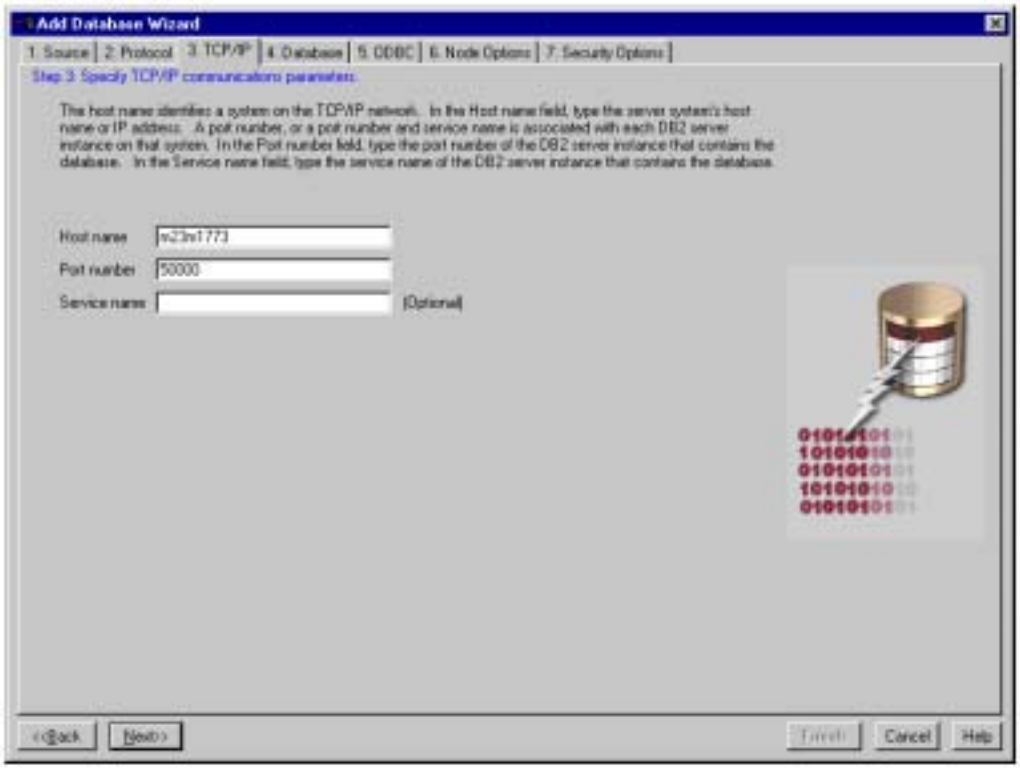

图3-19 添加数据库向导——步骤3

在图 3-20 所示的窗口中输入数据库名 MQWFRTDB 及其别名，然后单击**下一步(Next)**。

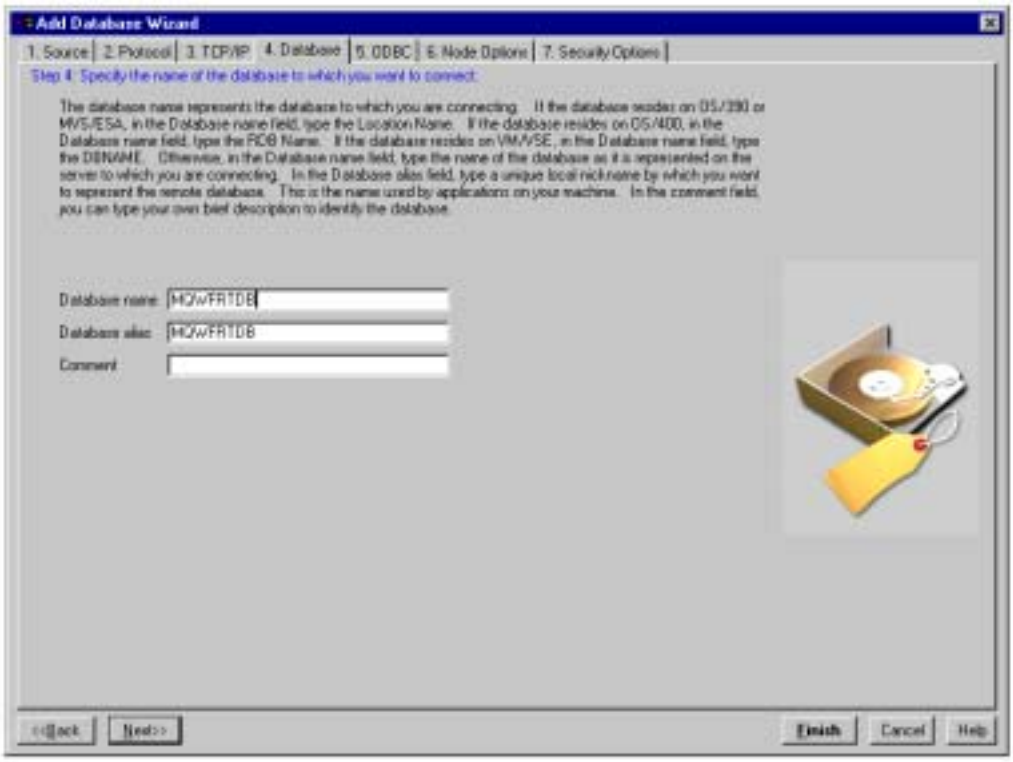

图3-20 添加数据库向导——步骤4

最后如图 3-21 所示，选择 **ODBC 注册数据库(Register this database for ODBC)**作为系统数据源并单击**完成(Finish)**。

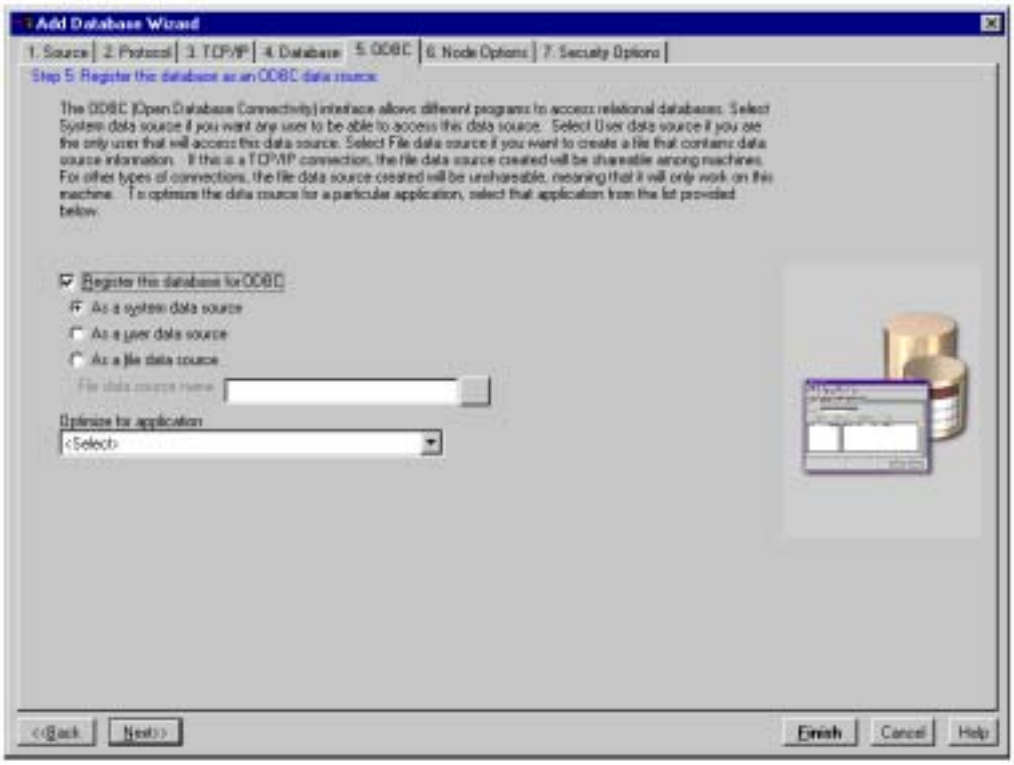

图3-21 添加数据库向导——步骤5

现在，我们可以测试一下已成功建立的连接。为此，在图 3-22 所示的窗口中单击**测试连接(Test Connection)。**

图3-22 添加数据库

在图 3-23 所示的窗口中，输入合法的 DB2 用户 ID 和密码，然后单击**确定**(**OK**)。在此使用您将分配给工作流服务器以访问正常正常正常运行时数据库的同一用户 ID。

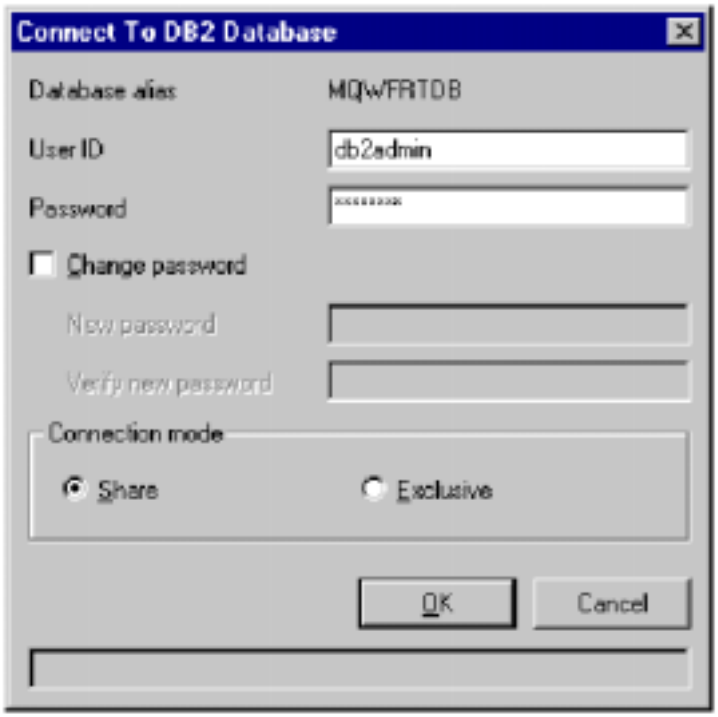

图 3-23 测试连接窗口

如果工作一切正常，将显示如图 3-24 所示的 DB2 确认窗口。

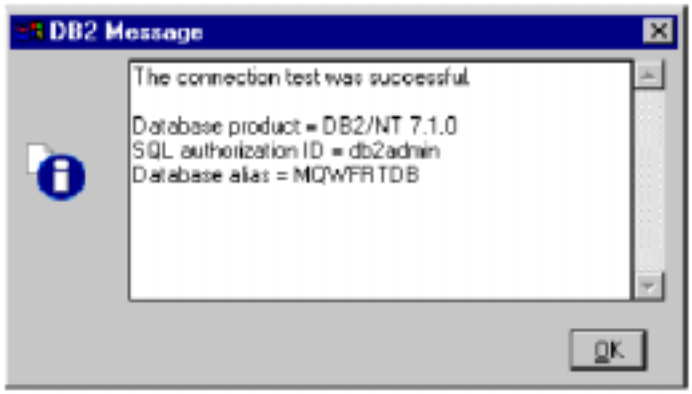

图 3-24 连接测试已完成

最后，作为确认，将在重新显示的客户机配置助手窗口中显示新建的数据库（图 3-25）。

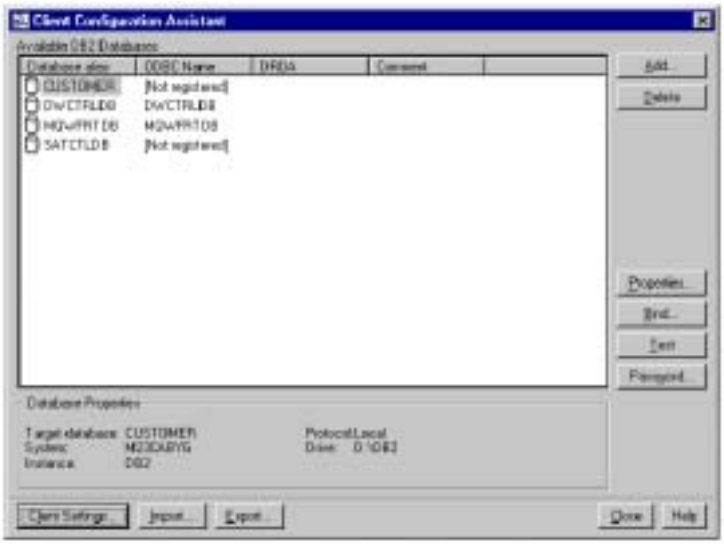

图 3-25 已注册的数据库列表

现在，我们已经配置了与正常正常正常运行时数据库连接的 DB2 客户机。下一步我们将配置工作流服务器设备。

3.3.2 MQSeries Workflow配置

现在，我们可以安装 MQSeries Workflow 产品了。我们将创建 XYZ 配置并将其指向现有的正常正常正常运行时数据库。这将强制我们使用本书第 25 页的 3.2.1 节“ 使用配置实用程序创建第一个配置 ”中所设置的参数。我们将把 WF01QM 队列管理器设置为 MQSeries 群集中的第一个队列管理器。

插入产品 CD 后，在图 3-26 所示的窗口中选择**所有组件（All Components）**并单击**下一步（Next）**。

图3-26 选择安装组件

如本书第 25 页的 3.2.1 节“ 使用配置实用程序创建第一个配置 ”中所述，为产品安装提供合适的目录名和程序文件夹名。

在安装完成之后，通过选择 **Start -> Programs -> IBM MQSeries Workflow -> MQSeries Workflow Configuration Utility（配置实用程序）**调用图形化配置实用程序。

虽然我们将指向数据库服务器上的现有配置，但是任何运行在工作流服务器上的配置都是新配置的一部分。因此，在配置实用程序的主窗口（如图 3-27 所示）中选择**新建（New）**。

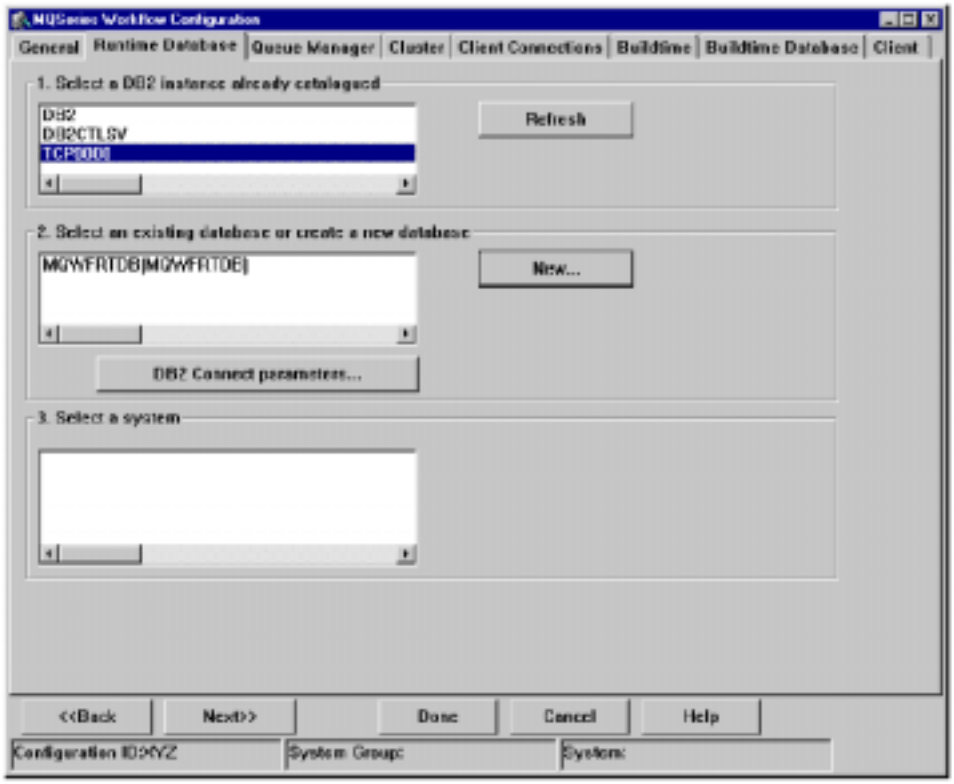

图 3-27 MQSeries Workflow 配置实用程序——主窗口

输入 XYZ 作为配置 ID 并单击**确定（OK）**。

图 3-28 给出配置 ID

现在，在图 3-29 中选择配置 XYZ，然后选择**服务器（Server）**、**运行时（Runtime）**、**数据库实用工具（Database Utilities）**、**构建时（Buildtime）** 和**客户机（Client）**。

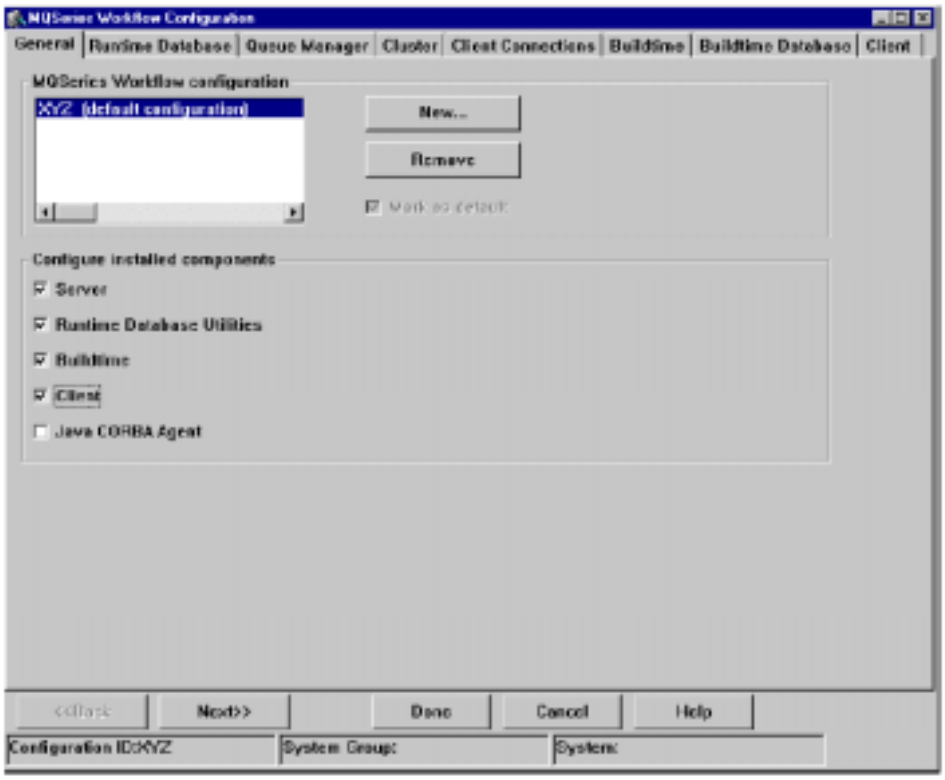

图 3-29 选择配置组件

单击**下一步（Next）**，将显示当前 DB2 实例的列表（图 3-30）。选择 **TCP0000**，将提示我们输入 DB2 用户 ID 和密码。正确输入之后单击**确定（OK）**，我们将在面板 2 中看到现有数据库的列表，如图 3-29 所示。

选择数据库 **MQWFRTDB** 并单击**下一步（Next）**。

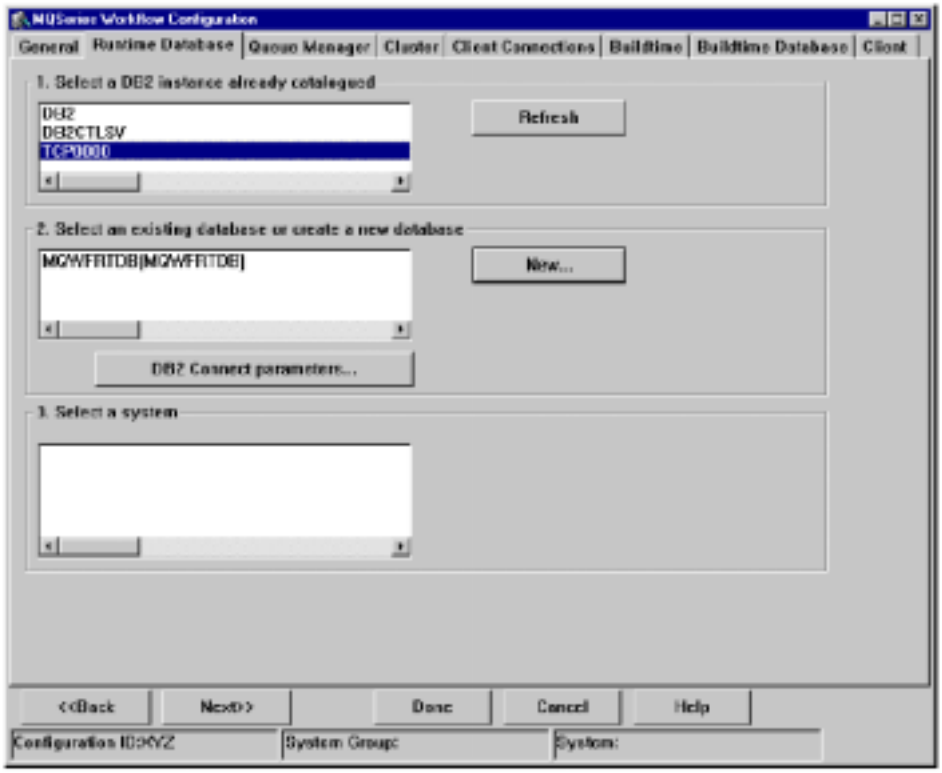

图3-30 选择预定义数据库

现在预定义系统列表显示在图 3-31 中，选择 **BUYXYZ01** 并单击**下一步（Next）**。

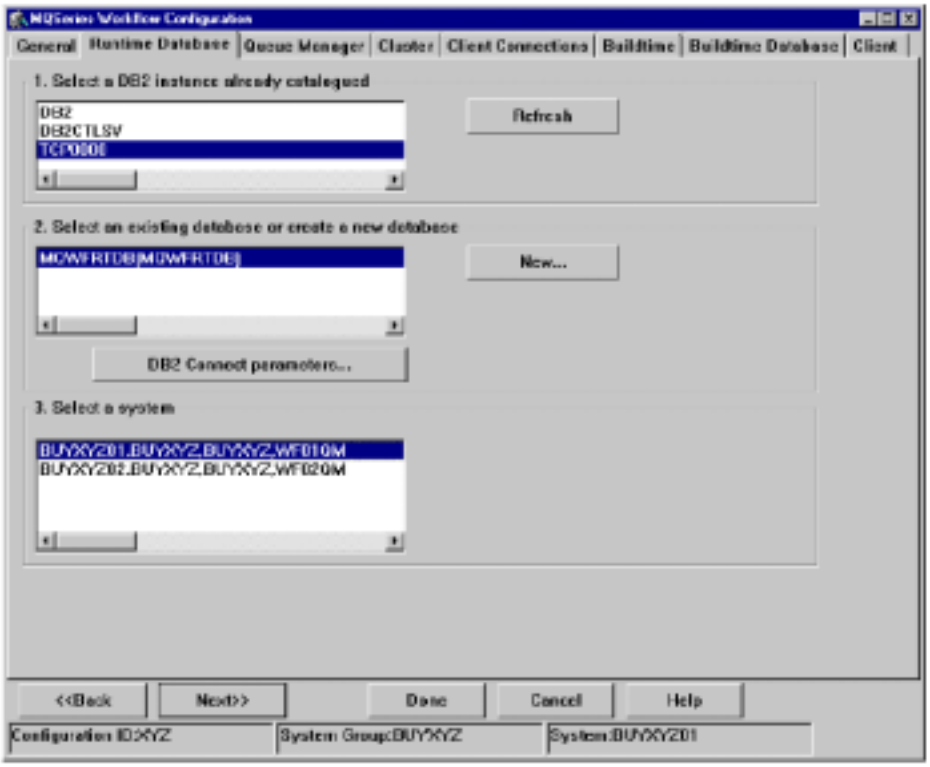

图3-31 选择预定义 MQSeries Workflow 配置

现在我们必须确认 MQSeries 使用的 TPC/IP 主机名和端口号（图 3-32），然后单击**下一步（Next）**。

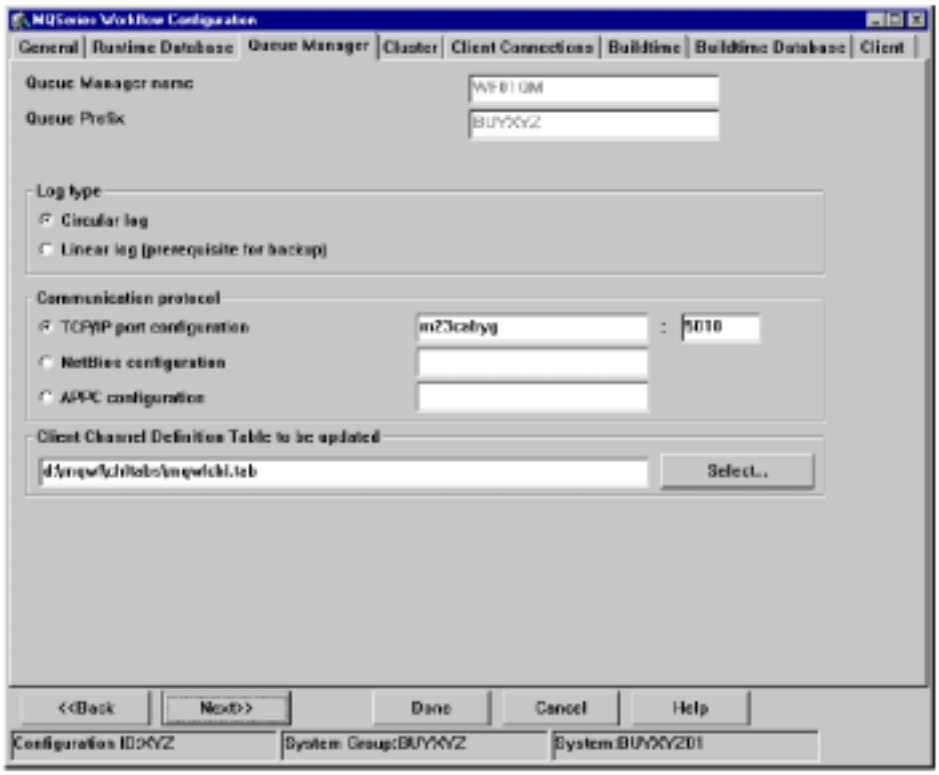

图 3-32 MQSeries 配置

在图 3-33 中，输入群集名称 BUYXYZ 并单击**群集中的第一队列管理器（the first Queue Manager in the Cluster）**。请记住我们还没有在数据库服务器上创建队列管理器，而只是给出了应在工作流服务器上定义的队列管理器的名称而已。单击**下一步（Next）**。

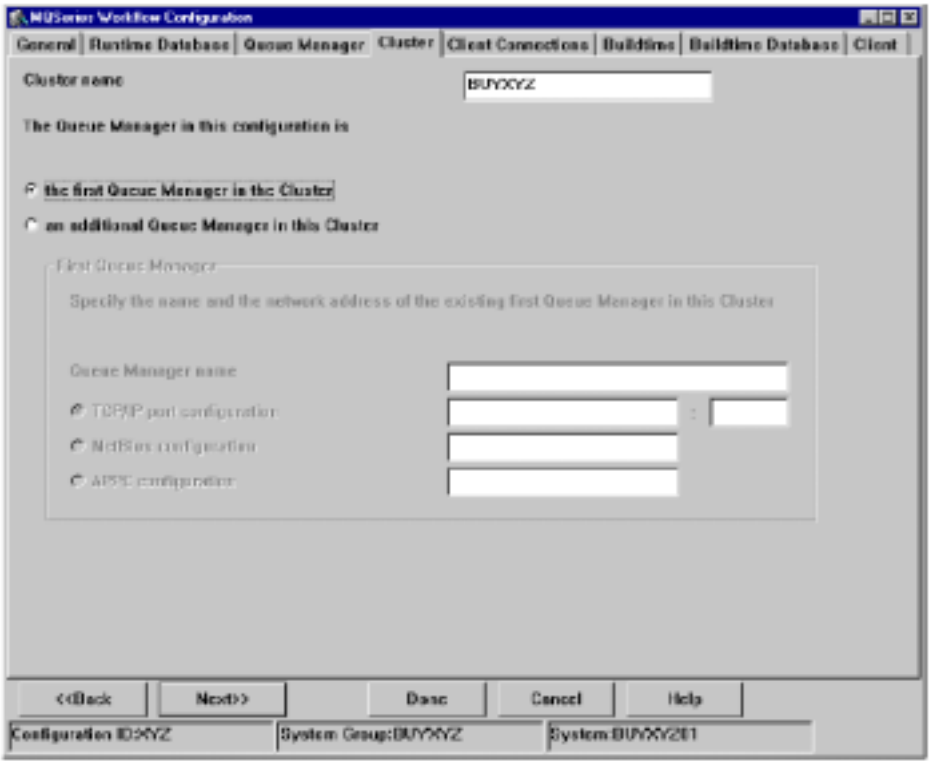

图 3-33 MQSeries 群集设置

在图 3-34 中，确认 MQSeries 客户机通道表的目录名并单击**下一步（Next）**。我们必须把该通道表文件分发给 MQSeries Workflow 客户机。因为我们将使用其自身的队列管理器 MQSeries Workflow 的 Web 客户机功能组件，所以并不真正需要此文件。

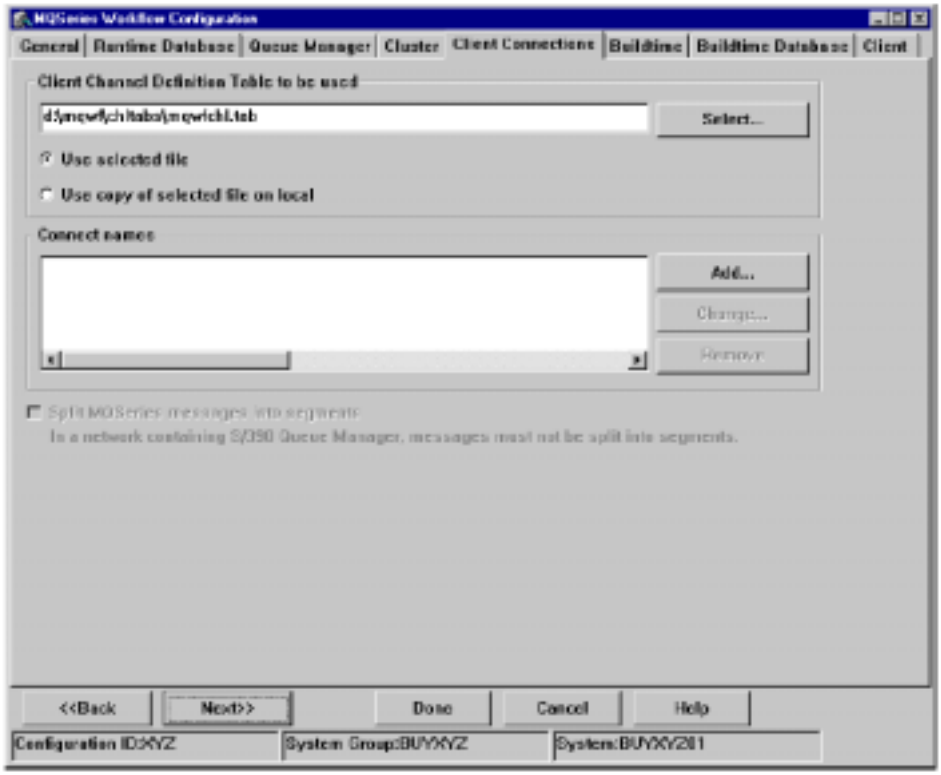

图3-34 MQSeries Workflow 客户机连接设置

通常，工作流服务器并不同时支持构建时数据库。您应在单独的数据库服务器上创建供工作流开发人员使用的构建时数据库。

确定构建时图标目录和数据库类型，然后单击**下一步（Next）**。

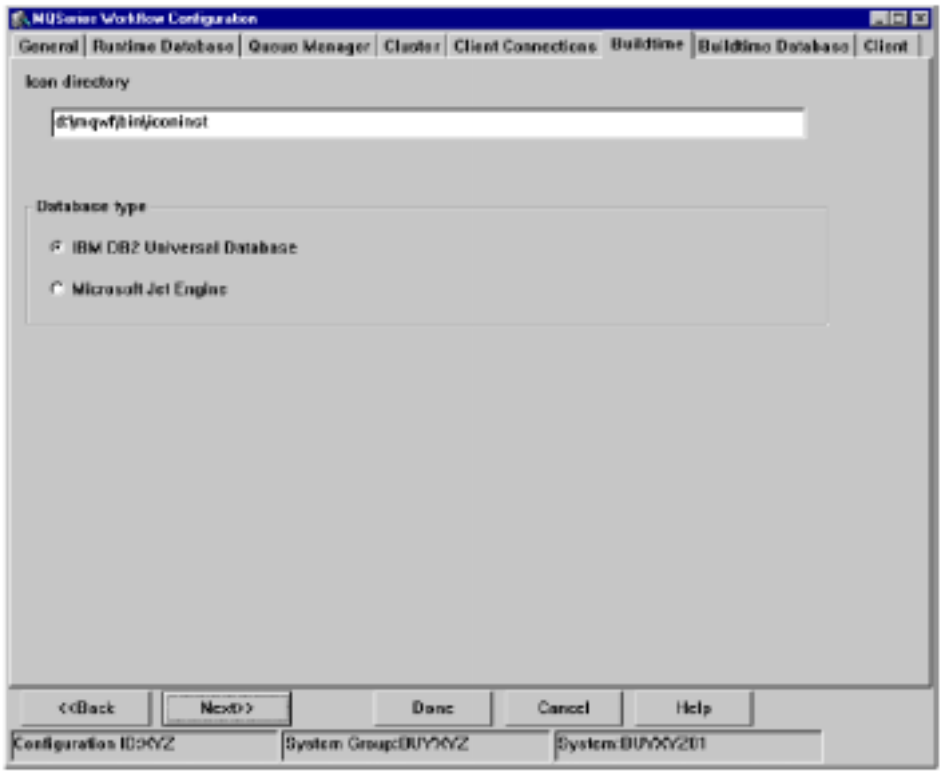

图 3-35 选择构建时数据库系统

选择 **DB2 实例**（如图 3-36 所示），然后单击**新建（New）**。

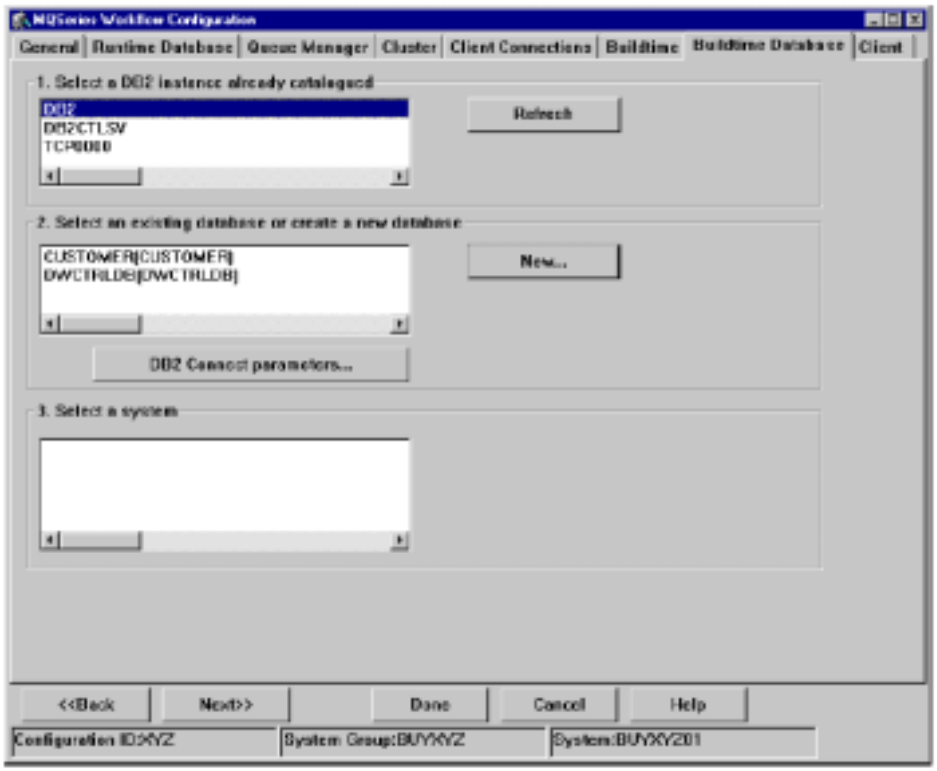

图 3-36 选择 DB2 实例

如图 3-37 所示，为构建时数据库指定名称并为 DB2 容器和 DB2 日志文件指定目录名。

单击**确定（OK）**后，将显示窗口要求输入 DB2 用户 ID 和密码。输入该信息单击**确定（OK）**继续。

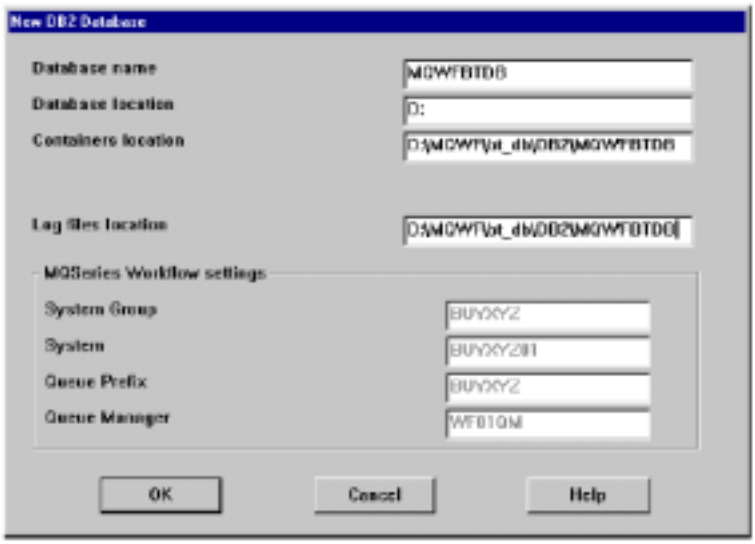

图3-37 数据库设置

最后，在图 3-38 中显示的窗口中确定构建时图标目录，单击**完成（Done）**。

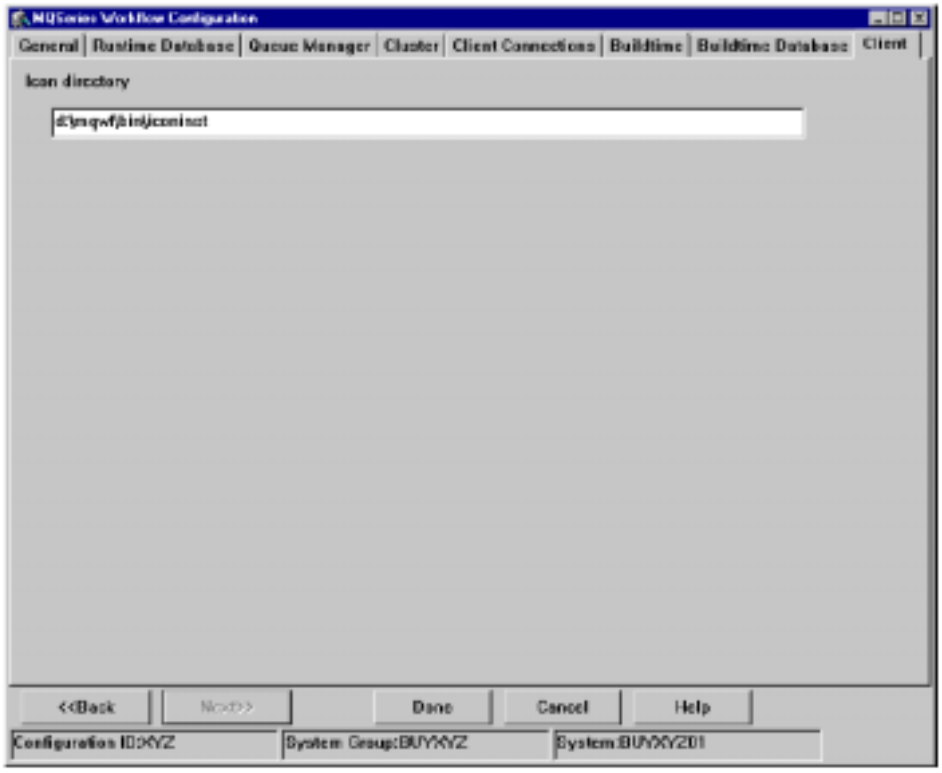

图 3-38 MQSeries Workflow 客户机的图标目录设置

如图 3-39 所示，带有进程条的新窗口表明构建时数据库和新队列管理器，以及其正在创建的对象。请记住现在还没有创建正常正常正常运行时数据库，因为我们将使用现有正常正常正常运行时数据库。

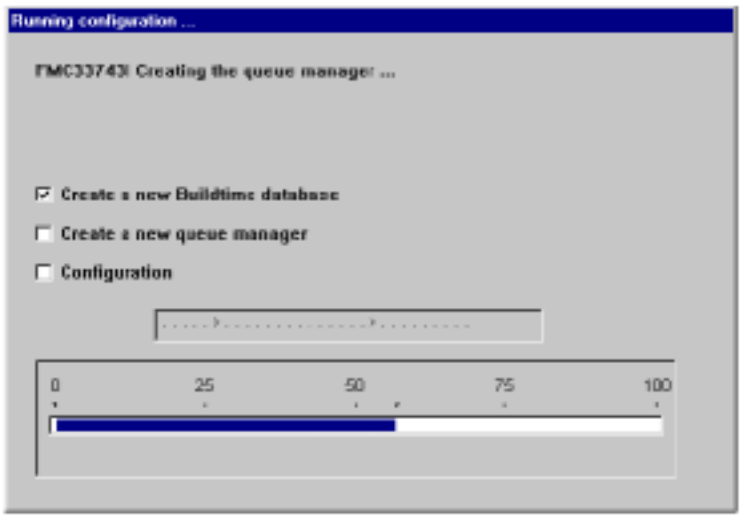

图 3-39 数据库创建进程

最后，通过选择 **Start -> Settings -> Control Panel** 启动 Services JAVA 小程序，确认其中含有 MQSeries Workflow 3.3 - XYZ 新条目。单击**开始（Start）**。

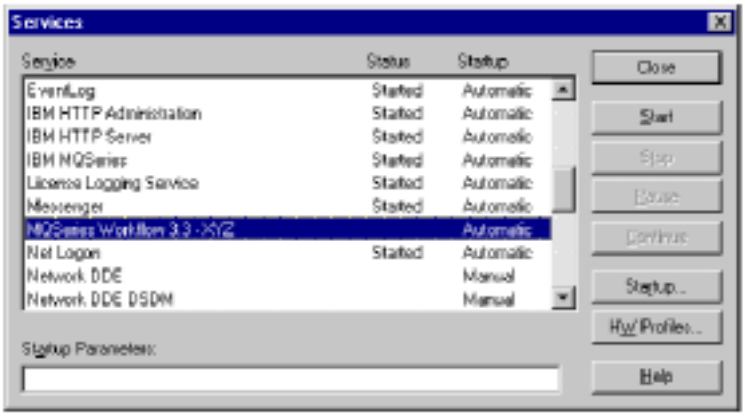

图 3-40 启动 MQSeries Workflow 服务

注意程序文件夹 IBM MQSeries Workflow 中现在出现了另外一些条目：

- MQSeries Workflow 客户机 – BUYXYZ
- MQSeries Workflow 构建时 - BUYXYZ

也要注意，我们在程序文件夹中指出系统组名称，而在 Services JAVA 小程序中进行配置（见图 3-40）。

通过选择 **Start -> Programs -> IBM MQSeries Workflow ->MQSeries Workflow Client – BUYXYZ** 启动 MQSeries Workflow 正常正常正常运行时客户机以验证工作流服务器运行是否正常。

一旦连接客户机，您就会看到主 MQSeries Workflow 客户机（如图 3-41 所示）。使用 ADMIN 用户 ID 登录。

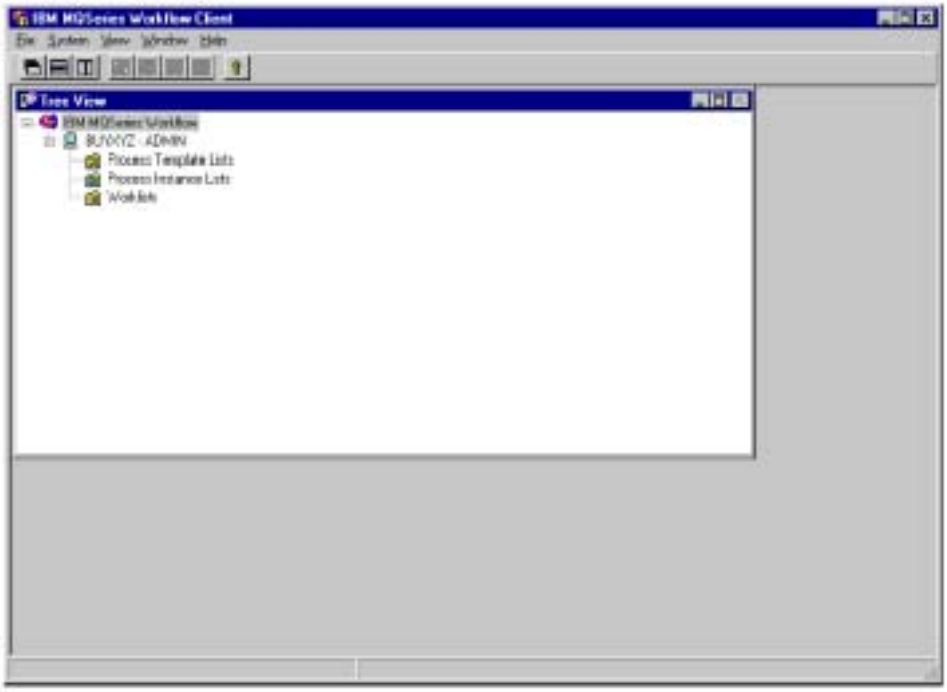

图3-41MQSeries Workflow 正常正常正常运行时客户机

现已完成设备 M23CABYG 上的工作流服务器的初始配置。

现在我们可以在设备 M23CAAAD 上重复这些配置步骤以创建其队列管理器并将其加入 MQSeries 群集。安装 MQSeries Workflow 产品并选择相同组件。同样也要在数据库服务器 M23M1773 上为正常正常正常运行时数据库创建 ODBC 连接。

我们将只显示明显不同于第一个工作流服务器的窗口。

在启动 MQSeries Workflow 配置实用程序之后，通过选择**新建（New）...**来创建新配置。给出配置名 XYZ，然后选择该新创建配置 **XYZ** 作为默认配置。

选择组件**服务器（Server）**、**客户机（Client）**、**正常正常正常运行时数据库实用程序（Runtime Database Utilities）**并选择**下一步（Next）**（见图 3-42）

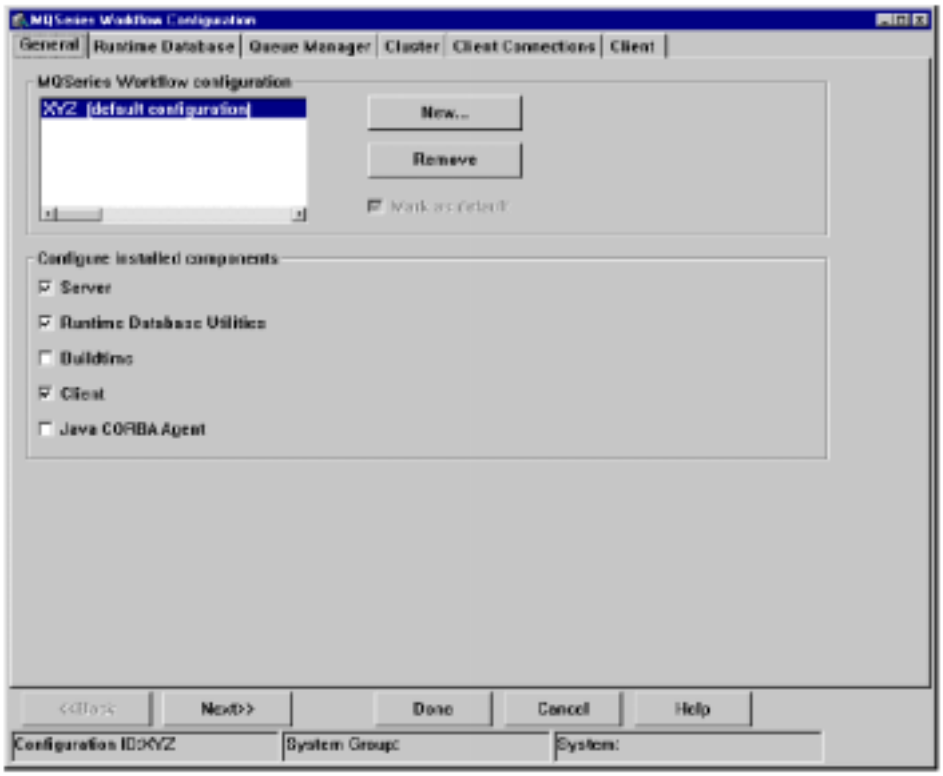

图 3-42 为第二个服务器配置 MQSeries Workflow

在正常正常正常运行时数据库标签上（如图 3-43 所示）选择 DB2 实例 **TCP0000** 和数据库 **MQWFRTDB**。然后，您需要从预定义系统列表中选择 **BUYXYZ02** 条目。

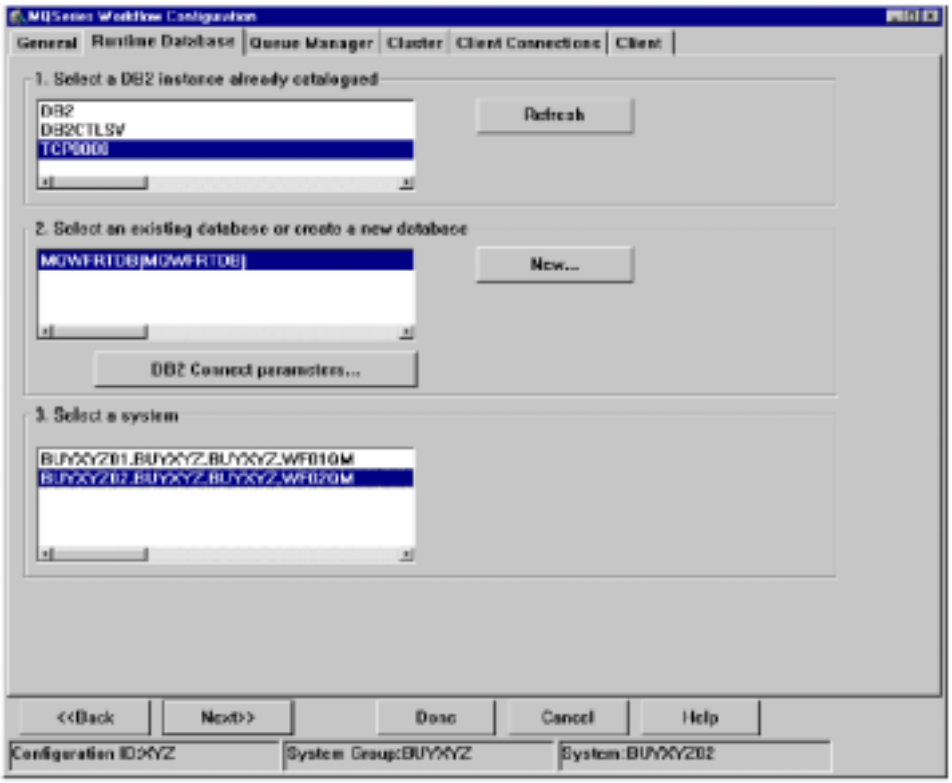

图3-43 选择预定义配置

在队列管理器标签中（如图 3-44 所示），输入由 MQSeries 使用的主机名 M23CAAAD 和端口号 5010。

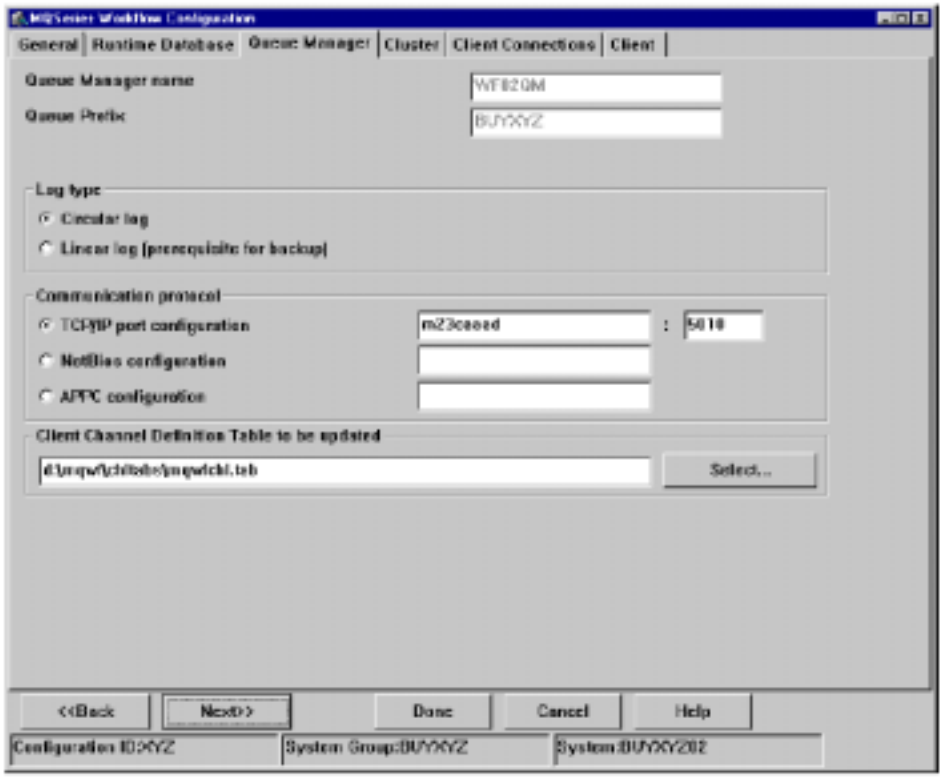

图 3-44 MQSeries 设置

在群集标签中（如图 3-45 所示），检查 WF02QM 队列管理器是否是群集中的其他队列管理器。在群集 BUYXYZ 中输入第一个队列管理器 WF01QM，以及其主机名和端口号 5010。

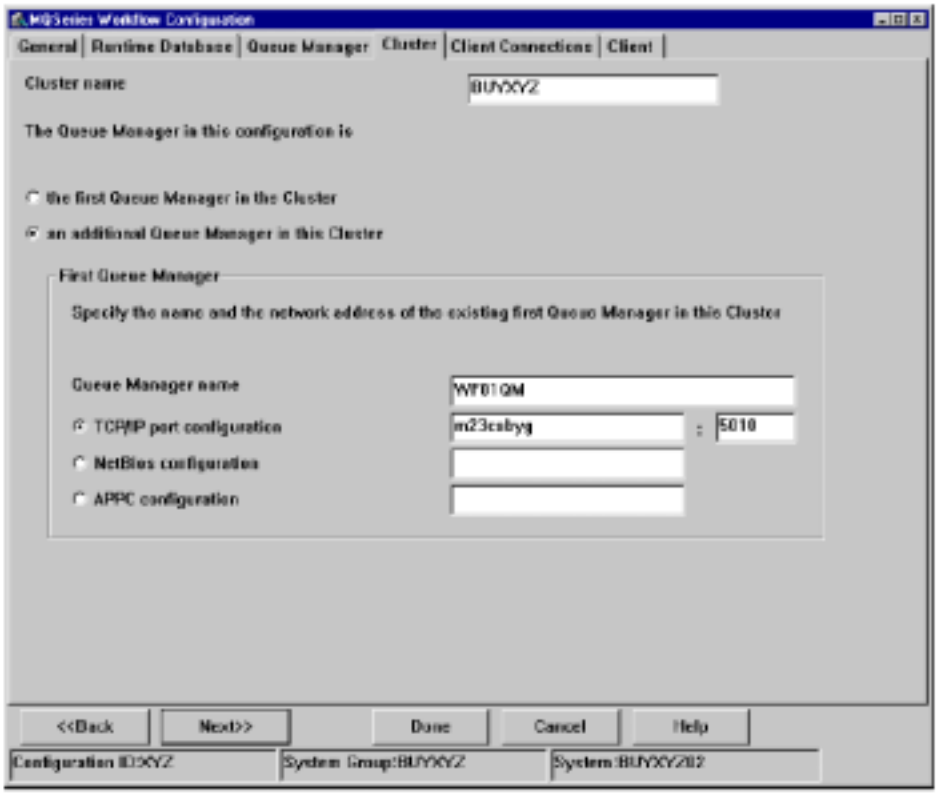

图3-45 加入现有群集

如图 3-46 所示的配置窗口表明正在存储配置信息和创建队列管理器对象。

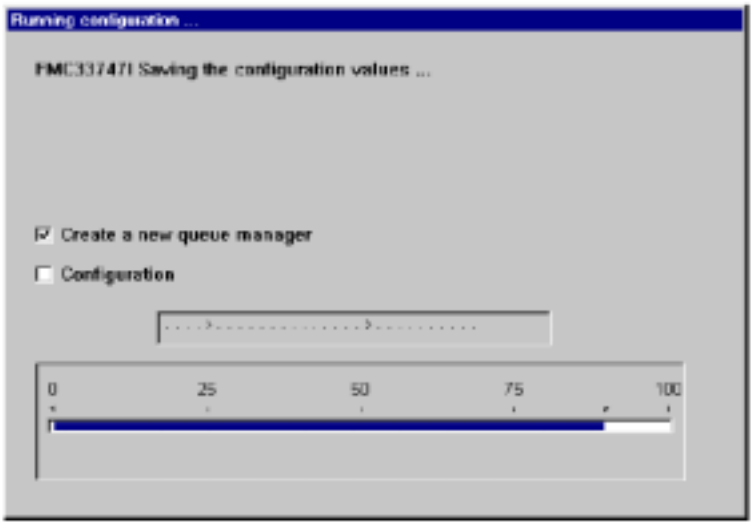

图3-46 队列管理器创建进程

重要事项：在前面的配置步骤中，我们已经创建了服务器连接通道。但是，可能在每台设备上错误命名了该通道。如果出现该情况，将需要手动解决这一问题。打开 MQSeries Explorer 并选定称为 SYSTEM.ADMIN.SRVCONN 的通道。如果该通道存在，删除它并用具有正确名称 SYSTEM.ADMIN.SVRCONN 的通道替换它。注意最后限定词中字母 V 和 R 的顺序。

该问题将使我们不能使用远程服务器上的 MQSeries Explorer 来管理 MQSeries Workflow 创建队列管理器。

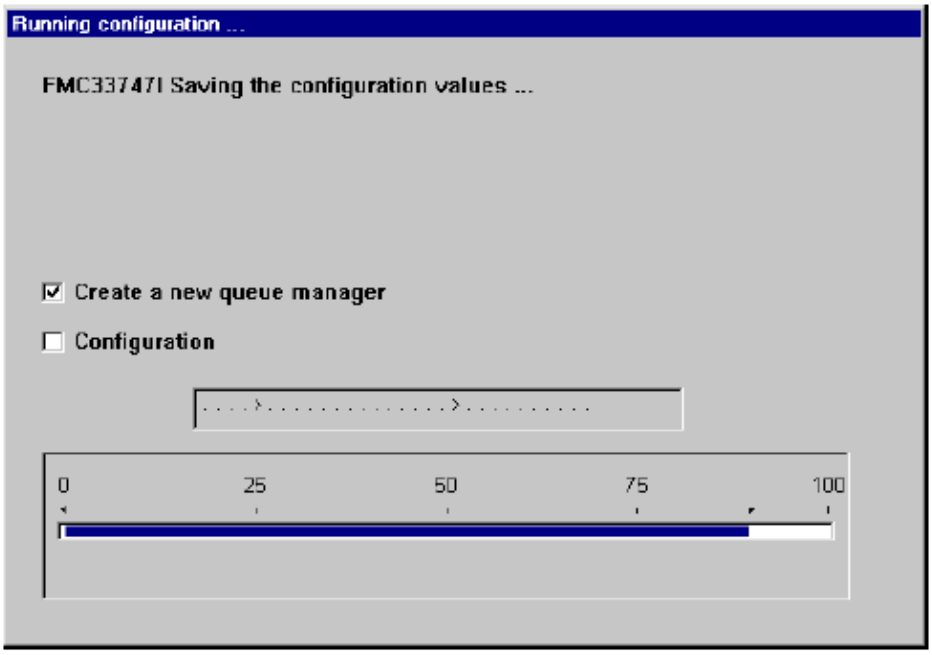

图3-47 确认 MQSeries 配置

在此，您的 MQSeries 配置明显是不正确的。除非连接共享正常正常正常运行时数据库，否则每个工作流服务器仍将独立工作。为了确保 MQSeries 群集正常工作，启动 MQSeries Explorer。确认群集发送和接收通道正在运行，也要确认您能够通过打开左边窗格中的 BUYXYZ 文件夹来访问群集中的其它队列管理器。打开每个队列管理器的群集队列管理器文件夹时，确认您在该文件夹中可以看到两个队列管理器条目。如果您接收到安全错误消息，检查本地用户组 mqm 是否具有您的用户 ID 条目。

最后，启动两个服务器上的 MQSeries Workflow 正常正常正常运行时客户机以检验活动和数据库的连通性。

现在，我们已经完成了工作流服务器设备的安装和配置。在下一节中，我们将通过在同一群集中使用 MQSeries Workflow Web 客户机和 MQSeries Integrator 服务器来扩展该配置。因此，在此处确认该群集配置是非常重要的。因为修改群集配置错误可能会非常困难。

3.4 MQSeries WorkflowWeb客户机环境

MQSeries WorkflowWeb 客户机功能是 JAVA 小程序和其它 Web 资源的集合，该 Web 资源使用 MQSeries WorkflowJava API 与工作流服务器通信。此功能将基于浏览器的接口作为工作流的正常正常正常运行时客户机。代替发布 MQSeries Workflow 客户机代码和配置，安装和配置工作将主要集中在一台单独的 Web 服务器或者可能一组 Web 服务器上。

MQSeries WorkflowWeb 客户机功能有多种可能性来连接工作流服务器。它可以使用 MQSeries 客户机通信方式与工作流服务器通信，也可以使用 Web 服务器上的本地队列管理器或者使用 RMI 连接 MQSeries 设备与工作流服务器通信。在本案例中，Web 服务器上没有安装 MQSeries 软件。当使用 Web 服务器上的本地队列管理器时，该队列管理器将作为客户机集中器运行。第一种情况，对于由 JAVA 小程序创建的每个 MQSeries Workflow 连接来说，您都将在工作流服务器上看到 MQSeries 客户机通道实例。在第二种配置类型中，您只有一对 MQSeries 服务器到服务器通道。本质上讲，第二个选项从工作流服务器中消除了一些负载。有时由于考虑到安全原因我们也会选择第三个选项。

使用客户机集中器更有趣的结果是实现了 MQSeries 群集技术固有的负载均衡和故障恢复功能。这些优点在使用 MQSeries 客户机连接时是不存在的。

现在我们将在设备 M23BZZYP 上应用 MQSeries WorkflowWeb 客户机功能。为此，假设以下产品已经安装：

- MQSeries for Windows NT 版本 5.2
- MQSeries Java 支持用于 SupportPac MA88。您需要确保已安装了最新的程序包，可由以下网址获得该工具包：

  http://www-4.ibm.com/software/ts/mqseries/txppacs/ma88.html
- DB2 for Windows NT V7 +FixPak 1
- Java 开发工具包版本 1.2.2
- WebSphere 高级版版本 3.5.3

重要事项：必须根据您的 Windows NT 版本来更新系统上的 Windows Installer。欲获得最新版本，请从以下 Microsoft 网站下载：
http://www.microsoft.com/downloads/release.asp?ReleaseID=17344

安装 MQSeries Java support 需要该新 Windows Installer。

您可以通过在 DOS 窗口中输入以下命令来确认设备上的 Java 版本：
java –version

如果该窗口显示的不是 Java 版本 1.2.2，您必须立刻更新 Java JDK 或 JRE。

通过两个步骤创建 MQSeries Workflow 配置。首先，我们将通过客户机集中器队列管理器安装标准的 MQSeries Workflow 正常正常正常运行时客户机。其次，我们将为 Web 客户机添加另一个配置 ID。从理论上讲，并不需要第一步。然而，在调试配置或正常正常正常运行时问题时，使用标准 MQSeries Workflow 正常正常正常运行时客户机可能是有帮助的，因为它可以帮助我们查出问题。如果 GUI 客户机运行正常，而 Web 客户机却不能工作，您应知道必须在 Websphere 环境中查找问题。如果 GUI 客户机与工作流服务器连接出现故障，那么您也应知道消除 Websphere 环境中的所有调试。

在开始所有配置之前，我们必须安装 MQSeries Workflow 必需的组件。我们已经安装了：

- 客户机
- Java CORBA 代理

在 MQSeries Workflow 安装窗口中有一些需要选择的选项。参见第 45 页图 3-29 所示的该窗口的实例。

首先，我们需要创建含有称为 WASQM 队列管理器的配置 XYZ，并且使其成为现有 MQSeries 群集 BUYXYZ 的一部分。然后，通过选择 **Start -> Programs -> IBM MQSeries Workflow-> MQSeries Workflow 配置实用程序 9MQSeries Workflow Configuration Utility**）来启动该配置实用程序。

除了下面所示，此处大多数步骤都与先前描述的设备 M23CABYG 和 M23CAAAD 的配置步骤相同。

在配置实用程序打开的窗口中（图 3-48），选择**新建**（**New**）开始在该设备上创建的第一个配置。

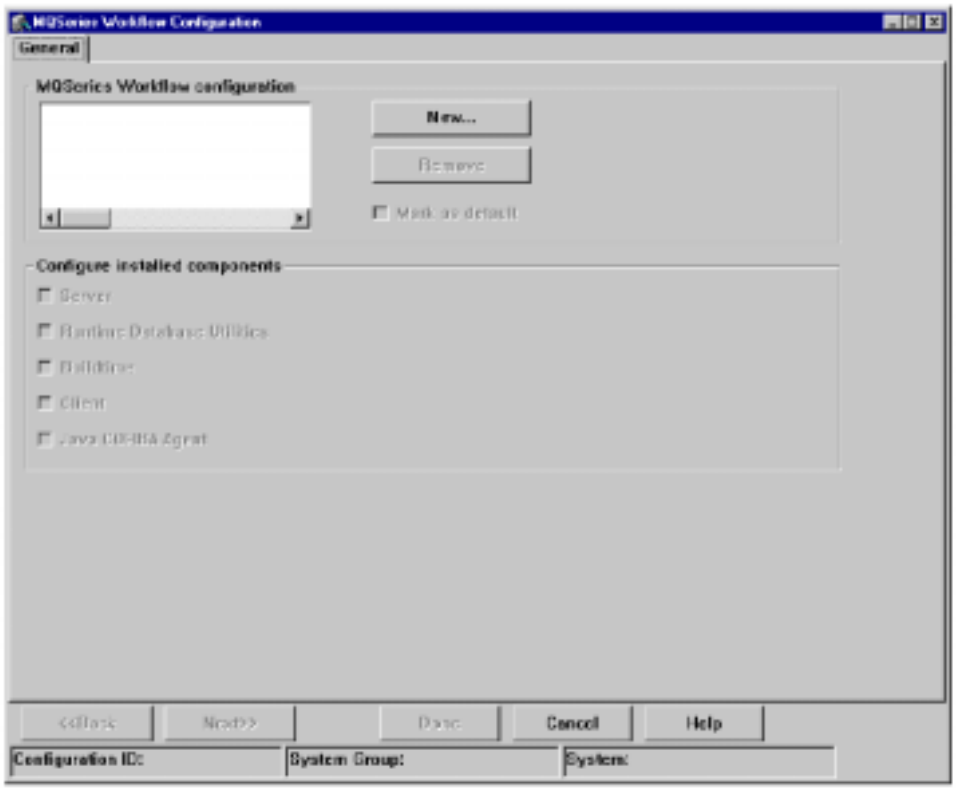

图 3-48 添加新的 MQSeries Workflow 配置

输入 XYZ 作为配置 ID 并单击**下一步（Next）**继续。

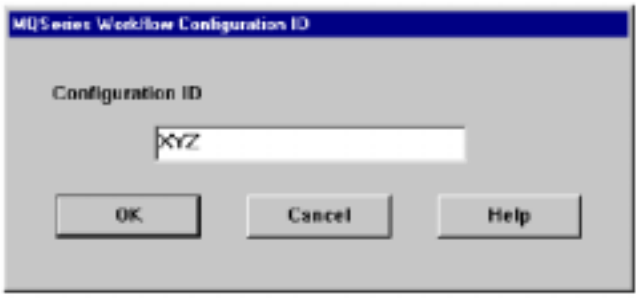

图 3-49 给出配置 ID

返回主窗口选择默认配置 **XYZ**。选择**客户机（Client）**复选框并单击**下一步（Next）**继续。

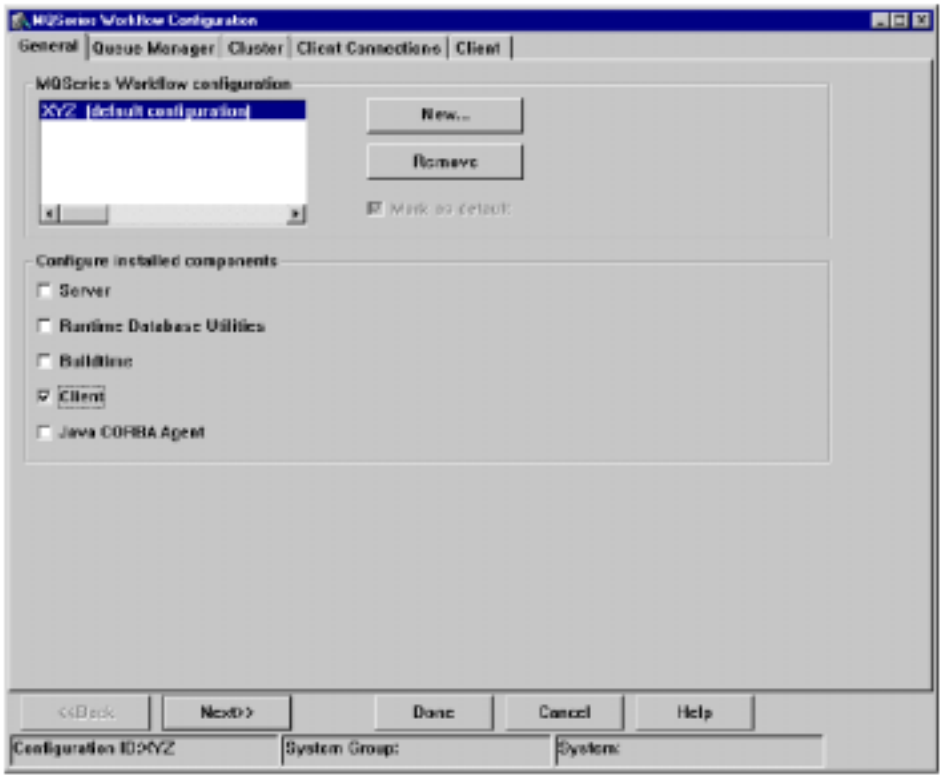

图3-50 选择配置组件

输入队列管理器名称 WASQM 和队列前缀 BUYXYZ。

选择**循环日志（Circular log）**。

选择 **TCP/IP 端口配置（TCP/IP port configuration）**并给出主机名 M23BZZYP 和端口号 5010。

单击**下一步（Next）**继续。

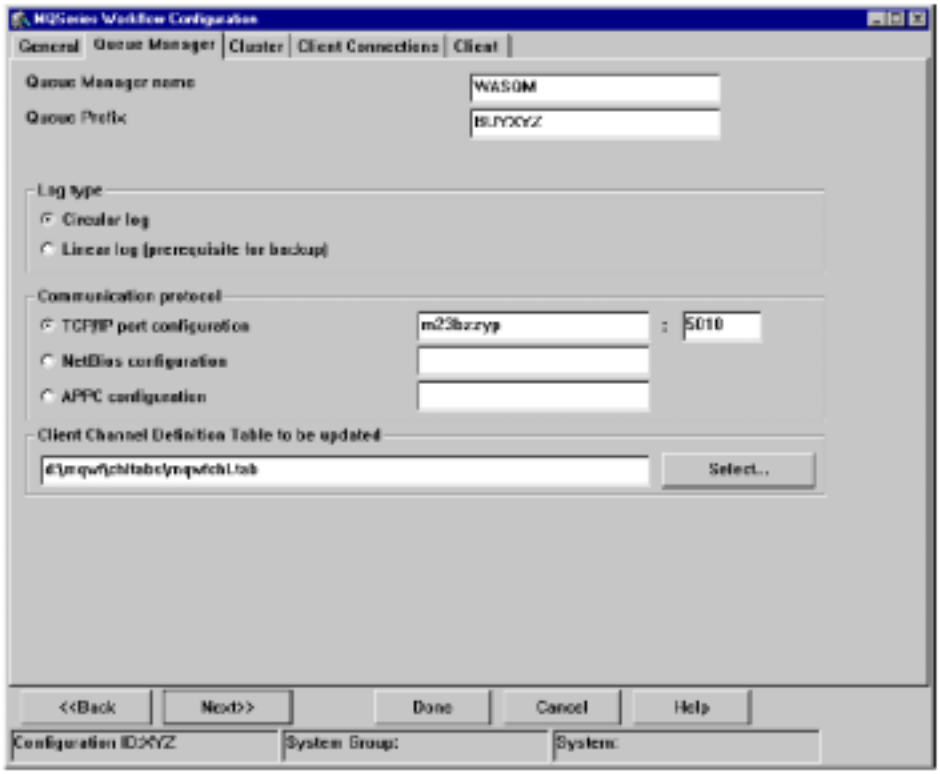

图3-51MQSeries 配置

输入群集名称 BUYXYZ 并确认它是群集中另外的队列管理器。

给出第一个队列管理器的名称 WF01QM，以及其主机名称和端口。

单击**下一步（Next）**继续，请见图 3-52。

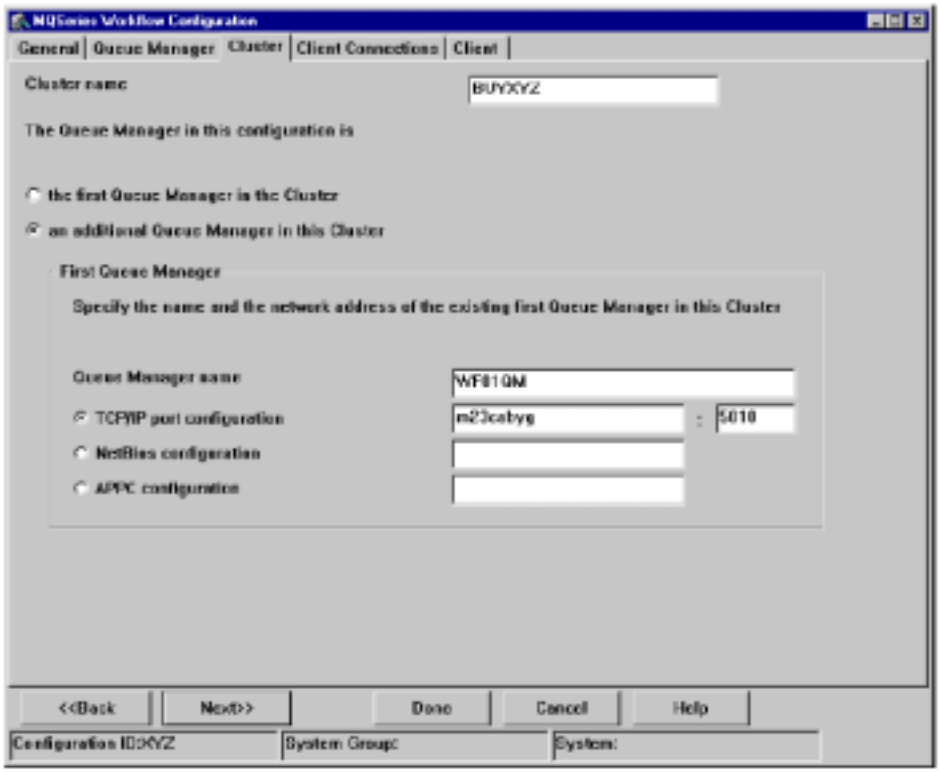

图3-52 群集设置

提供 MQSeries Workflow 客户机连接细节，单击**添加（Add）**。

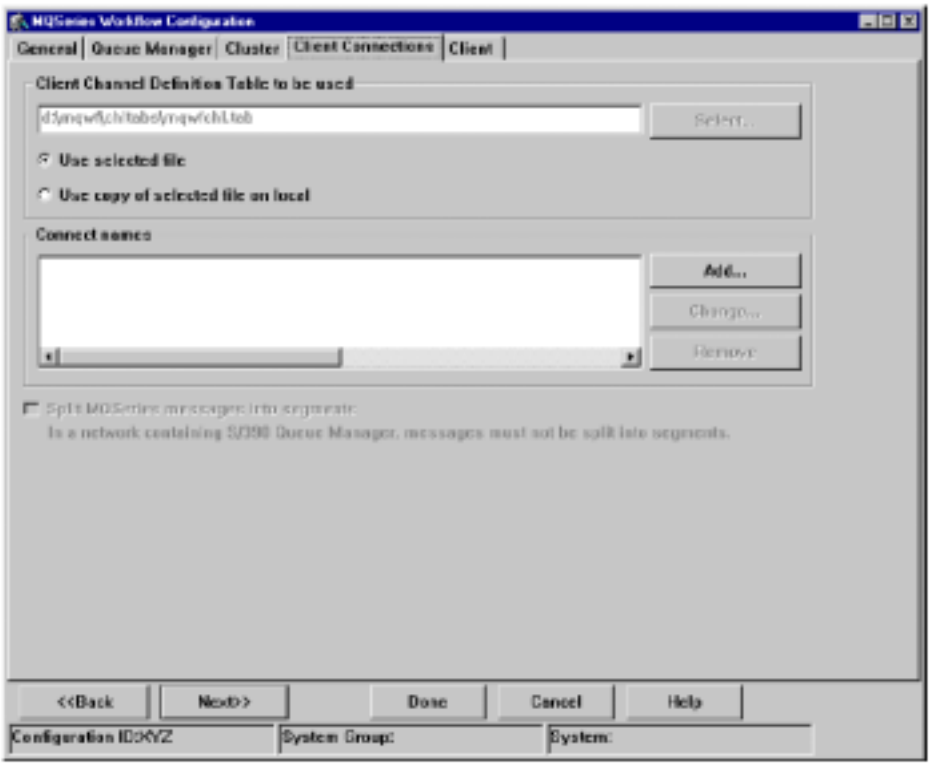

图3-53 MQSeries Workflow 客户机连接设置

确认队列管理器名称（如图 3-54 所示）并单击**添加（Add）**。

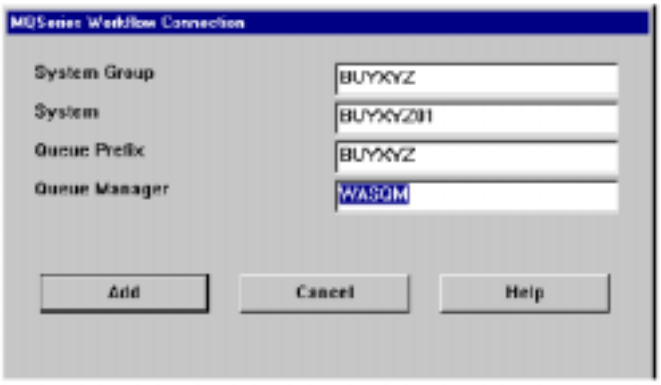

图3-54 MQSeries Workflow 客户机连接设置

在图 3-55 中，单击出现在**连接名**（Connect names）字段中的新条目并单击**下一步（Next）**继续。

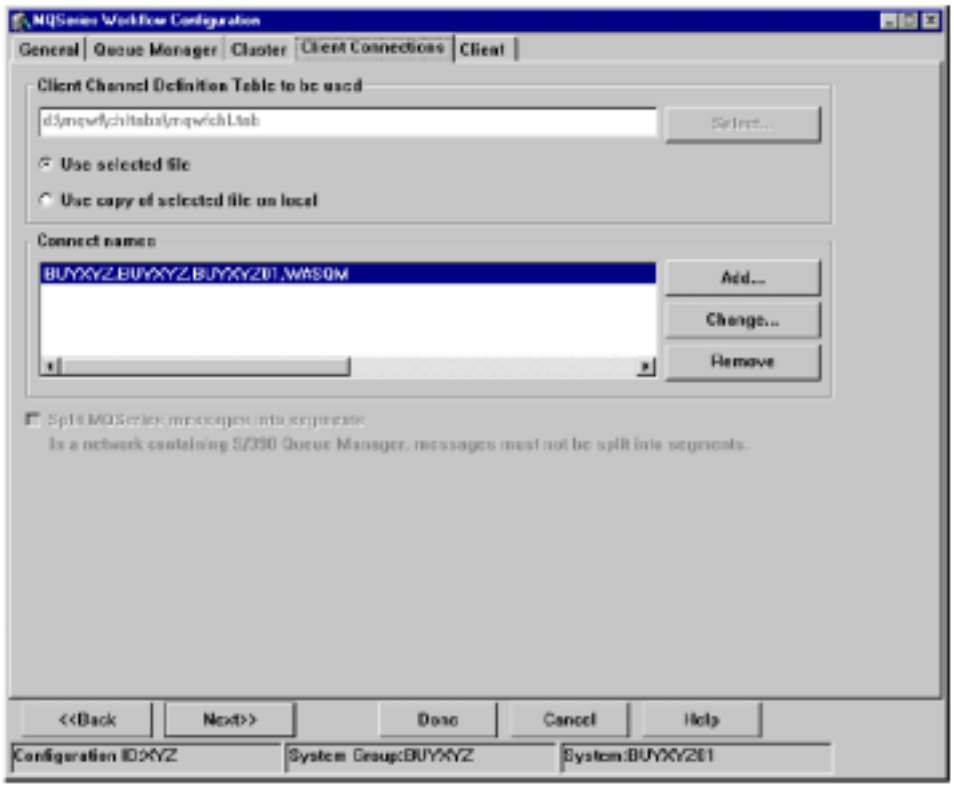

图 3-55 已完成的客户机设置

在图 3-56 中，确认图标目录，然后单击**下一步（Next）**继续。

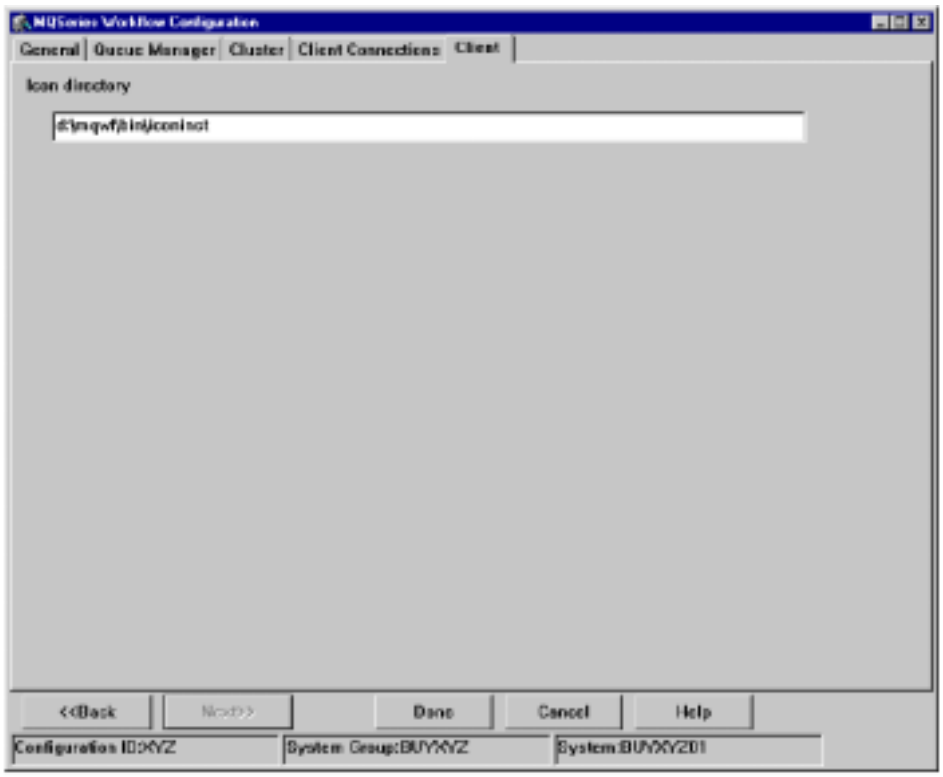

图 3-56 MQSeries Workflow 客户机图标目录设置

如图 3-57 所示，当配置发生时将显示带有进程条的窗口。

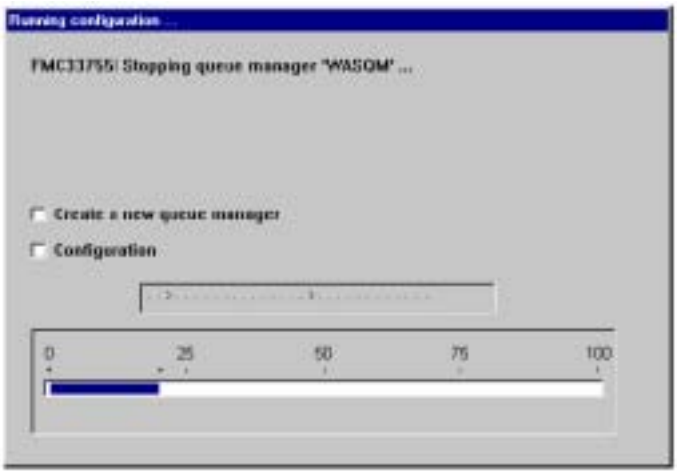

图 3-57 正在创建的配置

最后，如图 3-58 所示的确认窗口表明安装成功。

图 3-58 配置完成

现在我们已完成了基本 MQSeries Workflow 客户机的安装。下一步我们将运行基于命令行的实用程序 FMCZUTIL 来创建配置 XYZ1，以便使用现有队列管理器 WASQM 来配置 Web 客户机。Web 客户机是不能用图形化配置实用程序来配置的。

以下脚本显示了命令流程：

```
C:\>FMCZUTIL
FMC33201I Configuration Commands Menu (FMC33201I 配置命令菜单) :
l ...List
s ...Select
```

```
c ...Create
x ...Exit Configuration Commands Menu (退出配置命令菜单)
C
Configuration identifier  (配置标识符) :[FMC ]XYZ1
FMC33210I Select Category Menu (FMC33210I 选择目录菜单) :
s ...( )Server
i ...( )Runtime Database Utilities (正常正常正常运行时数据库实用程序)
b ...( )Buildtime (构建时)
c ...( )Client with queue manager (队列管理器客户机)
j ...( )Java Agent
w ...( )Web Client
a ...all
n ...none
x ...Exit Select Category Menu (退出选择目录菜单)
JW
FMC33210I Select Category Menu (FMC33210I 选择目录菜单) :w
FMC33210I Select Category Menu (FMC33210I 选择目录菜单) :
s ...( )Server
i ...( )Runtime Database Utilities (正常正常正常运行时数据库实用程序)
b ...( )Buildtime
c ...(A)Client with queue manager (队列管理器客户机)
j ...(X)Java Agent
w ...(X)Web Client
a ...all
n ...none
x ...Exit Select Category Menu (退出选择目录菜单)
X
- Configuration of queue manager (配置队列管理器)  ...
System group name (系统组名)  :[FMCGRP ]BUYXYZ
System name (系统名)  :[FMCSYS ]BUYXYZ01
Queue manager name (队列管理器名)  :[WASQM ]
Queue prefix (队列前缀)  :[BUYXYZ ]BUYXYZ

- Configuration of client (配置客户机)  ...
Channel definition table file  (通道定义制表文件) :[d:\mqwf \chltabs \mqwfchl.tab ]
- Configuration of Java Agent (配置 Java 代理)  ...
- FMC33749I Selected Locator Policy (FMC33749I 选择定位器策略)  :Local bindings

FMC33606I Specify information about garbage collection (reaper) (有关垃圾收集的 FMC33606I 详
细信息) (收割者) ...:
Agent cycle (in seconds):[300 ]
Client threshold (number of objects):[1000 ]
Client cycle (in %of agent cycle):[90 ]
- Configuration of Web Client (配置 Web 客户机)  ...
```

```
FMC33777I Select application server（FMC33777I 选择应用服务器） ...:
w ...(X)WebSphere
o ...( )Other
Code Version :[3300 ]

FMC33607I Specify information about the WebSphere Application Server （有关应用服务器的
FMC33607I 详细信息） ...:
Installation directory（安装目录） :[d:\WebSphere \AppServer ]
TCP/IP address of administration node （管理节点的 TCP/IP 地址）:[m23bzzyp ]
XML configuration skeleton file name（XML 配置框架文件名） :[fmcoh35.skel ]FMCOH352.SKEL
c ...Create configuration profile for 'XYZ1'now（现在创建'XYZ1'文件）
s ...Save input to file
r ...Review/change input
x ...Exit (input for configuration 'XYZ1'will be lost)
C
- FMC33680I The profile for the configuration 'XYZ1'was updated successfully.（成功更新配置
'XYZ1'的 FMC33680I 文件）
- Do you want to configure the Web Client within the WebSphere Application
Server now?（在 WebSphere 应用服务器中您想配置 Web 客户机吗？）
y ...Yes
n ...No
Y
[01.04.06 14:01:52:620 EDT ] 887a4a5a NodeConfig A XMLC0053I:Importing Node
:m23bzzyp
[01.04.06 14:01:52:850 EDT ] 887a4a5a ApplicationSe A XMLC0053I:Importing
ApplicationServer :MQWF Web Client -XYZ1
[01.04.06 14:01:53:792 EDT ] 887a4a5a ServletEngine A XMLC0053I:Importing
ServletEngine :Servlet Container
[01.04.06 14:01:54:363 EDT ] 887a4a5a WebApplicatio A XMLC0053I:Importing
WebApplication :MQWFClient
[01.04.06 14:01:56:055 EDT ] 887a4a5a ServletConfig A XMLC0053I:Importing
Servlet :ErrorReporter
[01.04.06 14:01:57:587 EDT ] 887a4a5a ServletConfig A XMLC0053I:Importing
Servlet :file
[01.04.06 14:01:59:600 EDT ] 887a4a5a ServletConfig A XMLC0053I:Importing
Servlet :jsp11
[01.04.06 14:02:01:042 EDT ] 887a4a5a ServletConfig A XMLC0053I:Importing
Servlet :Main
[01.04.06 14:02:01:102 EDT ] 887a4a5a ServletConfig W XMLC0055W:Updating
Servlet :Main,since it was already created
[01.04.06 14:02:02:484 EDT ] 887a4a5a SessionManage A XMLC0053I:Importing
SessionManager :Session Manager
- FMC33659I The configuration within the WebSphere Application Server was
successful.（在 WebSphere 应用服务器中成功配置了 FMC33659I）
FMC33201I Configuration Commands Menu（FMC33201I 配置命令菜单）：
l ...List
```

```
s ...Select
c ...Create
d ...Change default configuration（更改默认配置）
x ...Exit Configuration Commands Menu（退出配置命令菜单）
X
```

该实用程序已在 WebSphere 中配置了服务件集合，并将移植包含所有所需文件的目录 \MQWF\cfgs\XYZ1。

检验并更新下列项目应。请参考 Web 客户机文档以获取更详细的信息。

- 检查 MQSeries 群集通道配置
  - 群集队列管理器文件夹：您应查看所有三个队列管理器：WASQM、WF01QM、WF02QM(见图 3-59)。

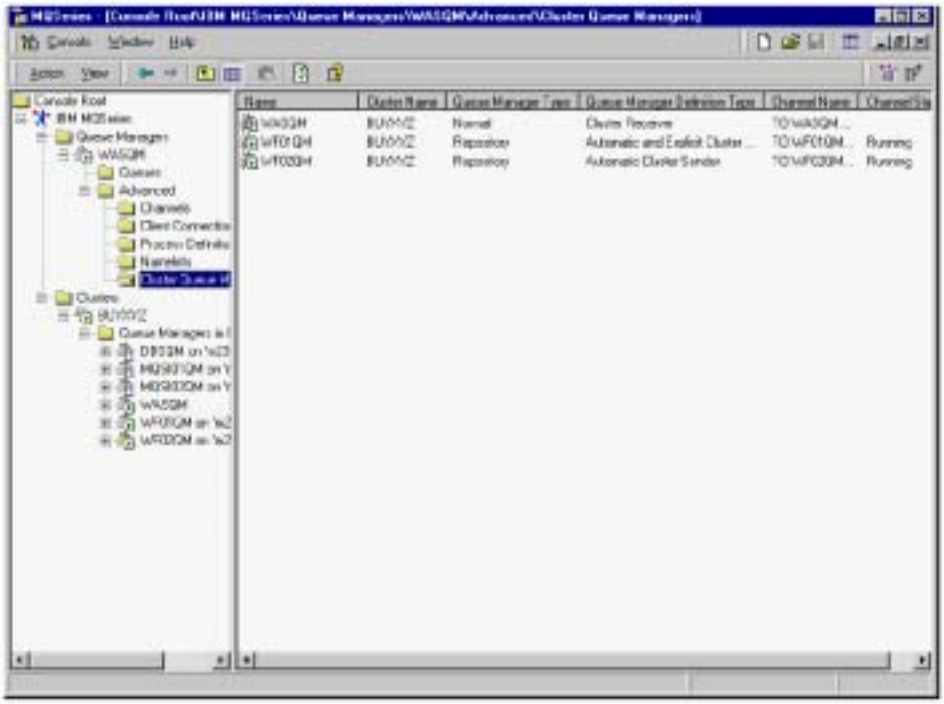

图3-59MQSeries 群集确认

- 在 MQSeries 资源管理器的 WASQM 队列文件夹中， MQSeries Workflow 的两个实例输入队列。在看到这些队列之前，您可能需要刷新群集。

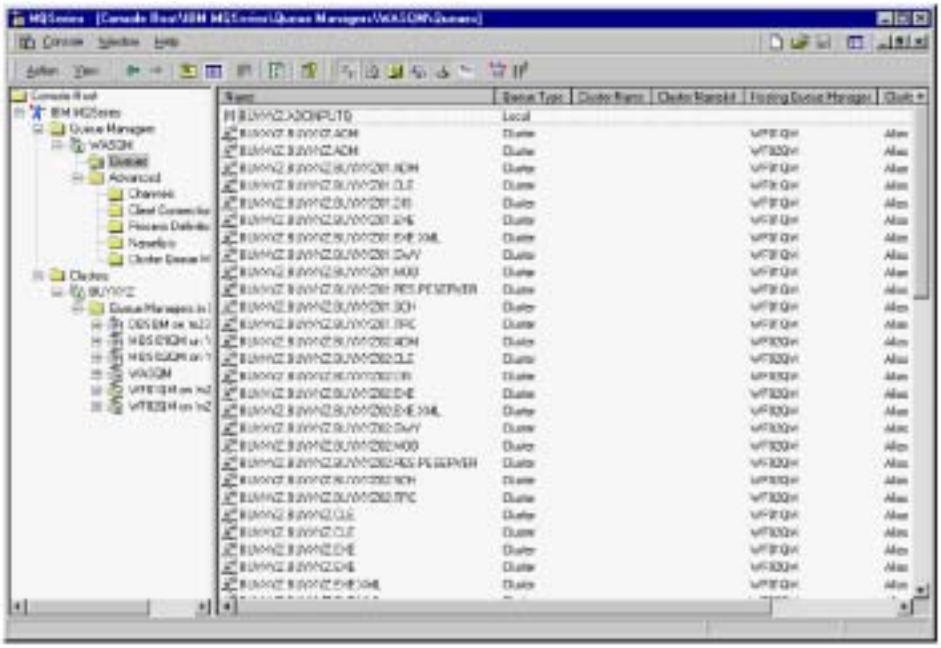

图3-60 Web 服务上的MQSeries Workflow 系统队列的检测

- 检查路径\MQWF\cfgs\XYZ1\WebClient 中的 webclient.properties 文件。您可能需要将以下属性更新或设成有效行（uncomment）：
  - Logfile
  - CommandHandler
  - DefaultViewer
  - StarterUserId

- 更新 logon.html 和 logon.jsp 文件，用您的服务件名称代替@rootURI@，本案例中是\MQWFClient-XYZ1。

3.4.1 在MQSeries Workflow中创建工作列表

现在我们已经有了已配置的 MQSeries Workflow 正常正常正常运行时客户机和 MQSeries WorkflowWeb 客户机。为了检验配置，我们将在 MQSeries Workflow 中创建一些默认列表。这意味着要更新一些数据库。我们将使用 GUI 客户机来创建该列表，然后使用 Web 客户机来验证我们是否可以访问同一对象。

启动 MQSeries Workflow 正常正常正常运行时客户机并以 ADMIN 身份登录。右键单击**进程模板列表(Process Template List)**（如图 3-61 所示）。

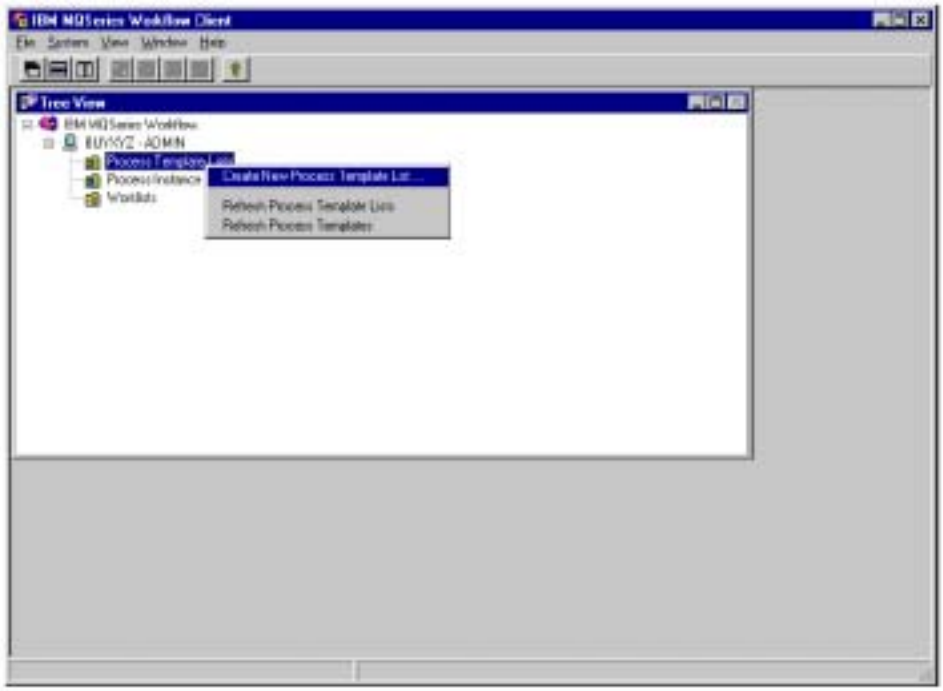

图3-61 创建进程模板列表

给出模板列表名称，然后单击**确定(OK)**。

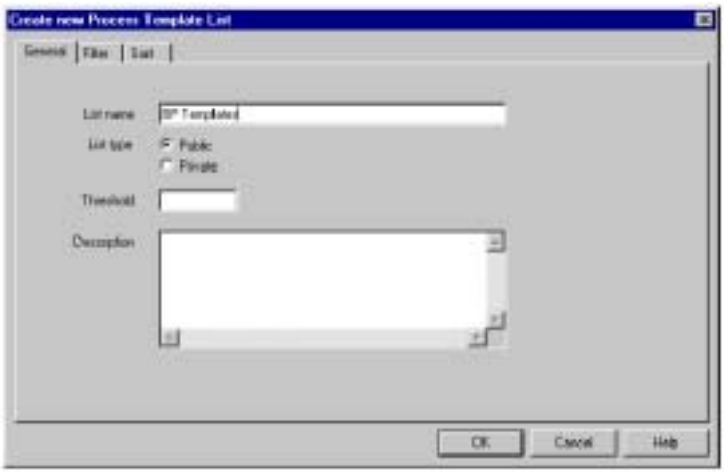

图 3-62 进程模板列表设置

现在右键单击**过程实例列表(Process Instance Lists)**并选择**创建新过程实例列表(Create New Process Instance List)**。

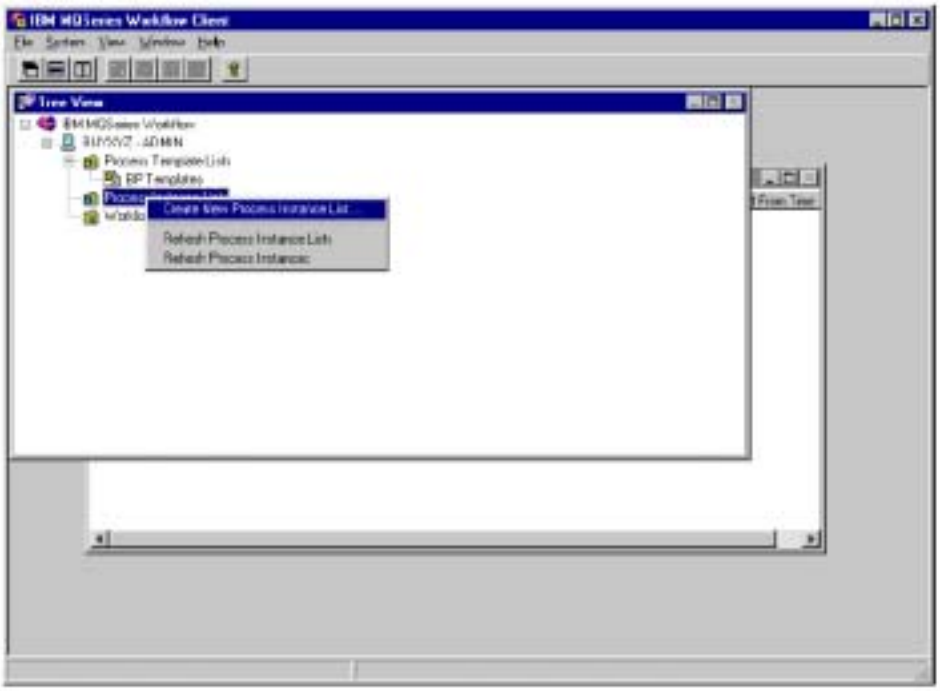

图3-63 创建进程实例列表

给出进程实例列表名称，本案例中为“ 进程实例 ”。

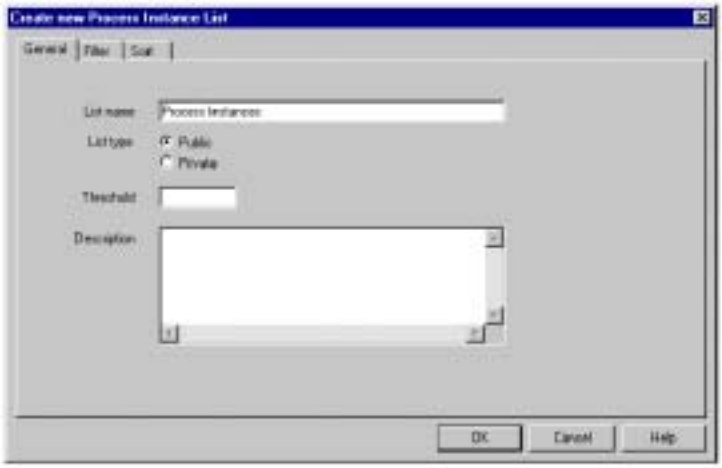

图 3-64 进程实例列表设置

现在右键单击**工作列表(Worklists)**并选择**创建新工作列表(Create New Worklist)**。

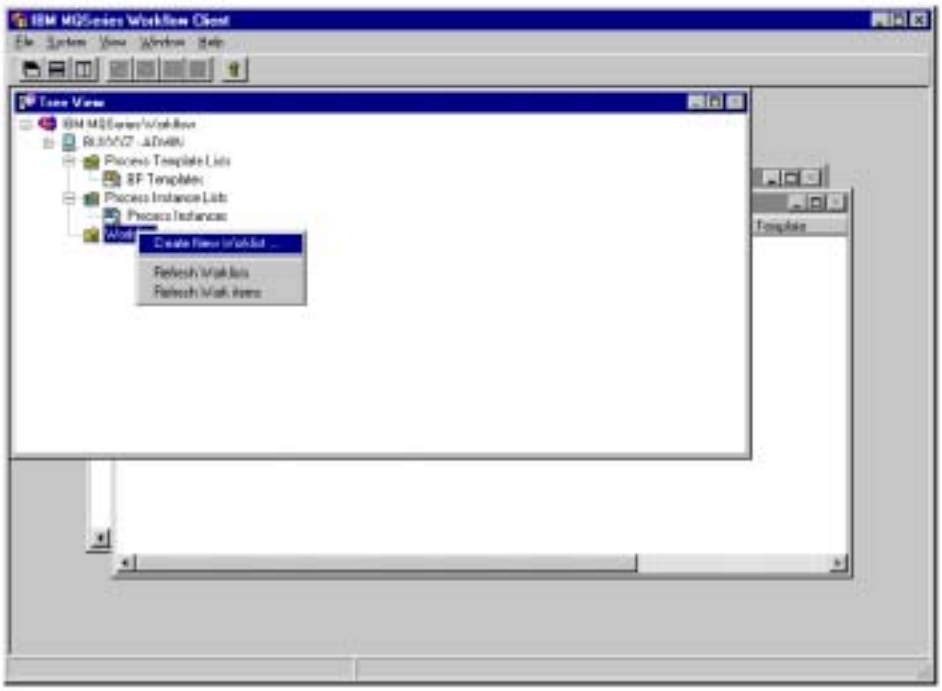

图3-65 创建新工作列表

给出工作列表名称，本案例中为“ 工作项目 ”(Work Items)。

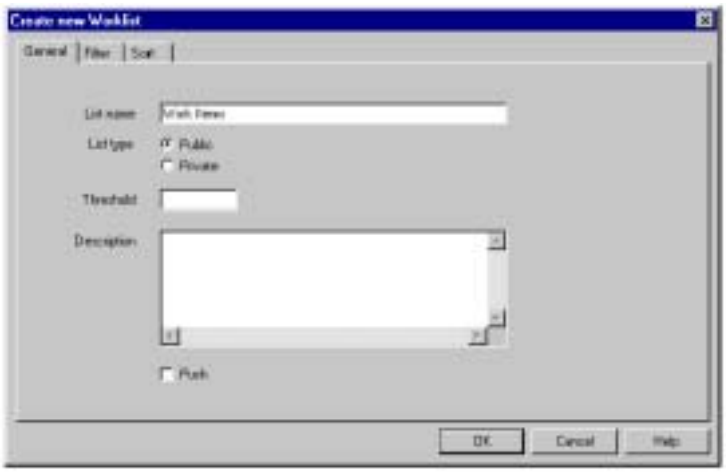

图 3-66 工作列表设置

注意您可以尽可能创建多个列表。对于更复杂的环境，您可能需要包含过滤器以消除数据库中的超时队列。

3.4.2 检验Web客户机

启动您喜爱的浏览器并键入以下 URL 地址：

http://m23bzzyp/MQWFClient-XYZ1/RTC.html

其中，m23bzzyp 是包含 WebSphere 设备的主机名。

*图3-67*Web *客户机登录*

- 输入 ADMIN 作为用户 ID 并且将 password 作为密码。
- 单击**登陆(Logon)**按钮。

图 3-68 Web *客户机工作项目列表*

- 通过从**导航图标**(Navigate)下拉列表中选择它们以检验模板列表和实例列表的运行。

以上总结了 MQSeries Workflow 环境的基本检验。这也是检验 MQSeries 群集技术的负载均衡功能的好机会。这样做最容易的方法是打开 WebSphere 设备上的 MQSeries 资源管理器，然后监控群集队列管理器文件夹中的通道状态。双击该列表中的 WASQM 条目，并在 Web 客户机工作时定时单击**刷新**(Refresh)按钮。在群集接收通道 TO.WASQM.TCP 上，两个实例接收消息的数量将同步增加。

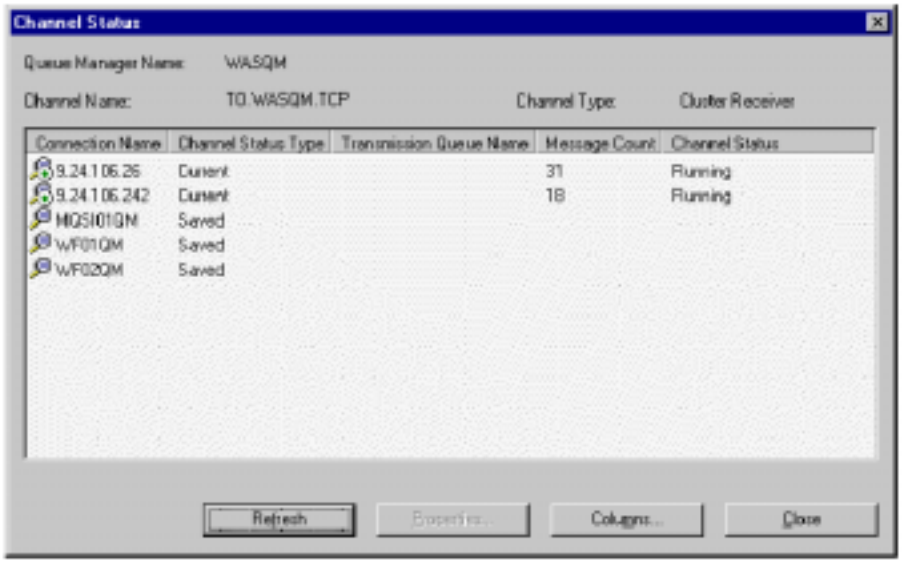

图3-69 监控通道状态

注意在上面的窗口中，第一个通道实例明显比第二个发送的消息要多。这很容易解释。第一个实例将 WASQM 队列管理器连接到完整的数据仓库上，同时用来交换群集信息，所以其发送的消息较多。

3.5 工作流环境的检验

本书第 82 页的 3.4.2 “ 检验 Web 客户机 ” 中所做的检验是非常基本的。它只证明 Web 服务器与工作流服务器能够通信。更全面的检验可以通过安装 Web Credit 实例完成。该实例由 SupportPac WA82 提供，可以从以下的网址中下载：

http://www-4.ibm.com/software/ts/mqseries/txppacs/wa82.html

以下是由 WA82 提供的 readme.html 文件操作说明。大致步骤如下：

- 更新所提供的 webcredit.fdl 文件以与配置名称匹配。组名和系统名值需要改为 BUYXYZ 和 BUYXYZ01。

- 将 strarter 和 programs 目录复制到 WebClient\webpages 目录中。假设您已将该压缩文件解压缩到 Web 服务器上的 D:\MQWF\Scenario 目录中，这意味着：

  ```
  xcopy d:\mqwf\scenario\WebCredit\programs\*
  d:\mqwf\cfgs\XYZ1\WebClient\webpages\programs\*

  xcopy d:\mqwf\scenario\WebCredit\starter\*
  d:\mqwf\cfgs\XYZ1\WebClient\webpages\starter\* /S
  ```

- 修改 starter 目录中的.jsp 和.html 文件以指定非默认配置名称（用 XYZ1 代替 FMC）：
  /MQWFClient/xxx becomes /MQWFClient-XYZ1/xxx
- 更新 customerUPES：
  - 更新 env.bat 文件以反映正确目录名。
  - 用正确的 MQSeries 名称和 db2 用户 ID，以及密码更新 CustomerUPES.properties 文件。
- 导出正常正常正常运行时配置（包括两个工作流服务器）：

  ```
  fmcibie -u admin -p password -e config.fdl
  ```

- 将 config.fdl 文件输入清洁的构建时环境。
- 将修改过的 WebCredit.fdl 文件输入构建时环境。
- 将构建时的所有事物输出同时将其输入正常正常正常运行时环境。

  ```
  fmcibie -u admin -p password -o -t -i webcredit2.fdl
  ```

Web Credit 应用程序是一种 Web 驱动工作流应用程序。启动 MQSeries Workflow 构建时环境并选择 **WebCreditRequest** 进程。研究该流程，如图 3-70 所示。

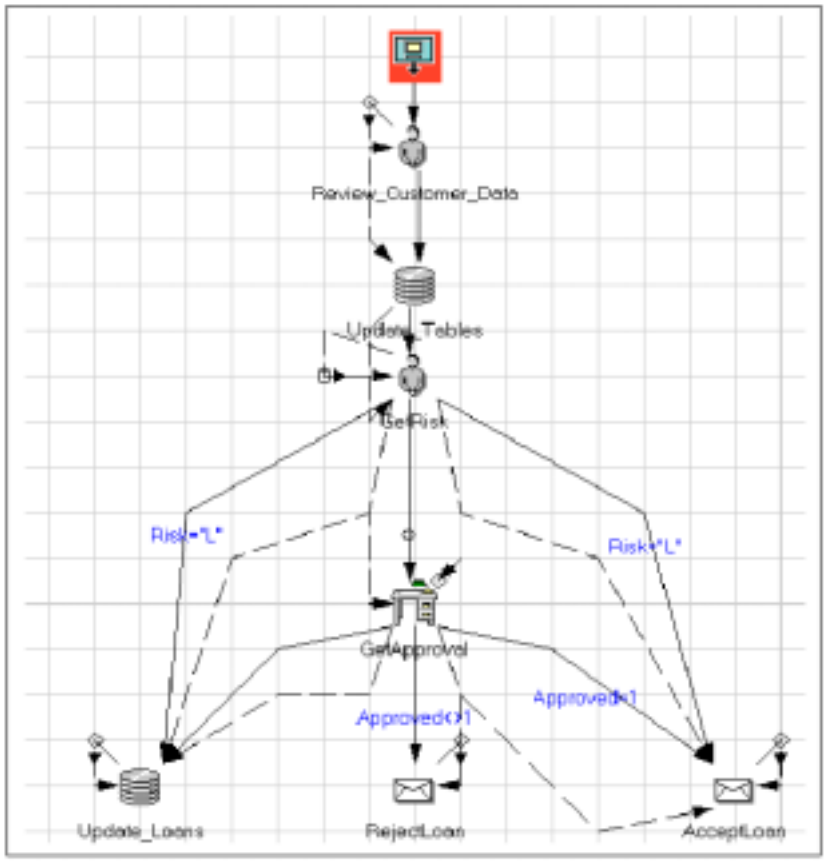

图 3-70WebCredit 应用程序流程模型

现在我们准备测试 WebCredit 应用程序。参考 readme.html 中关于启动 CustomerUPES 应用程序和使用浏览器启动新进程的操作说明。如图 3-71 所示，在登录 MQ Workflow Web 客户机时，在 BP 模板列表中我们会看到 WebCredit。

图 3-71 包含已输入的 WebCreditRequest 进程模板列表

启动 Web Credit 应用程序，请访问以下地址：

```
http://M23BZZYP/MQWFClient-XYZ1/starter/webcreditrequest.html
```

如果正确复制所有文件，您将看到如图 3-72 所示的 Web 页。

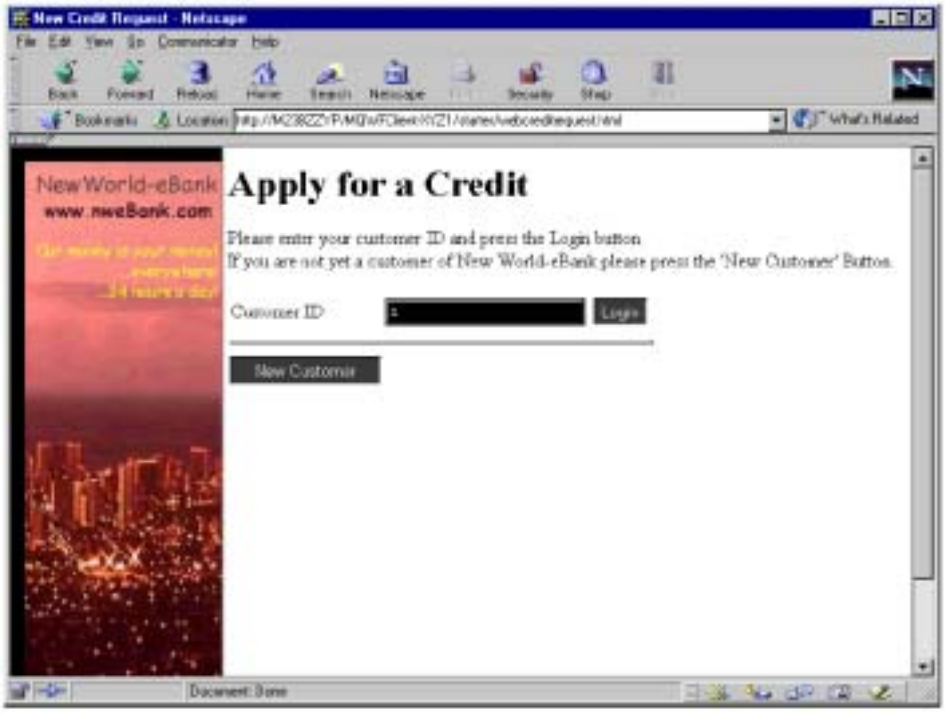

*图*3-72 *启动Credit 请求应用程序窗口*

给出客户 ID 并选择**新建客户（New Customer）**（或选择**登陆（Login），**如果您使用现有客户号）。这样将带您进入如图 3-73 所示的下一个 Web 页。

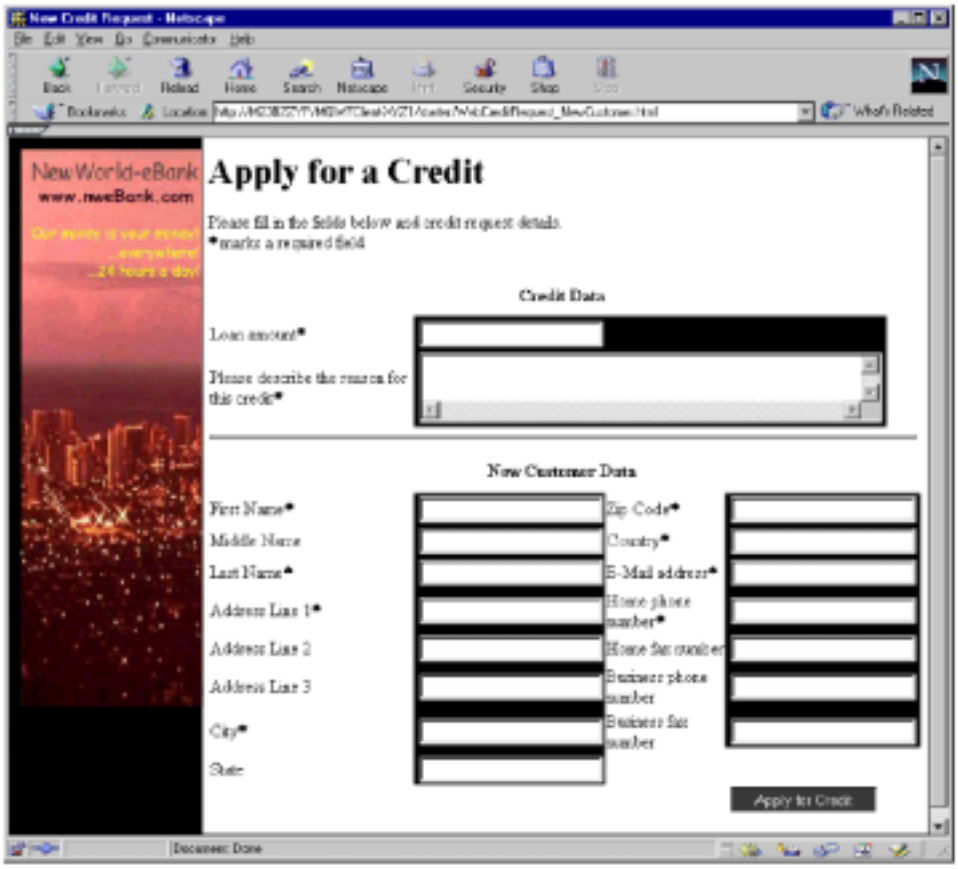

图3-73 客户注册窗口

提供关于您自己的详细资料和借贷请求，然后单击 **Credit 请求（Apply for Credit）**按钮。在登录 Web 页中使用现有客户 ID 登录时，您将进入如图 3-74 所示的网站。

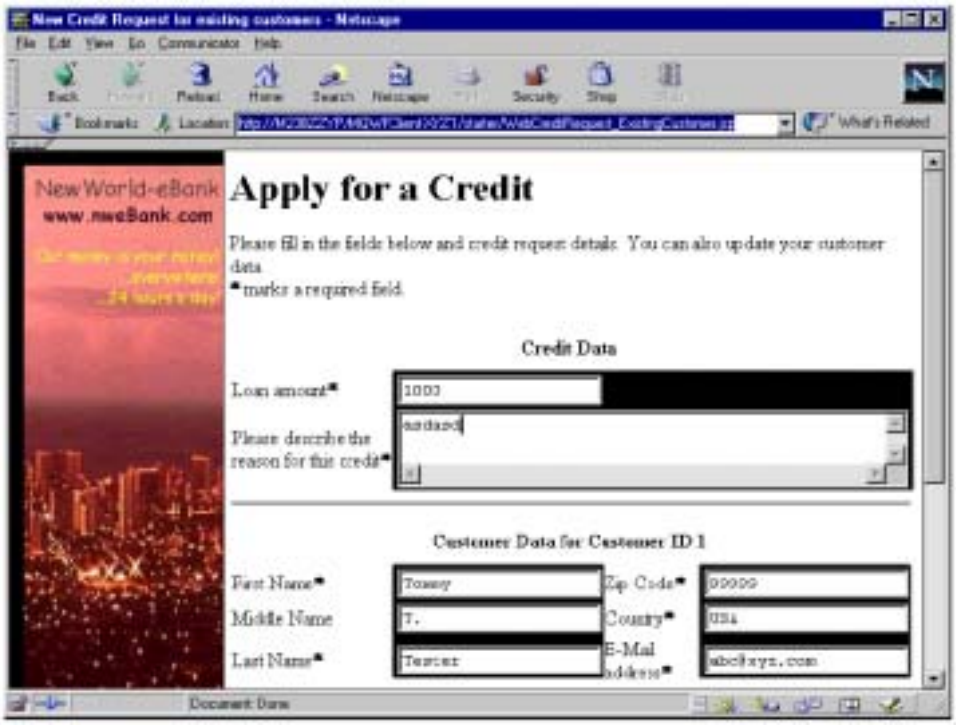

图3-74 现有客户新的借贷请求

就这一点来说，借贷请求将通过进程启动其方法。拥有查看所有工作项目权限的流程管理员可以看到该借贷请求（作为 WEBBANK_CLERK 用户列表中工作项目）（图 3-75）。

图 3-75 工作列表的管理视图

注意 ADMIN 用户控制该工作项目的同时其属用户 WEBBANK_CLERK 所有。要进行控制，请单击窗口左边圈示的按钮。

现在启动另一个浏览器窗口并登录 MQSeries WorkflowWeb 客户机作为 WEBBANK_CLERK 用户。当该用户选择工作项目视图时，他将获得如图 3-76 所示的 Web 页。

图 3-76 用户 WEBBANK_CLERK 的工作列表

通过单击**撤销**（check out）按钮（如图 3-76 圈示），用户 WEBBANK_CLERK 要求检验该工作项目并进入下一个 Web 页（如图 3-77 所示）以检查借贷请求。

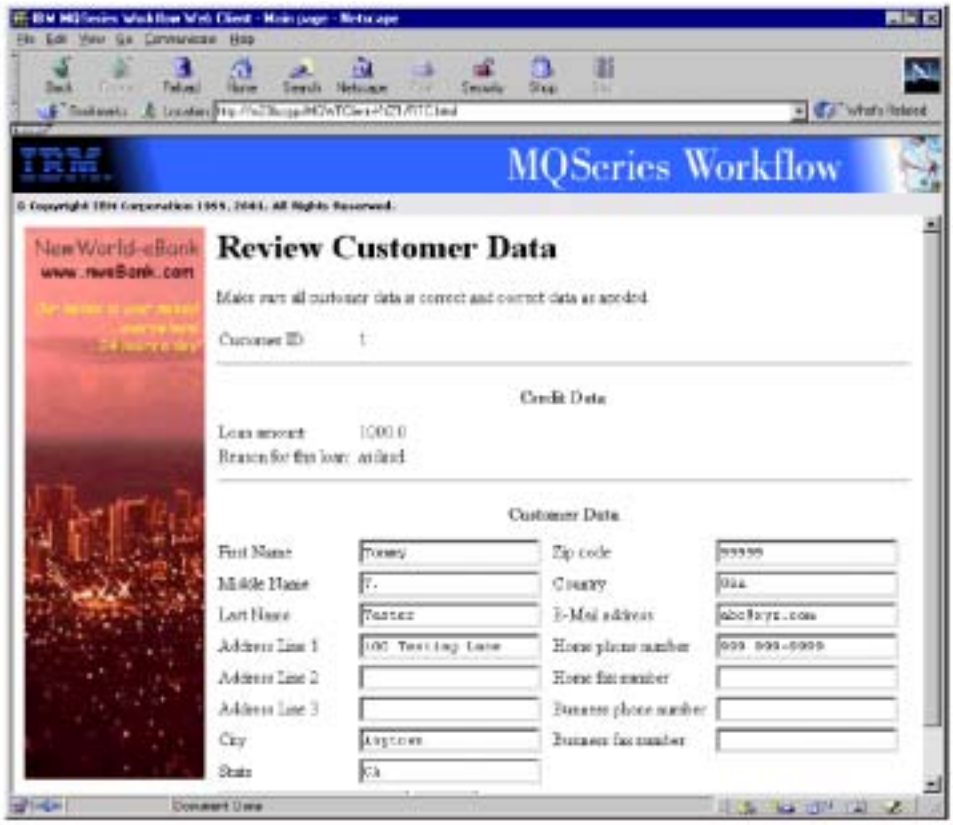

图3-77 检查客户数据窗口

在检查完数据之后，用户 WEBBANK_CLERK 将单击**完成工作项目（Complete Work Item）**按钮。此时，该请求将被发送到用户定义程序执行服务器（UPES）上，以便将其存储在数据库中。如果忘记启动 UPES 或者 UPES 出现故障，您可以利用 MQSeries Workflow Web 客户机监控工具来确认借贷请求是否已存入数据库。

在 ADMIN 用户工作项目视图中，单击**监控按钮**（the monitor button）：

图3-78 监控按钮

如图 3-79 所示的流程实例监控器，该 Web 页显示了完全工作流模型，以及实例的当前位置。当 Update_Tables 步骤运行的时候（双箭头），Review_Customer_Data 步骤已经完成（复选标记）。单个向下箭头表明该步骤还没运行或还没完成。

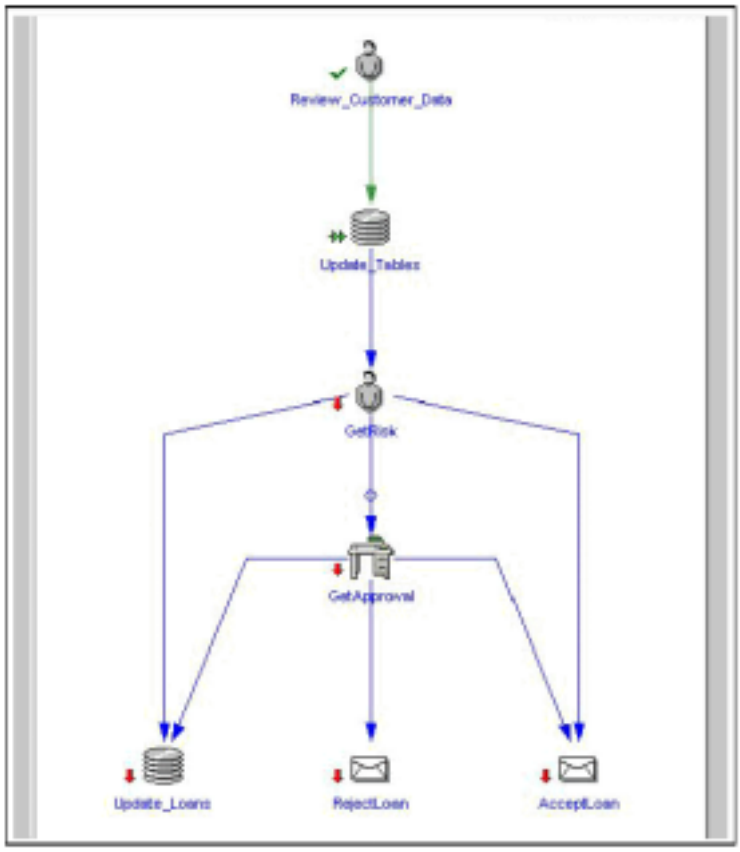

图3-79 进程实例状态

在该特殊情况下，由于错误指定数据库密码，UPES 程序运行失败。

在UPES完成其工作以后，新的工作项目将出现在用户WEBBANK_CLERK工作列表中。如果没有激活 MQSeries Workflow Web 客户机中的自动刷新功能，您可能需要单击**刷新（Refresh）**按钮。当 WEBBANK_CLERK 用户控制该工作项目时，他或她会显示在访问信用风险（Assess Credit Risk）页中（如图 3-80 所示）。

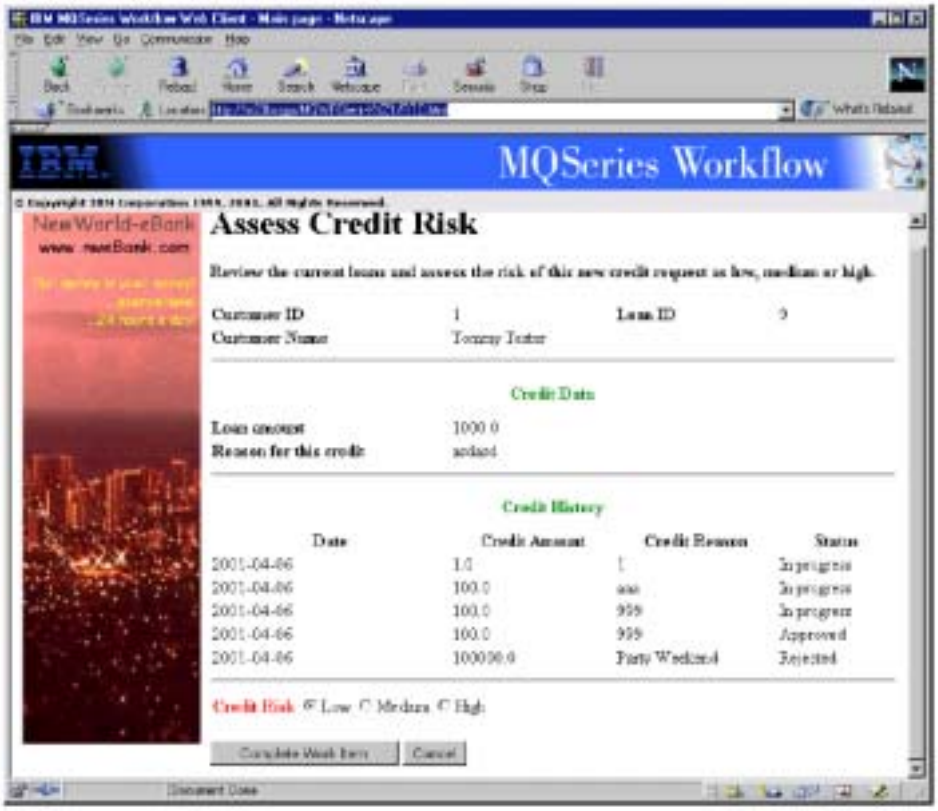

图 3-80 访问信用风险窗口

在完成该工作项目后，用户 WEBBANK_CLERK 将返回其工作列表。在那里，该流程实例将显示接受借贷的最后时间。如果信用风险设置得很高，该工作项目将传给用户 WEBBANK_MANAGER 来批准或者拒绝此项借贷。

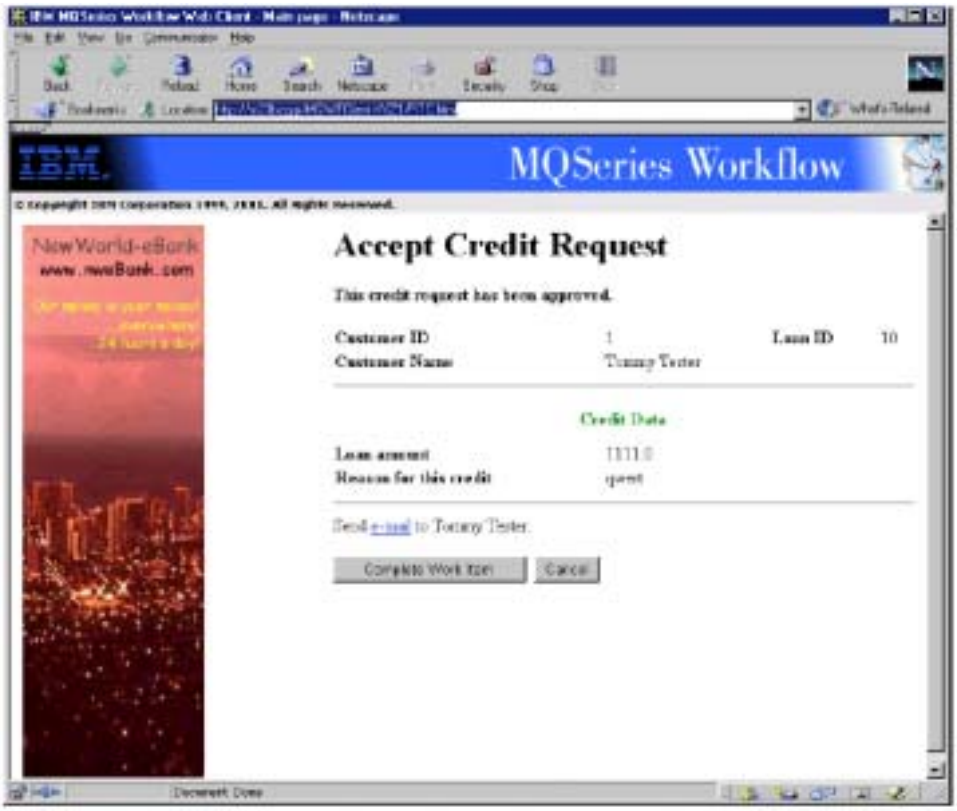

*图*3-81 *接受信用请求（Accept Credit Request）窗口*

该实例在很大程度上是对实际情况的一种简化，把三个不同步骤传给同一组织中不同的人。而在真实银行中，借贷请求的处理是相当复杂的。但是，该实例显示了 Web 客户机的力量。除了 UPES 以外，客户无需写任何代码。正如我们将在下一步（第 159 页的 4.3.2 “ JSP 快速布置向导 ”）中看到的，该应用程序的 Web 页是基于 Web 页生成的，您可以添加一些图片以使其更具吸引力。但是从本质上说，工作流模型会在 Web 页上瞬间出现在您面前。

示范该实例还有其它两个原因。首先，它可以帮助我们确认 Web 服务器、数据库服务器和两个工作流服务器的安装。当您在 MQSeries 资源管理器中看到通道统计表时，您也可以看到发送到两个工作流服务器中的消息。同时，该实例使我们能够解释工作流客户机的某些概念，特别是 MQSeries WorkflowWeb 客户机。

3.6 MQSeries Integrator环境

现在，我们已经配置了 MQSeries Workflow，接着我们准备在设备 M23CABWZ 和 M23CAAXY 上创建 MQSeries Integrator 环境。

我们将在数据库服务器 M23M1773 上创建配置管理器、配置和消息仓库数据库，而且我们将在其它两台设备上创建代理，每个代理都拥有自已的本地正常正常正常运行时数据库。

该配置原理将在本书的第 109 页的第四章“ 在 MQSeries Workflow 中实施模型 ”中加以解释。该设置将提供给我们负载均衡和故障恢复水平上的 MQSeries Integrator 代理，并且其尽可能地将数据库集中在一台服务器上。在本地创建代理数据库以增强代理的性能。

3.6.1 创建配置管理器数据库

首先，我们必须创建配置所需的 MQSeries 和 DB2 对象。

在数据库设备 M23M1773 上：

- DB2 已安装
- MQSeries 已安装

如需实现简易性和可重复性，我们必须使用脚本来创建所有 MQSeries 对象和 DB2 数据库，如实例 3-1 所示：

实例 3-1 创建 MQSeries 子系统和组件的命令脚本

```
echo Create a Queue Manager DBSQM (批处理 echo 创建队列管理器 DBSQM)
crtmqm -q -lf 1024 -lp 5 -ls 2 -u SYSTEM.DEAD.LETTER.QUEUE DBSQM

echo Create a Listener
amqmdain crtlsr DBSQM -t TCP -p 1414
```

```
echo Create a Channel Initiator（批处理 echo 创建通道启动程序）
amqmdain crtchi DBSQM SYSTEM.CHANNEL.INITQ

echo Set up automatic startup（批处理 echo 创建自动启动程序）
amqmdain auto DBSQM

echo Start a DBSQM Queue Manager（批处理 echo 启动 DBSQM 队列管理器）
amqmdain start DBSQM

echo Show the status of MQSeries services（批处理 echo 显示 MQSeries 服务的状态）
amqmdain status all

echo Run Script Commander for Queue Manager DBSQM to create objects（批处理 echo 运行队列管
理器 DBSQM 的脚本命令工具以创建对象）
runmqsc DBSQM <_dbsqm.cfg

pause
```

由上面脚本调用的脚本_dbsqm.cfg (该脚本创建所有必需的群集通道)如实例 3-2 所示：

实例 3-2 创建 MQSeries *对象的命令脚本*

```
*Define the cluster channels（定义群集通道）

DEFINE CHANNEL(TO.DBSQM.TCP)+
  CHLTYPE(CLUSRCVR)+
  TRPTYPE(TCP)+
  CONNAME(‘m23m1773 ’)+
  CLUSTER(BUYXYZ)+
  DESCR(‘Cluster-receiver channel for qm DBSQM ’)+
  REPLACE

DEFINE CHANNEL(TO.WF01QM.TCP)+
  CHLTYPE(CLUSSDR)+
  TRPTYPE(TCP)+
  CONNAME(‘m23cabyg(5010)’)+
  CLUSTER(BUYXYZ)+
  DESCR(‘Cluster-sender channel for qm DBSQM to qm WF01QM ‘)+
  REPLACE

DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN)+
  CHLTYPE(SVRCONN)+
  TRPTYPE(TCP)+
  DESCR(‘Server connection channel ’ )+
  REPLACE
```

将两个脚本复制到已在数据库设备 M23M1773 上命名并运行的文件中去。

在运行两个脚本之后，我们需要使用 MQSeries 资源管理器确认 MQSeries 配置是否在正常工作：

- 确认群集发送器和接收器正在运行
- 确认您可以在群集队列管理器文件夹中看到 WF01QM 和 WF02QM 队列管理器

在成功确认 MQSeries 配置之后，我们可以创建数据库 MQSICMDB 并使其 ODBC 可用。创建数据库脚本（必须在 DB2 命令窗口中运行），如实例 3-3 所示：

实例 3-3 创建 DB2 组件的命令脚本

```
echo Run DB2 Command Window before you execute this file（在您执行该文件之前批处理 echo 运行 DB2 命令窗口）

pause

echo Create the MQSICMDB database（批处理 echo 创建 MQSICMDB 窗口）
db2 create database MQSICMDB（db2 创建数据库 MQSICMDB）

echo Registering MQSICMDB as system ODBC source（批处理 echo 注册 MQSICMDB 作为系统 ODBC 源）
db2 catalog system odbc data source MQSICMDB（db2 目录系统 odbc 数据源 MQSICMDB）

echo List all your system ODBC data sources（批处理 echo 列出所有您系统 ODBC 数据源）
db2 list system odbc data sources（db2 列出系统 odbc 数据源）
```

如必要，可以从以下地址下载最新 MA88 SupportPac：

http://www-4.ibm.com/software/ts/mqseries/txppacs/ma88.html

注释：可能必须更新您的 Windows 安装程序系统。请从微软网站下载最新版本：

http://www.microsoft.com/downloads/release.asp?ReleaseID=17344

请按以下说明安装 MA88 SupportPac

将 MA88 安装到 mqseries\java 目录并确认 CLASSPATH，必须指出：

d:\mqseries \java \lib \com.ibm.mq.jar;d:\mqseries \java \lib \com.ibm.mqbind.jar;d:\ mqseries \java \lib

如使用 JMS，将必须在 CLASSPATH 中添加其他文件，其在 MA88 安装说明中有详细描述。然而，JMS 并非该 MQSeries Integrator 配置所必需。

现在，确认 PATH 环境变量，其包含：

d:\mqseries \java \lib

3.6.2 安装MQSeries Integrator

现在我们准备在数据库服务器 M23M1773 上安装 Windows NT 的 MQSeries Integrator 版本 2.01。

安装 MQSI 将要求您指定目录。您可能想选择比默认名称更短的名字，例如 MQSI。

选择**定制安装(Custom Install)**，然后选择如图 3-82 所示的组件。

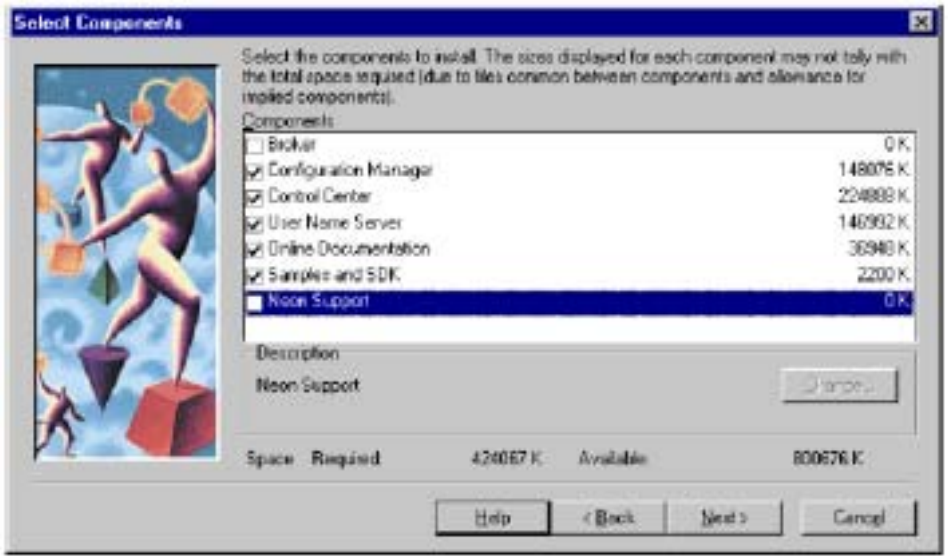

图3-82MQSeries Integrator 安装组件

如有需要，请重启机器。

安装 MQSeries Integrator 版本 2.01 CSD 1 (U200132)，可从以下地址获得该安装程序：

http://www-4.ibm.com/software/ts/mqseries/support/summary/mqsi.html

按照给出的安装说明进行安装，然后重启机器。

现在我们可以使用实例 3-4 中的脚本创建配置管理器：

实例 3-4 创建 MQSeries Integrator 配置管理器命令脚本

```
mqsicreateconfigmgr -i db2admin -a db2admin -q DBSQM -s DBSQM -n MQSICMDB -m
MQSICMDB
```

使用实例 3-5 所示的脚本创建用户名服务器。

实例 3-5 创建 MQSeries Integrator 用户名管理器命令脚本

```
mqsicreateusernameserver -i db2admin -a db2admin -q DBSQM
```

记住要在新的 MQSeries Integrator 组中添加以下用户名：

- Mqbrasgn
- Mqbrdevt
- Mqbrks
- Mqbrops
- mqbrtpic

还要检查您的用户名是否已添加到下面 MQSeries 组中：

- mqm

如果您正在使用 Windows NT 域，有关详细信息请参考《MQSeries Integrator 2.0.1 安装指南》编号：GC34-5600 中的第二章 “ 安装计划 ”。

使用 Services JAVA 小程序设置配置管理器并且将用户名服务器设置为自动启动，然后启动它们。

检查事件日志以确认安装成功。

启动 MQSeries Integrator 控制中心以确认配置管理器已经成功配置。输入如图 3-83 所示的详细信息。

图 3-83 控制中心的连接设置

3.6.3 创建代理

现在我们准备创建 MQSeries Integrator 代理。

以下软件已安装：

- Windows NT 的 MQSeries 版本 5.2（MQSeries for Windows NT V5.2）
- Windows NT 的 DB2 版本 7 + FixPak 1（DB2 for Windows NT V7 + FixPak 1）

下一步，使用实例 3-6 中的脚本在设备 M23CABWZ 上创建队列管理器 MQSI01QM：

实例 3-6 创建代理及其队列管理器的命令脚本

```
echo Create a Queue Manager MQSI01QM（echo 创建队列管理器 MQSI01QM）
crtmqm -q -lf 1024 -lp 5 -ls 2 -u SYSTEM.DEAD.LETTER.QUEUE MQSI01QM

echo Create a Listener（echo 创建侦听器）
amqmdain crtlsr MQSI01QM -t TCP -p 1414

echo Create a Channel Initiator（echo 创建通道启动器）
amqmdain crtchi MQSI01QM SYSTEM.CHANNEL.INITQ

echo Set up automatic startup（echo 设置自动启动）
amqmdain auto MQSI01QM

echo Start a MQSI01QM Queue Manager（echo 启动 MQSI01QM 队列管理器）
amqmdain start MQSI01QM

echo Show the status of MQSeries services（echo 显示 MQSeries 服务状态）
amqmdain status all

echo Run Script Commander for Queue Manager MQSI01QM to create objects（批处理 echo 运行队列
管理器 MQSI01QM 的脚本命令工具以创建对象）
runmqsc MQSI01QM <_mqsi01qm.cfg

pause
```

由上面脚本调用的脚本_mqsi01.cfg (该脚本创建所有必需的群集通道)如实例 3-7 所示：

实例 3-7 创建代理的 MQSeries 对象命令脚本

```
*Define the cluster channels（定义群集通道）
DEFINE CHANNEL(TO.MQSI01QM.TCP)+
  CHLTYPE(CLUSRCVR)+
  TRPTYPE(TCP)+
  CONNAME(‘m23cabwz ’)+
  CLUSTER(BUYXYZ)+
```

```
  DESCR( ‘ Cluster-receiver channel for qm MQSI01QM ’ )+
  REPLACE

DEFINE CHANNEL(TO.WF01QM.TCP)+
  CHLTYPE(CLUSSDR)+
  TRPTYPE(TCP)+
  CONNAME( ‘ m23cabyg(5010) ’ )+
  CLUSTER(BUYXYZ)+
  DESCR( ‘ Cluster-sender channel for qm MQSI01QM to qm WF01QM ‘ )+
  REPLACE

* Define the server connection channel（定义服务器连接通道）

DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN)+
  CHLTYPE(SVRCONN)+
  TRPTYPE(TCP)+
  DESCR( ‘ Server connection channel ’ )+
  REPLACE
```

现在修改这些脚本以便在设备 M23CAAXY 上创建队列管理器 MQSI02QM。

使用 MQSeries 资源管理器检查队列管理器是否在群集中，及是否处于正常工作之中。

使用以下脚本，我们可以在代理设备上创建数据库并使其 ODBC 接口可用。

如实例 3-8 所示，该脚本将用于在 M23CABWZ 上创建数据库 MQSI01BK，它应在 DB2 命令窗口中运行。

实例 3-8 创建数据库对象脚本

```
echo Run DB2 Command Window before you execute this file（在您执行该文件之前批处理 echo 运行 DB2 命令窗口）

pause

echo Create the MQSI01BK database（批处理 echo 创建 MQSI01BK 数据库）
db2 create database MQSI01BK（db2 创建数据库 MQSI01BK）

echo Registering MQSI01BK as system ODBC source（批处理 echo 注册 MQSI01BK 作为系统 ODBC 源）
db2 catalog system odbc data source MQSI01BK（db2 目录系统 odbc 数据源 MQSI01BK）

echo List all your system ODBC data sources（批处理 echo 列出所有您系统 ODBC 数据源）
db2 list system odbc data sources（db2 列出 odbc 系统数据源）
```

在脚本成功完成之后，安装先前下载的 MA88 SupportPac。

将 MA88 安装到 mqseries\java 目录中并确认 CLASSPATH 和 PATH 环境变量。

现在，我们必须将 Windows NT 的 MQSeries Integrator 版本 2.01 安装到设备 M23CABWZ 上代理服务器 MQSI01BK 中。

安装向导将要求我们指定安装 MQSeries Integrator 的目录，可以选择比默认目录名简短的名称，例如 MQSI。

选择**定制安装（Custom Install）**，然后选择如图 3-84 所示的组件。

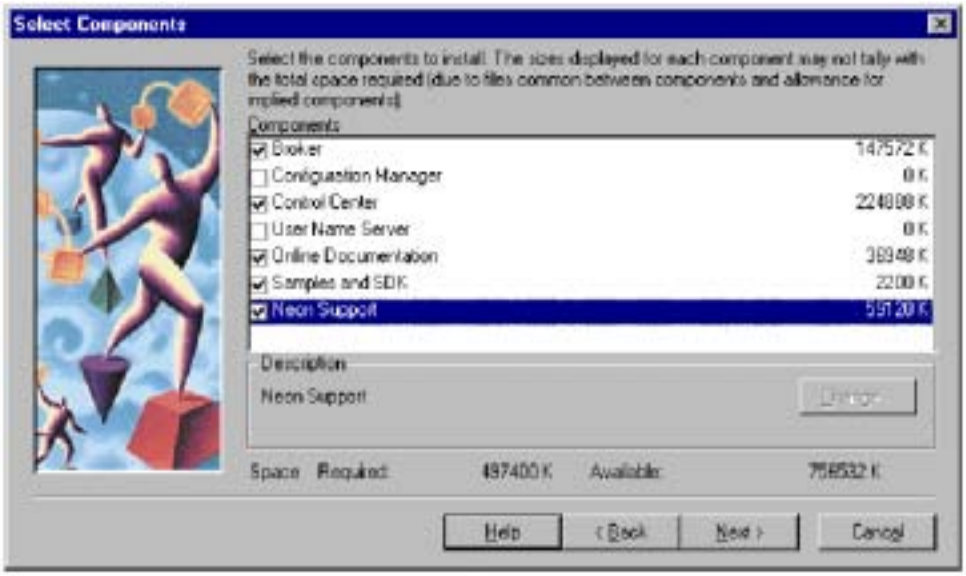

图 3-84 选择要安装的 MQSeries Integrator 组件

选择 **DB2 兼容程序（DB2 Compatible Libraries）**以支持 NEON Rules 和 Formats。这并不是我们解决方案所需的，但是若有需要允许将其包括进去。

如程序要求，请重启机器。

安装 MQSeries Integrator 版本 2.01 CSD 1 （U200132）并重启机器。

使用如实例 3-8 所示的脚本创建代理：

```
mqsicreatebroker MQSI01BK -i db2admin -a db2admin -q MQSI01QM -s DBSQM -n MQSI01BK
```

记住要将所有用户名添加到新的 MQSeries Integrator 组中：

- Mqbrasgn
- Mqbrdevt
- Mqbrks
- Mqbrops
- mqbrtpic

还要检查您的用户名是否已添加到 MQSeries 组中：

- mqm

如果您正在使用 Windows NT 域，有关详细信息请参考《*MQSeries Integrator 2.0.1 安装指南》编号：*GC34-5600 中的第二章“ 安装计划 ”。

使用 JAVA 小程序 Services 将代理设置为自动启动，并同时启动代理。检查事件日志以确认启动成功完成。

在创建和启动代理之后，建议您通过创建和应用简单消息流来检验安装。

启动 MQSeries Integrator 控制中心开始检验代理。

图 8-35 确定控制中心(Control Center)的连接参数

现在将两个代理加入拓扑：

1. 选择**拓扑(Topology)**标签。
2. 在左窗格中右键单击**拓扑(Topology)->注销（Check Out）**。
3. 右键单击**拓扑(Topology)->创建（Create） ->代理（Broker）**。

4. 给出代理名称：MQSI01BK。
5. 给出与代理相关的队列管理器名称：MQSI01QM。

为具有队列管理器 MQSI02QM 的第二个代理 MQSI02BK 重复这些步骤。

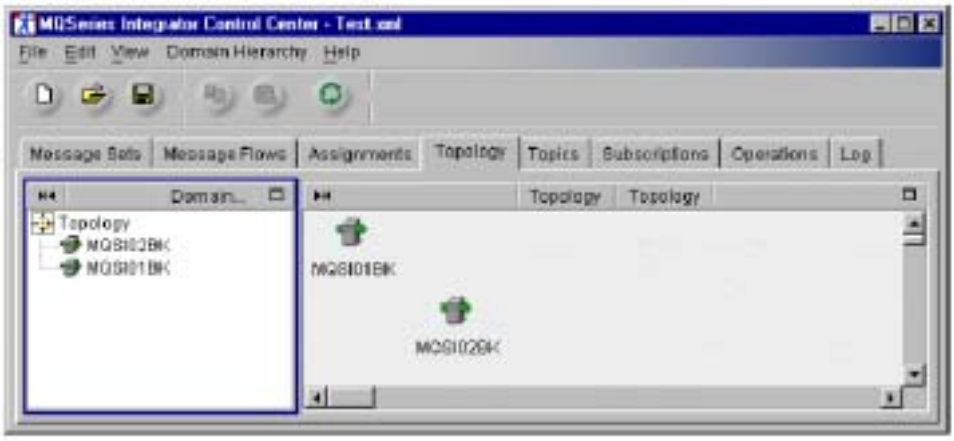

图3-86 将现有的代理加入拓扑

创建和应用简单消息流。输入节点应使用在两个代理队列管理管理器上定义的群集队列。指定消息域为 BLOB。定义带有默认约束选项 *notfixed* 的队列来使用群集。

在输出节点中，指定在工作流队列管理器中定义的群集队列名称。不要在输出节点中设置队列管理器名称。当定义输出群集队列时，您要再次确保将约束选项设为 *notfixe*d。

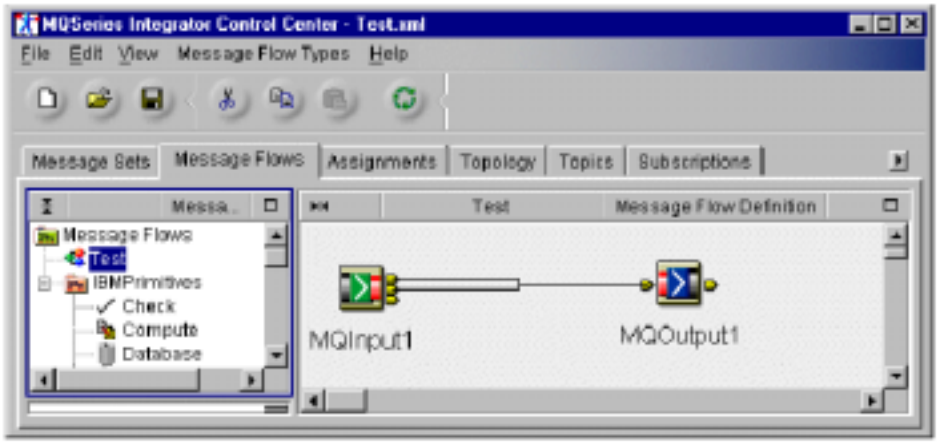

图3-87 确认简单消息流

将消息流分配到两个代理上并应用配置。检查 MQSeries Integrator 控制中心日志以确认成功应用。

将消息加入工作流服务器 BUYXYZ01 中的输入队列，然后检查它将到达指定的输出节点群集队列。我们可以使用 MQSeries 实例程序 amqsput 和 amqsget 来实现上述操作。因为输入队列不是工作流服务器中的本地队列，所以群集技术将在 MQSeries Integrator 代理中选择输入队列实例。当代理打开输出队列时，群集技术将再次在工作流服务器中选择输出队列实例。这样，当加入来自工作流服务器 BUYXYZ01 的消息时，结果消息可能到达同一工作流服务器或是其它工作流服务器，反之亦然。

现在我们已配置了完整的环境，并且准备开始创建实际工作流和消息流。这些内容将在以下各章中详细描述。

4

在MQSeries Workflow中实现模型

在本章中，我们将在 MQSeries Workflow 中通过解决方案模型来设置。我们将从简单模型开始，模型中仅有一个供应商并且无需审批过程。而且由于有相当多的实施活动和条件需要测试(在启始阶段,我们使用运行在 MQSeries Workflow 中的测试程序进行测试)，建模过程需要分阶段实施以实现进程实例。一旦确认进程流中的所有路径有效，所有活动将在真实环境中执行。

4.1 设计工作流：第一阶段

本节中，我们将开始实现业务案例。在该案例中，供应商总是以有效响应按时应答。

我们的工作流展示了支持客户定单履行功能的 BuyXYZ 业务流程。完全的工作流模型如图 4-1 所示。该流程从接受定单数据开始（映射成适当活动），然后核实客户信息并确认在库存中有足够数量存货以履行该定单。如果库存不足，将由库存控制人员决定订购货物数量的多少。在确定数量后，我们将创建供应品定单并发送给适合的产品供应商。一旦确定有充足的库存，我们将立刻执行内部计帐功能。完成计帐后，我们将确认客户定单并将其转发给运送部门。

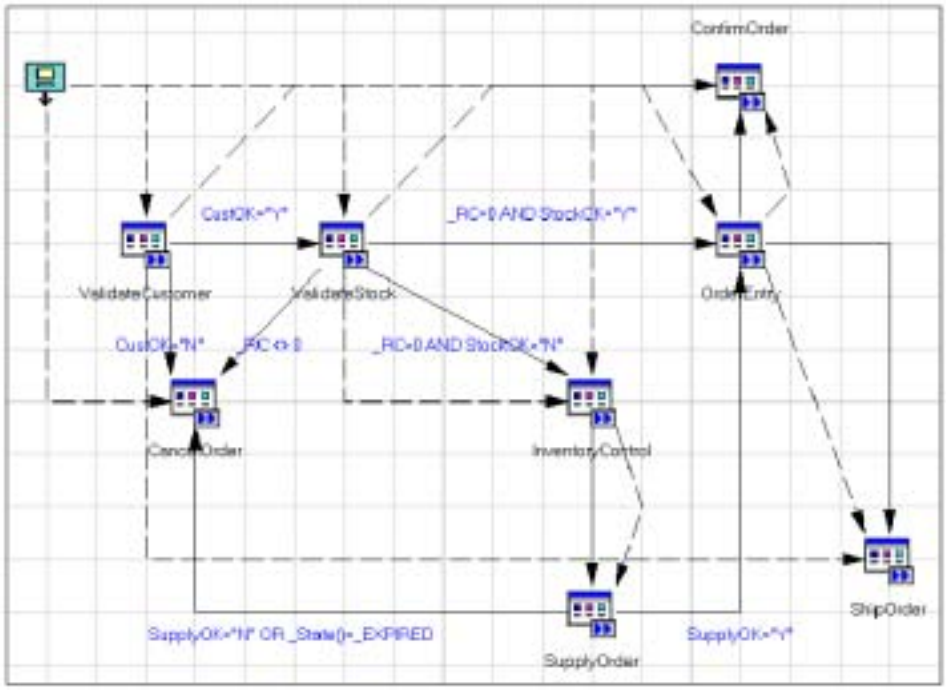

图 4-1MQSeries Workflow 构建时 BUYXYZ 定单流程

在第一阶段，我们集中在数据容器和控制逻辑上。基于数据容器中的某些数值，必须在工作流模型中选择确定路径。我们使用由 MQSeries Workflow 产品提供的称为 *fmcnshow* 的实用程序来帮助调试工作流。该实用程序可用于此步骤中的所有活动。它使您能够检验输入数据容器中的数值并指定输出数据容器。在数据容器中因为您已经完全控制指定了什么值，就更容易使通过工作流的所有可能路径有效。在第二阶段，我们将以真实活动代替该实用程序。

4.1.1 MQSeries构建时环境

本节中，我们将带您完成高级步骤以为 BUYXYZ 定单流程创建工作流程定义。我们假定您在某种程度上熟悉构建时环境。若需更详细的介绍，请参考红皮书《Windows NT 的 MQSeries Workflow 入门》编号：SG24-5848。

登录到构建时环境

流程中的第一步是登陆到构建时环境，如下所示：

1. 选择 Start -> Programs -> IBM MQSeries Workflow -> MQSeries Workflow Buildtime – XYZ，将出现构建时登录窗口（如图 4-2 所示）。

图 4-2 MQSeries Workflow 构建时登录窗口

2. 输入 ADMIN 作为用户 ID，password 作为口令。
3. 单击确定（OK）。
4. 您将收到提醒您改变口令的警告消息，单击确定（OK）。

这将带您进入构建时环境（如图 4-3 所示）。

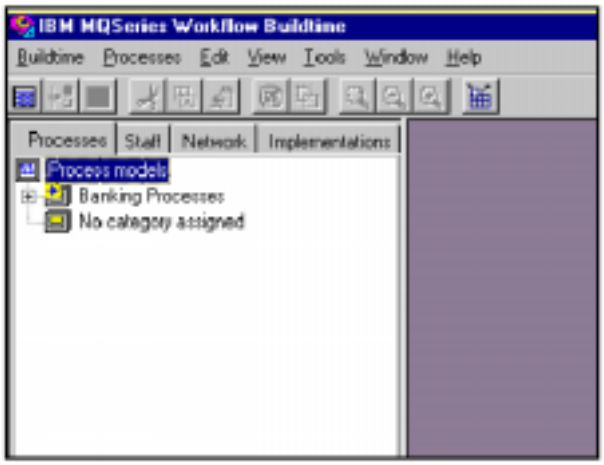

图 4-3MQSeries Workflow 构建时环境

创建目录和新流程模型

1. 右键单击流程模型（Process models）并选择新目录（New Category），将出现目录窗口（如图 4-4 所示）。

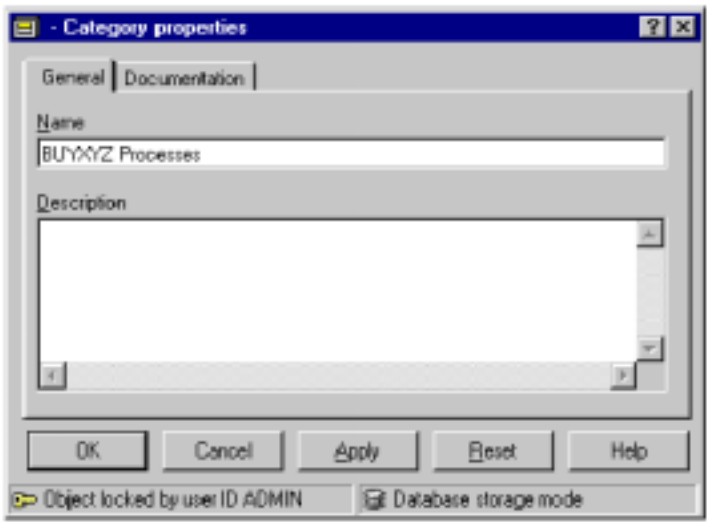

图 4-4 创建新目录

2. 输入目录名和可选描述。
3. 单击确认(OK)，将目录添加到模拟数据流中。
4. 右键单击 BUYXYZ Processes 并选择新流程（New Process），将出现流程属性窗口（如图 4-5 所示）。

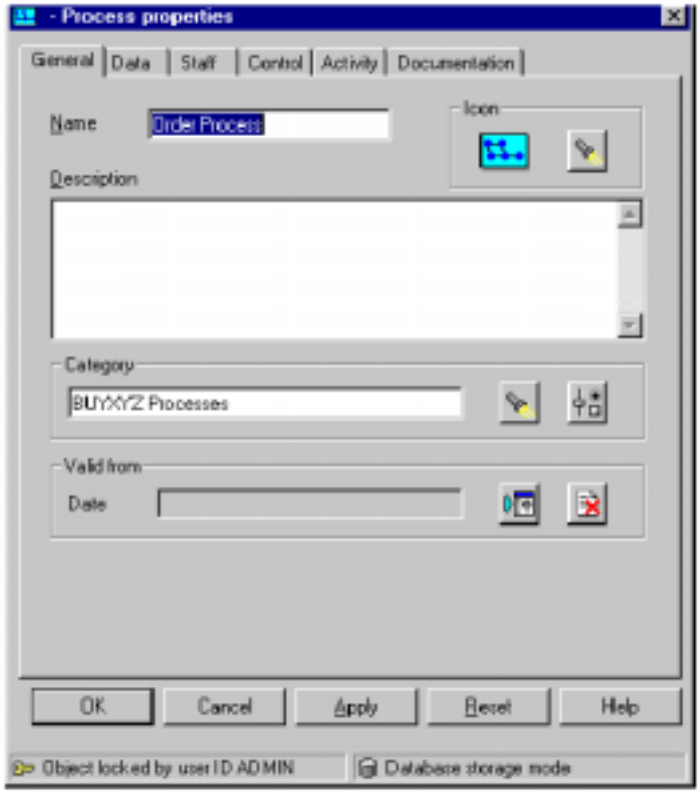

图 4-5 创建新流程

5. 输入流程名（我们选择“命令流程”）和可选描述。
6. 单击确定（OK）。

这将添加流程到构建时环境。在定义数据结构之后，我们将回到该窗口以完成流程定义。

4.1.2 确定数据结构

在分析 BUYXYZ 定单流程期间，确定每个业务流程活动的数据需求和相关性是很重要的。一旦确定该信息，我们就可以在 MQSeries Workflow 构建时环境中模拟数据流。

当创建工作流流程所需的数据结构时，存在两个选择。第一个选择是在单一结构中包含所有数据元素并确定将该结构作为每个活动的输入/输出容器。这将简化数据映射过程，但有几个缺点。如果有大量的数据需求，这样做将导致巨大的数据容器传递给每个活动，即使该活动只需其中的几个数据元素。对可维护性方面来说影响更显著。如果需要其他数据元素，其可能会影响使用该结构的所有活动。该影响对利用 UPES/XML 接口的自动化活动尤为突出。

因此随想而知，我们将实行第二个选择，其将模拟仅含每个活动所需的数据容器。完成详细数据流建模所花费的额外时间将由易于实现的可维护性来补偿。

订单数据结构

该结构包含启动业务流程所需的信息。

1. 选择执行（Implementations）标签，如图 4-6 所示。

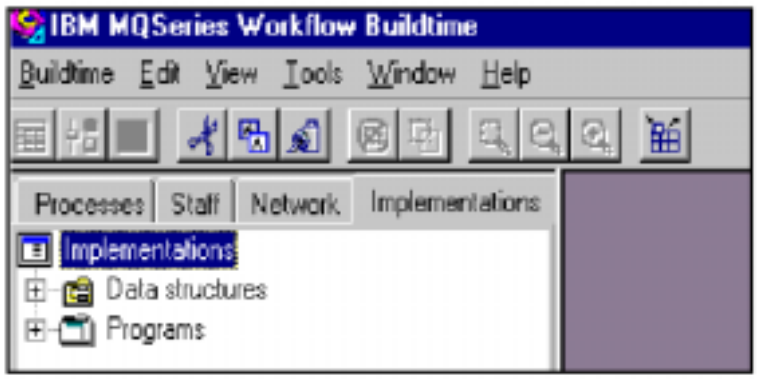

图 4-6 执行标签

在此，您将定义数据结构和支持业务流程活动的执行程序。

2. 右键单击数据结构（Data structures）并选择新数据结构（New Data Structure），将出现数据结构属性窗口（如图 4-7 所示）。

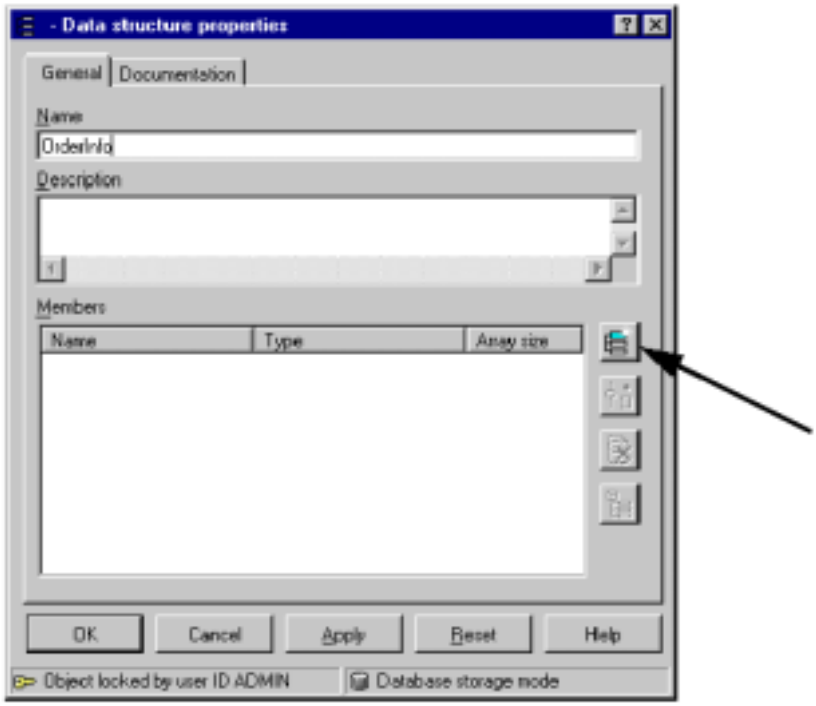

图 4-7 创建定单信息数据结构

3. 输入数据结构名和可选描述。
4. 单击下面的图标添加数据元素（参见图 4-7 中的箭头）：

将出现数据结构成员属性窗口（如图 4-8 所示）。

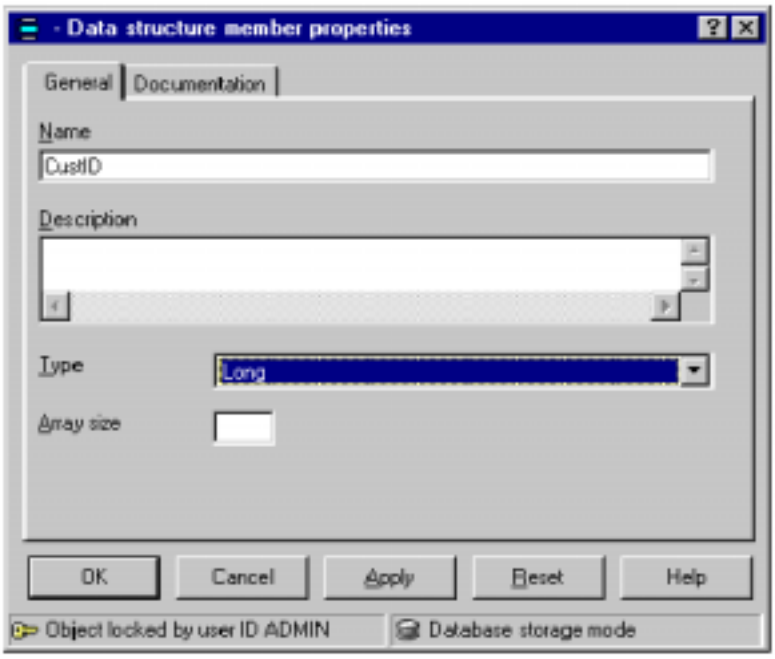

图 4-8 添加数据结构成员窗口

5. 输入数据成员名，本案例中为 CustID。
6. 下拉列表类型并选择 Long。
7. 单击确定（OK）。

重复该过程，输入剩下的数据结构元素。完成后，订单信息数据结构将如图 4-9 所示。

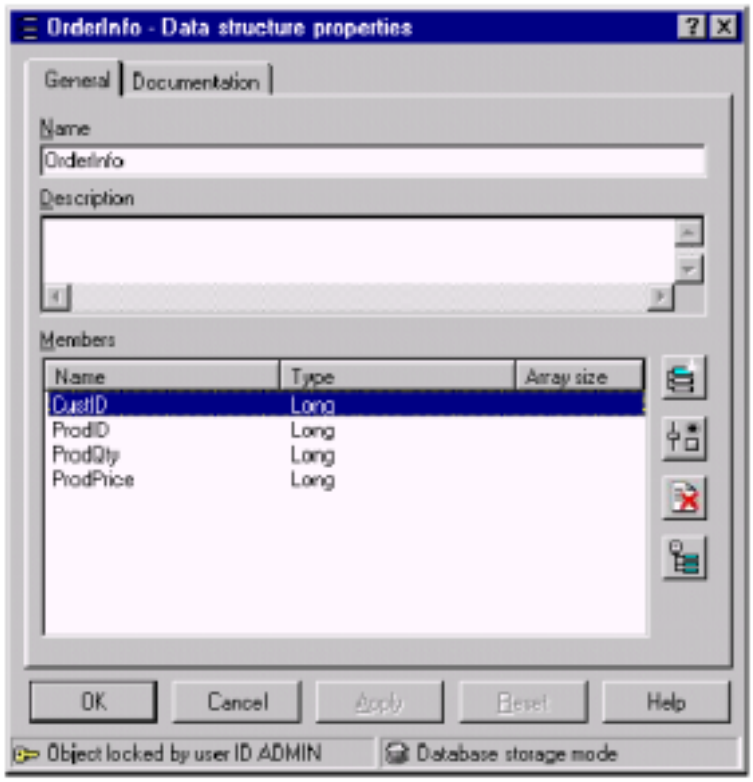

图 4-9 已完成的定单信息数据结构

重复该过程以保留数据结构。

确认客户

如图 4-10 所示定义数据结构。它将成为确认客户活动的输入容器。

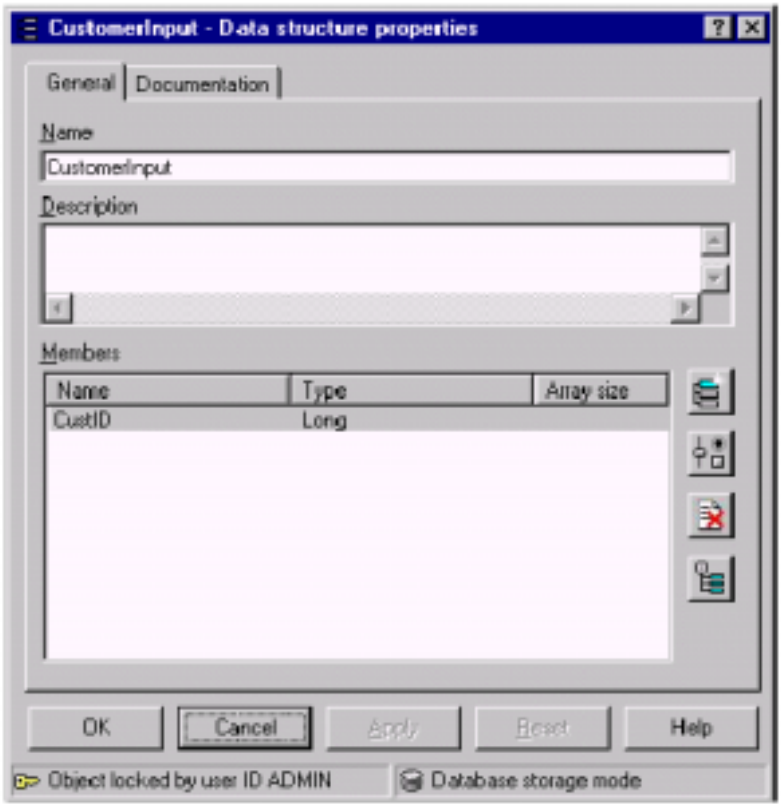

图 4-10 已完成的客户输入数据结构

如图 4-11 所示定义数据结构。它将成为确认客户活动的输出容器。

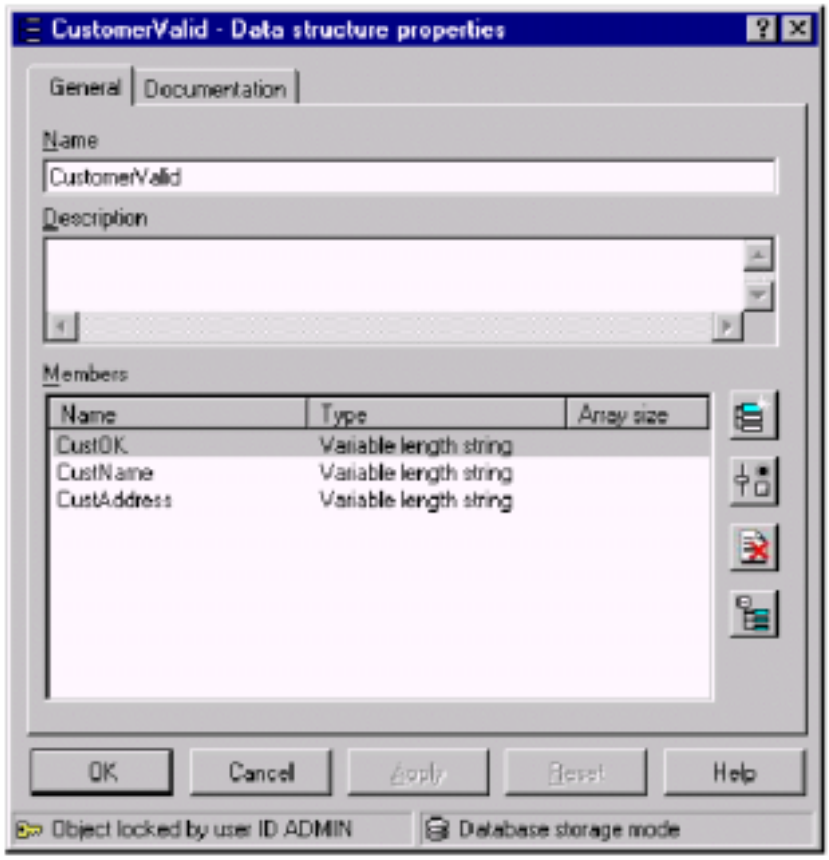

图 4-11 已完成的客户有效（CustomerValid）数据结构

确认库存

如图 4-12 所示定义数据结构。它将成为确认库存活动的输入容器。

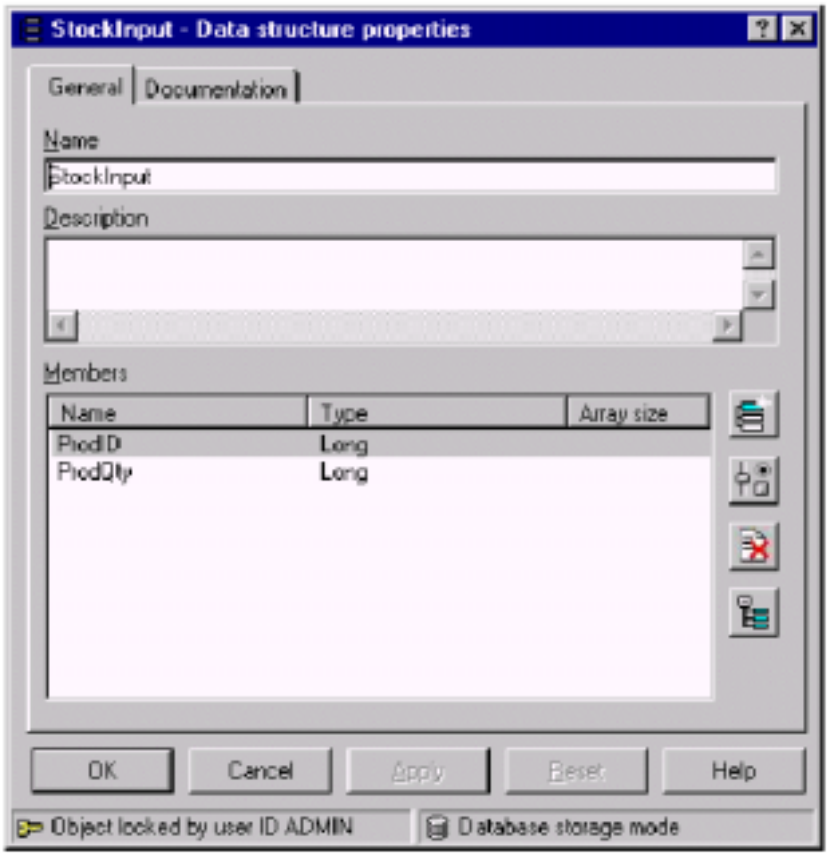

图 4-12 已完成的库存输入（StockInput）数据结构

如图 4-13 所示定义数据结构。它将成为确认库存（ValidateStock）活动的输出容器。

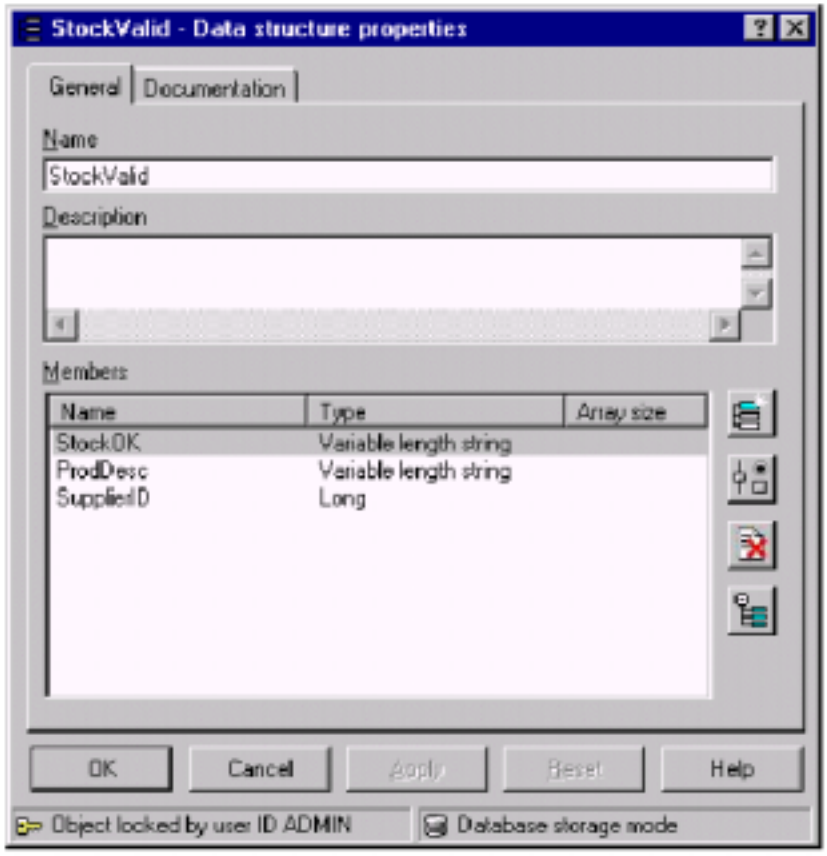

图 4-13 已完成的库存（StockValid）数据结构

库存控制和供应定单

如图 4-14 所示定义数据结构。它将成为库存控制活动的输入/输出容器，也将是供应订单活动的输入容器。

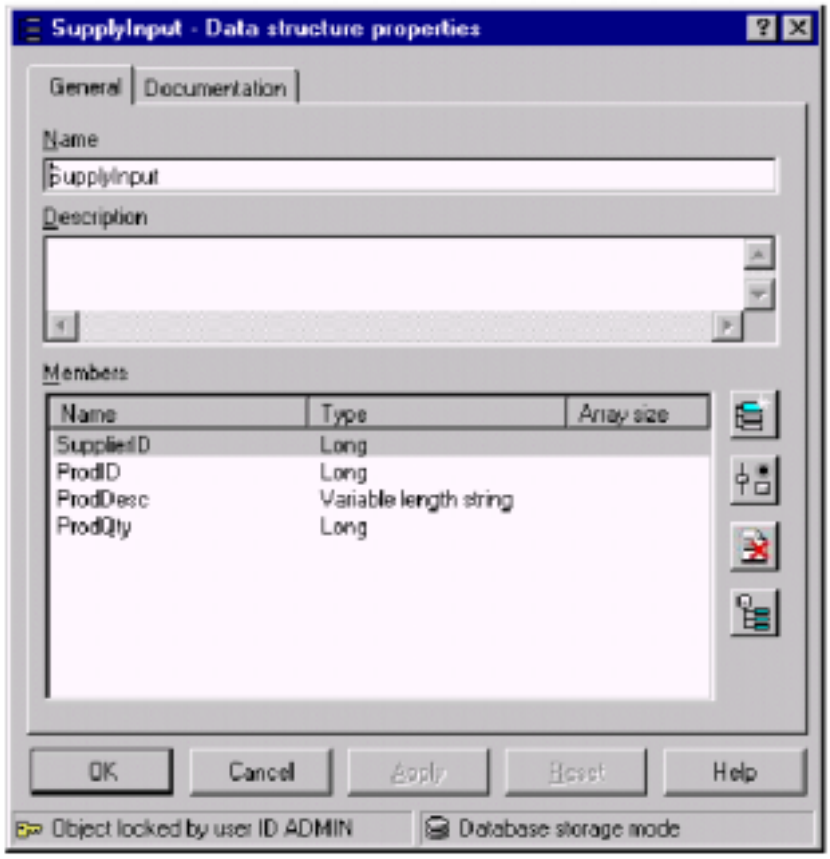

图 4-14 已完成的供应输入（SupplyInput）数据结构

如图 4-15 所示定义数据结构。它将成为供应订单活动的输出容器。

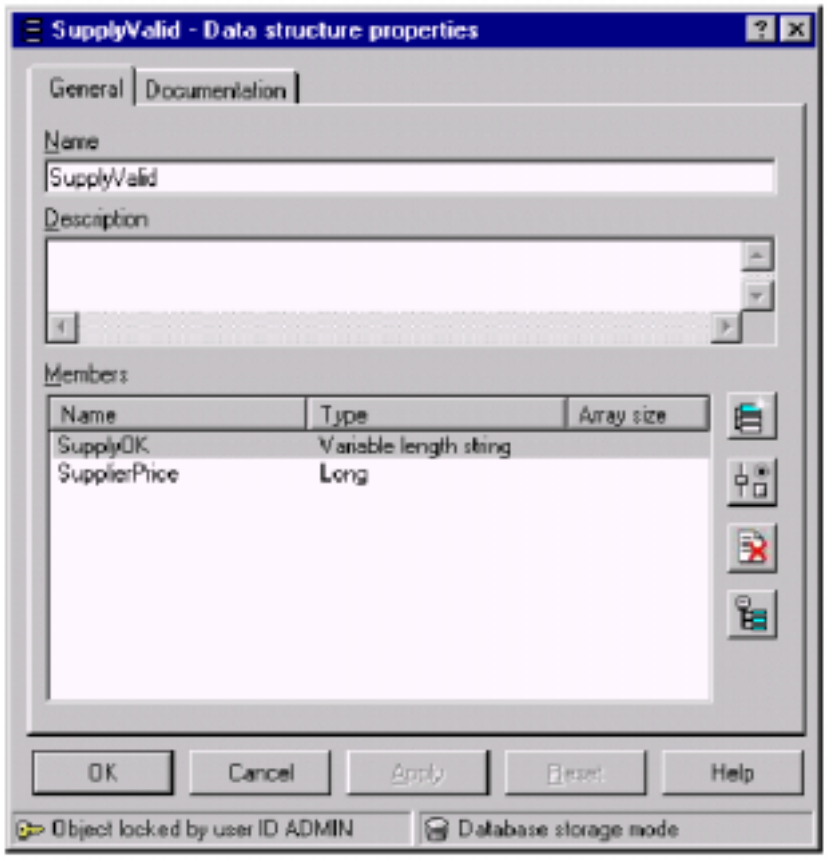

图 4-15 已完成的供应有效（SupplyValid）数据结构

运送订单（ShipOrder）

如图 4-16 所示定义数据结构。它将成为运送订单活动的输入容器。

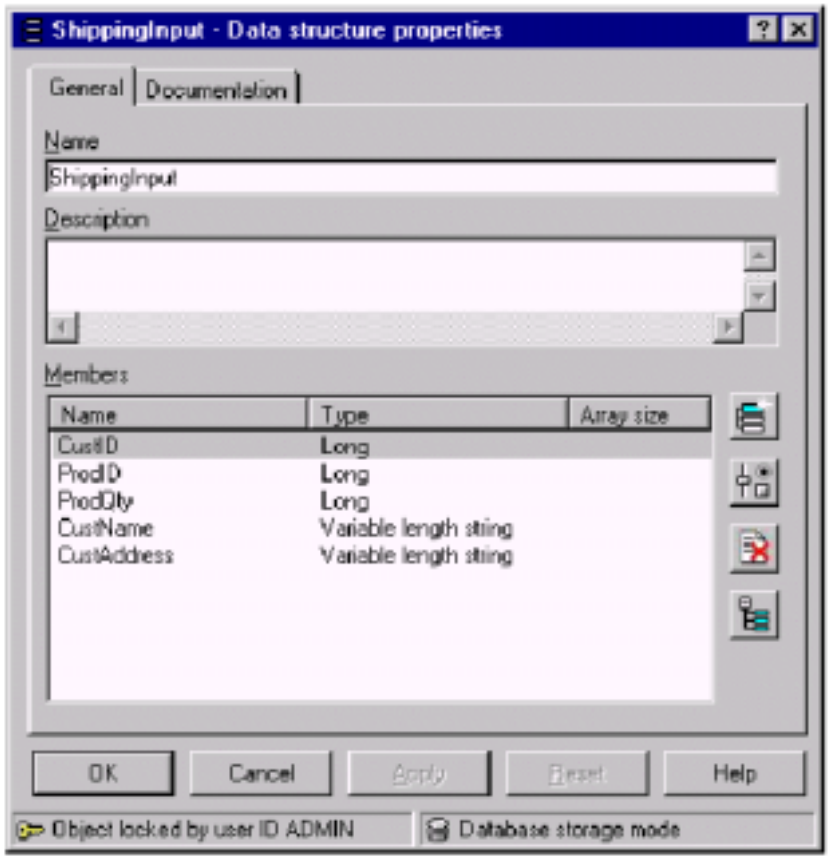

图 4-16 已完成的库存输入（StockInput）数据结构

客户确认

如图 4-17 所示定义数据结构。它将成为客户确认活动的输入容器。

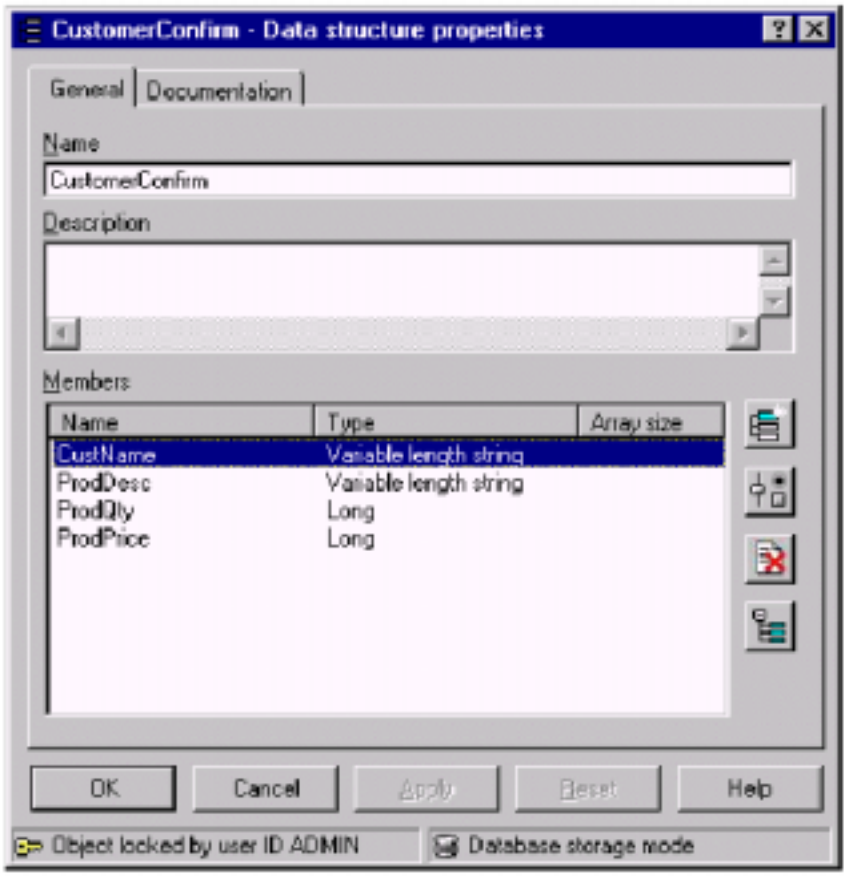

图 4-17 已完成的客户确认（CustomerConfirm）数据结构

4.1.3 创建执行程序fmcnshow

1. 右键单击程序（Programs）并选择新程序（New Program），将出现程序属性窗口。

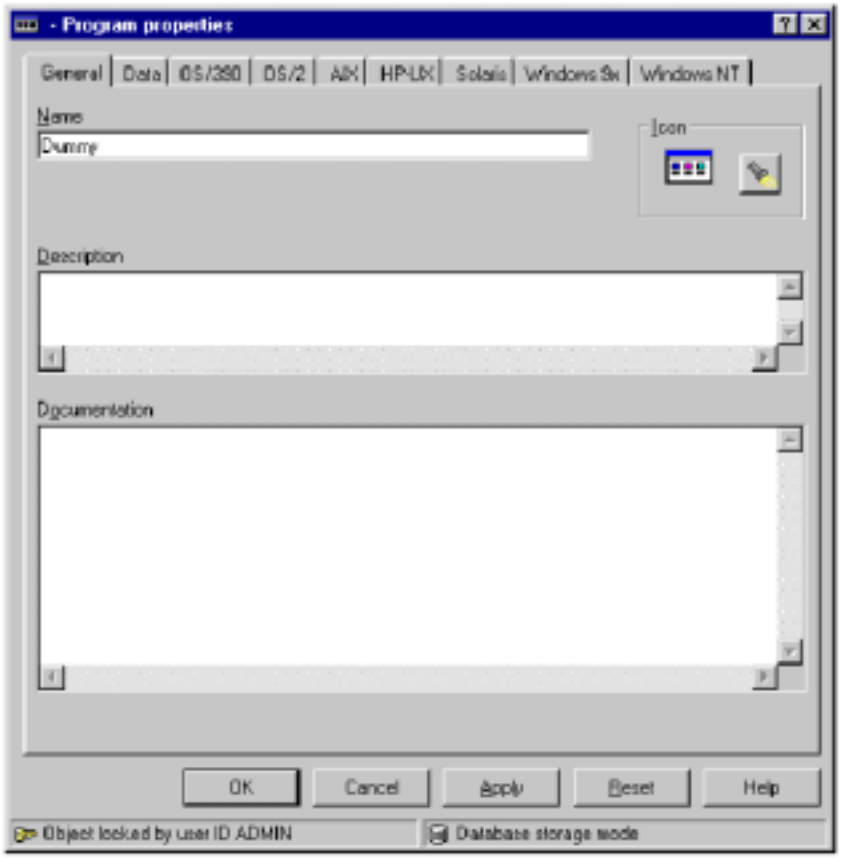

图 4-18 创建执行哑元（Dummy）程序

2. 输入程序名和可选描述。

3. 选择数据（Data）标签。

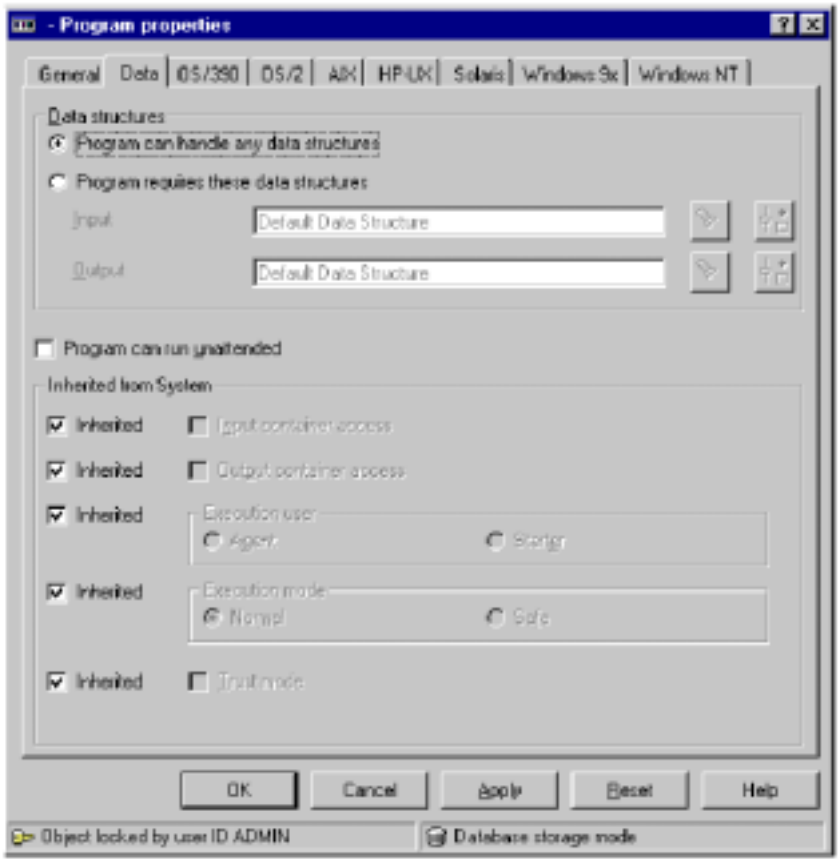

图 4-19 哑元（Dummy）数据详述

4. 选择可处理任何数据结构的程序（Program can handle any data structures）。

5. 选择 Windows NT 标签 。

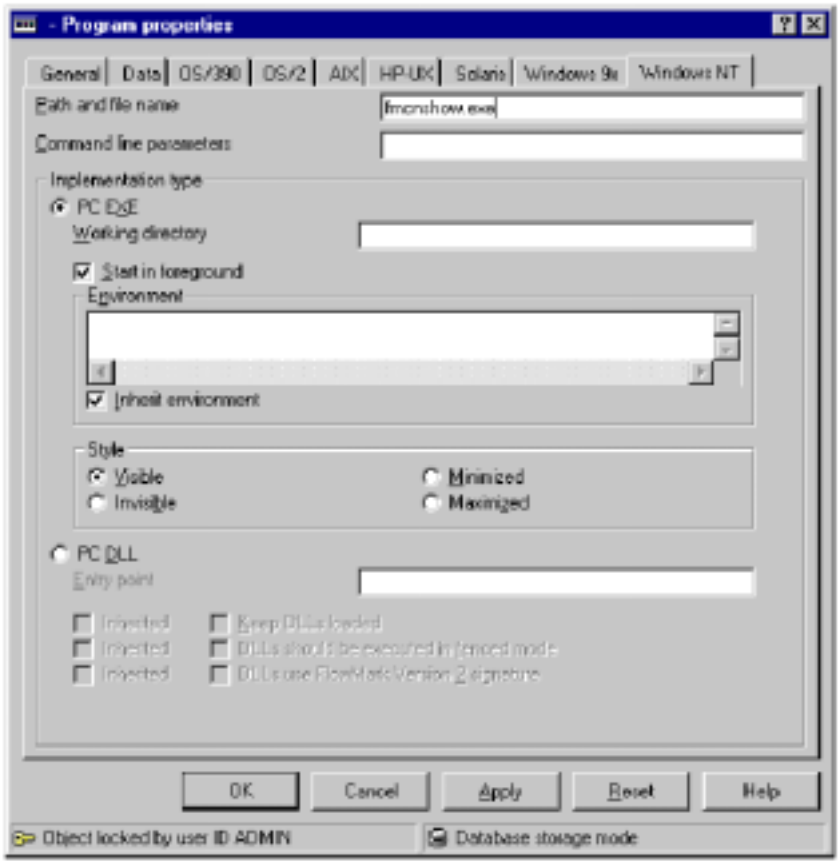

图 4-20 哑元可执行程序

6. 在路径和文件名输入框中键入 fmcnshow.exe。
7. 单击确定(OK)。

确保使用 PATH 环境变量能找到的 fmcnshow 程序。

4.1.4 用所有路径创建模型

现在，我们已经为初始测试定义了所有数据结构和默认执行程序。我们可以开始模拟活动和数据并控制 BUYXYZ 定单过程的流程。

- 选择程序(Processes)标签。

- 右键单击以前定义的**订单流程(Order Process)**，然后选择**图表(Diagram)**。

现在，我们就会有一个空白流程图表并准备开始建模。

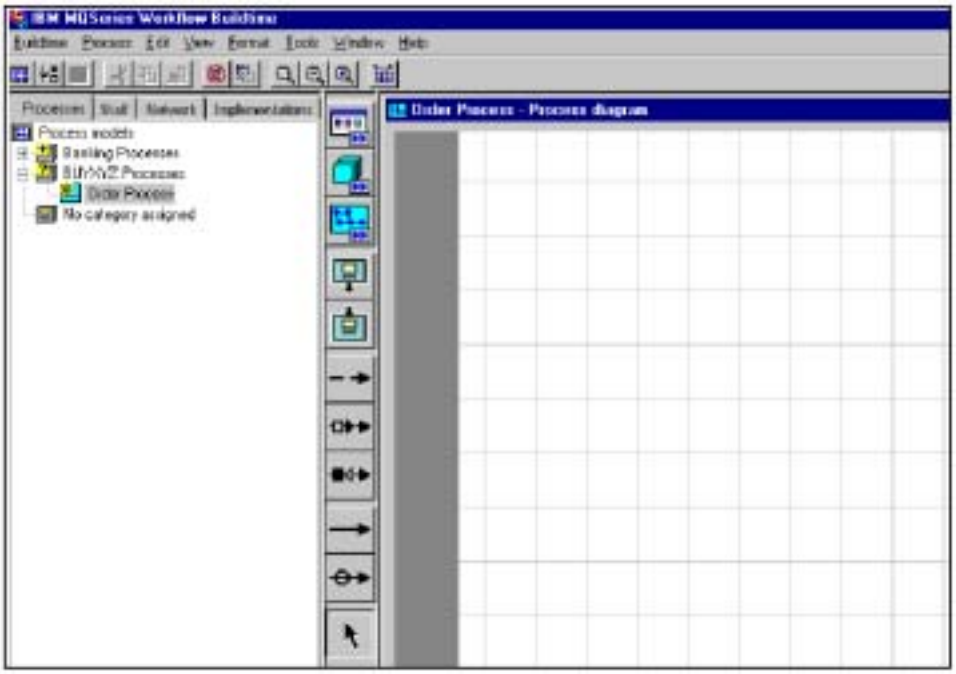

图 4-21 初始构建时窗口

完成流程定义

首先，我们将完成 BUYXYZ 订单流程的定义。

1. 右键单击**订单流程(Order Process)**并选择**属性(Properties)**。
2. 选择**数据(Data)**标签。
3. 在输入**数据结构(Input Data Structure)**边框中单击手电筒图标。

图 4-22 手电筒图标

将出现以前定义的数据结构列表。

4. 选择**订单信息(OrderInfo)**并单击确定(OK)。
5. 选择**启动流程时的数据强制命令(Force prompt for data at process start)**复选框。

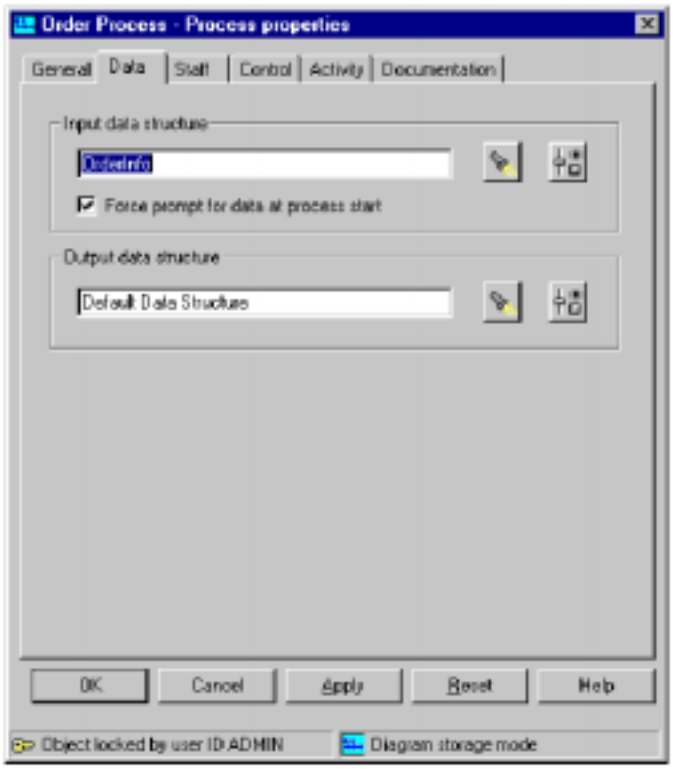

图 4-23BUYXYZ 订单流程的数据结构

6. 单击**确定(OK)**。

活动定义

现在，我们将定义组成 BUYXYZ 订单流程活动。一个程序活动代表业务流程的一个步骤。定义包含关于执行活动所需的资源（人和程序）信息。我们的流程中有八个活动，如第 110 页图 4-1 所示。

1. 从制图工具模板中选择程序活动图标。

图 4-24 程序活动图标

2. 在图表区域单击添加新活动。重复该过程直到我们有八个活动（如图 4-25 所示），每个 BUYXYZ 订单流程步骤对应一个活动。

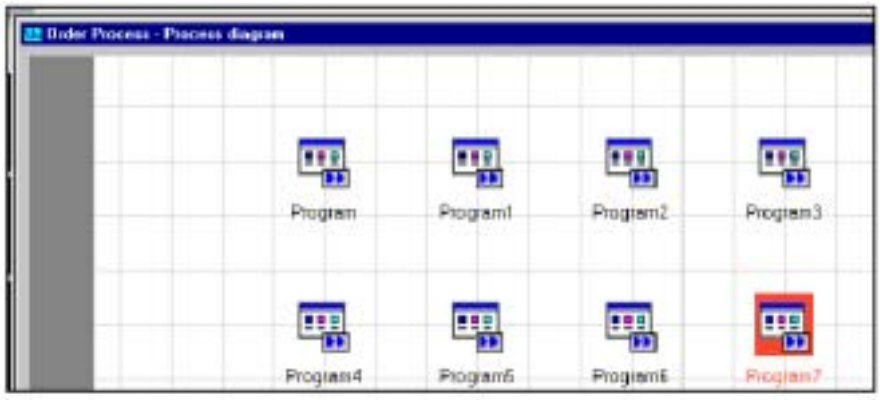

图 4-25 添加初始活动

3. 从制图工具模板中选择指针工具。
4. 右键单击第一个活动并选择**属性(Properties)**，将出现程序活动属性窗口。
5. 键入名称确认客户(ValidateCustomer)。
6. 在程序(Program)边框中单击手电筒图标，将出现查找程序(Find Program)窗口。
7. 从程序列表中选择**哑元(Dummy)**并单击**确认(OK)**，将出现类似图 4-26 所示的窗口。

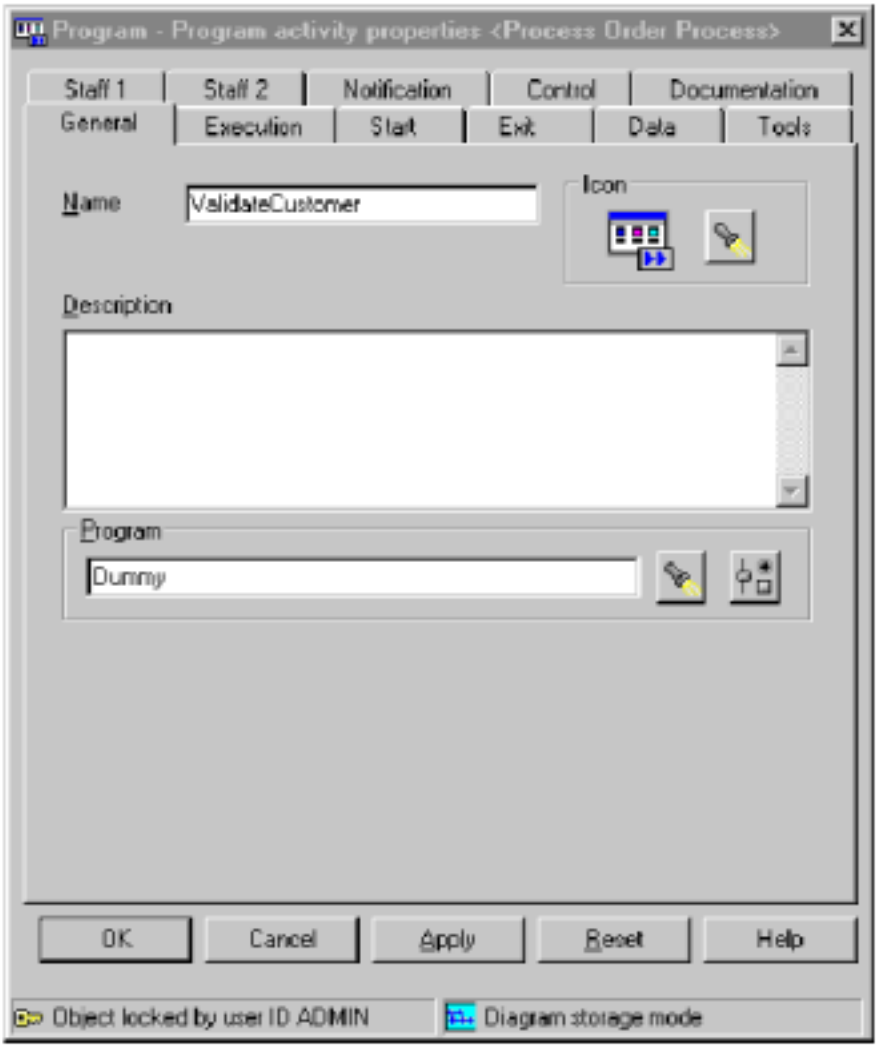

图 4-26 确认客户(ValidateCustomer)活动通用(General)标签

8. 单击**数据(Data)**标签。
9. 单击临近输入手电筒图标，将出现查找(Find)数据结构窗口。
10. 从数据结构列表中选择**客户输入(CustomerInput)**并单击**确认(OK)**。
11. 单击临近输出手电筒图标，将再次出现查找(Find)数据结构窗口。
12. 从数据结构列表中选择**客户有效(CustomerValid)**并单击**确认(OK)**，将出现类似图 4-27 所示的窗口。

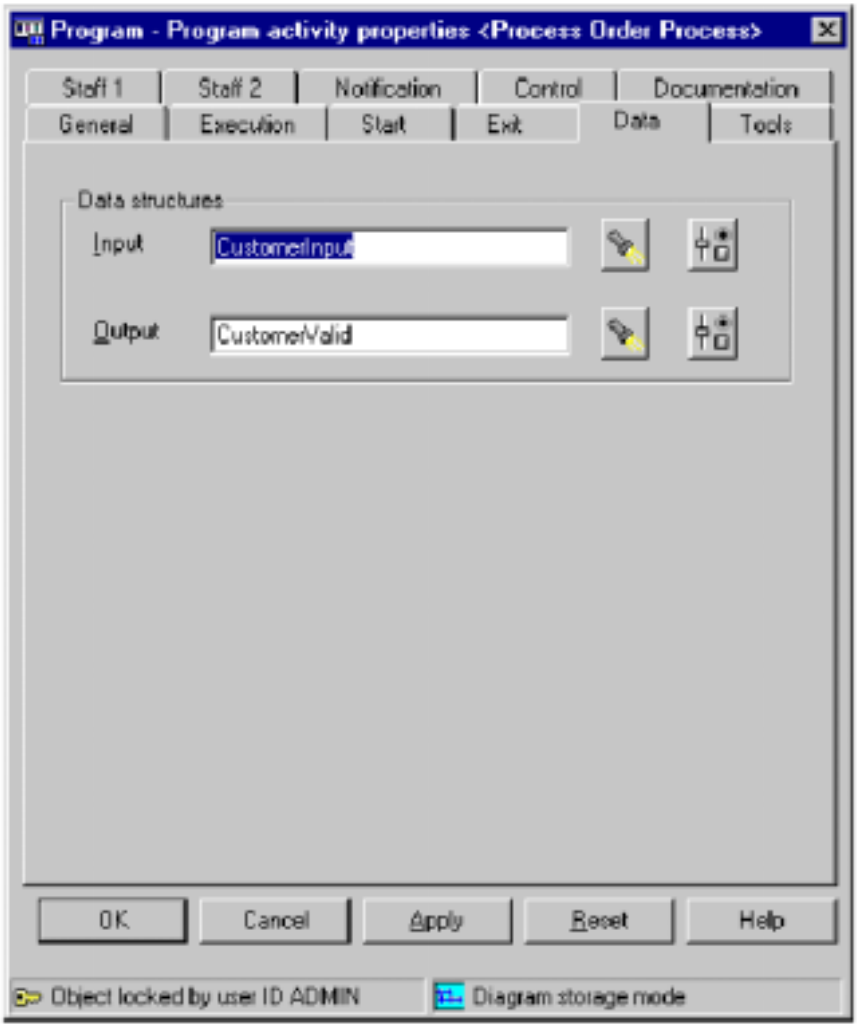

图 4-27 确认客户（ValidateCustomer）活动的数据（Data）标签

为了便于测试，我们将人员解决方案从 **All people** 改变成**流程启动器(Process starter)**。这将阻止为所有用户创建工作单元。

13. 单击 **Staff 1** 标签。
14. 选择**流程启动器(Process starter)**。
15. 单击**确定(OK)**。

这样就完成了有效客户(ValidateCustomer)活动的定义。重复定义每个活动的该过程。每次程序名都是哑元(Dummy)。剩余活动的活动名称和数据容器如下：

表 4-1 剩余活动定义

活动名称	输入容器	输出容器
有效库存(ValidateStock)	库存输入(StockInput)	库存有效(StockValid)
订单条目(OrderEntry)	订单信息(OrderInfo)	订单信息(OrderInfo)
确认订单(ConfirmOrder)	客户确认(CustomerConfirm)	默认数据结构(DefaultDataStructure)
取消订单(CancelOrder)	订单信息(OrderInfo)	默认数据结构(DefaultDataStructure)
存货控制(InventoryControl)	提供输入(SupplyInput)	提供输入(SupplyInput)
提供订单(SupplyOrder)	提供有效(SupplyInput)	提供有效(SupplyValid)
运送订单(ShipOrder)	运送输入(ShippingInput)	默认数据结构(DefaultDataStructure)

完成活动定义之后，出现如图 4-28 所示图表窗口。

图 4-28 已完成的活动定义

对于提供订单(SupplyOrder)活动，我们也将加入在 MQSeries Workflow 版本 3.3 中可用的新截止时间特征。业务规则规定，如果在三天内没有收到供应商的回信，我们将取消定单。为了达到测试目的，我们将该期限设置为一分钟。

1. 双击**提供订单(SupplyOrder)**活动。

2. 选择**退出(Exit)**标签。
3. 在截止时间(Expiration)边框中选择**持续时间(Duration)**。
4. 单击下面的图标设置时间间隔。

图 4-29 时间间隔

将出现设置时间窗口。

5. 在时间输入框中输入 1。
6. 单击**确定(OK)**。

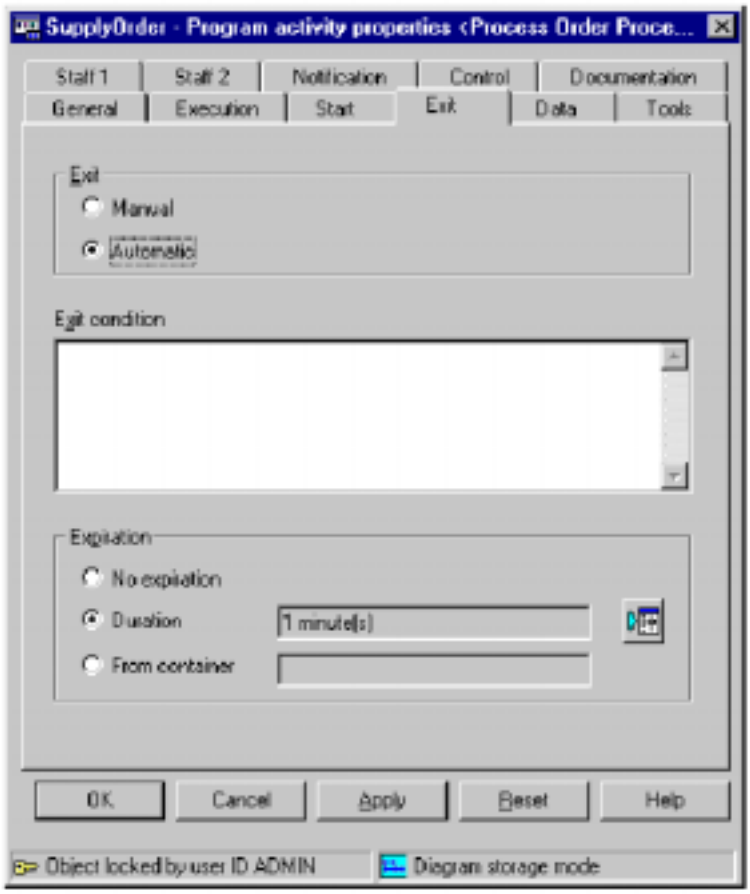

图 4-30 设置截止时间参数

7. 单击**确定(OK)**关闭程序活动属性窗口。

控制流连接器

现在我们将在图表中添加控制流连接器。这将帮助指定活动出现的订单。

1. 在制图工具栏中选择控制连接器图标。

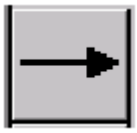

图 4-31 控制连接器图标

2. 单击有效客户(ValidateCustomer)活动一次，然后单击有效库存(ValidateStock)活动。这样将在两个活动之间插入控制连接器并表明只有在第一个活动完成后，第二项任务才可以开始。
3. 重复步骤 2 的操作，在有效库存(ValidateStock)和取消订单(CancelOrder)之间按装控制连接器。现在将显示如图 4-32 所示的图表。

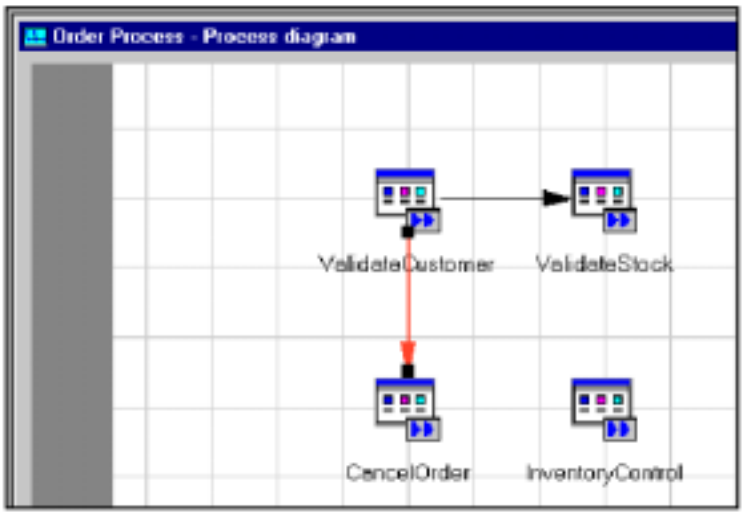

图 4-32 活动之间的控制连接器

该图表意味着一旦第一个活动完成，其他两个活动就可以执行。这并非是我们想要的结果。现在我们将确定决定下一步将执行哪个活动条件。

4. 从制图工具栏中选择指针工具。
5. 双击第一个控制连接器，将出现控制连接器属性窗口。
6. 键入 CustOK=”Y ”作为转换条件。

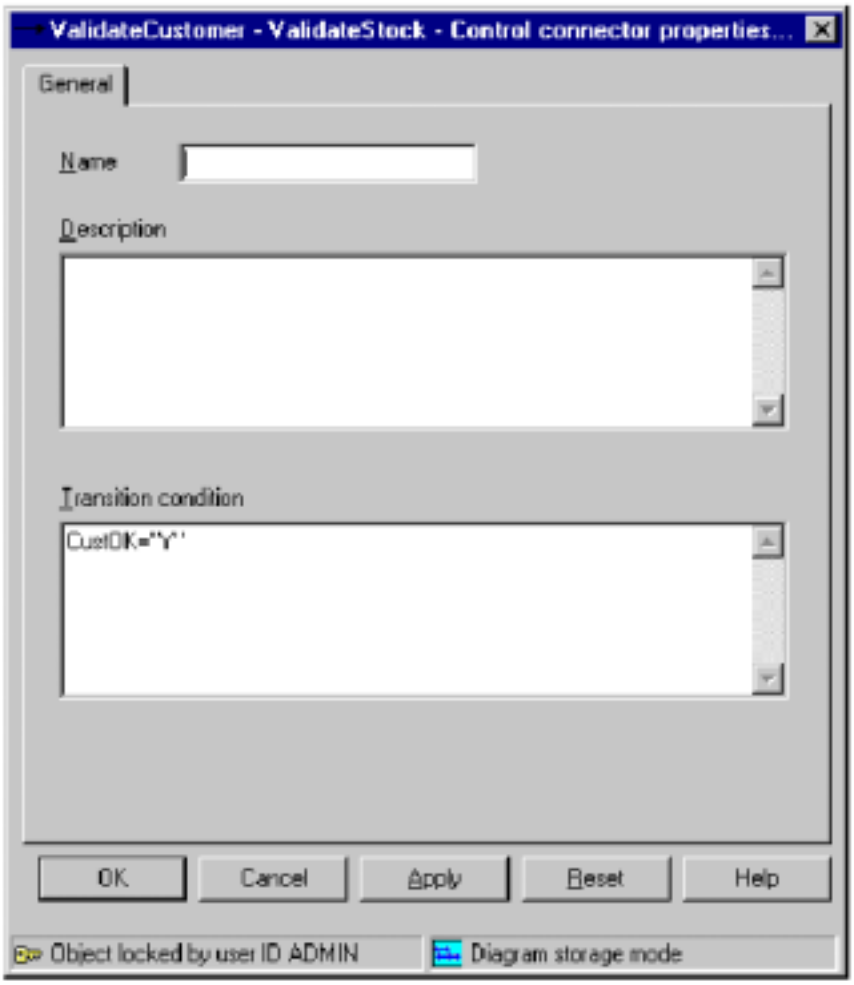

图 4-33 控制连接器条件

7. 单击**确定(OK)**。
8. 除非指定条件为 CustOK=”N ”，才对第二个连接器重复该过程。

现在我们已经确定了执行标准。该标准决定有效客户(ValidateCustomer)完成活动后将执行哪个活动。

为了查看图表窗口中的条件，选择 **View -> Conditions-> Transition conditions**。

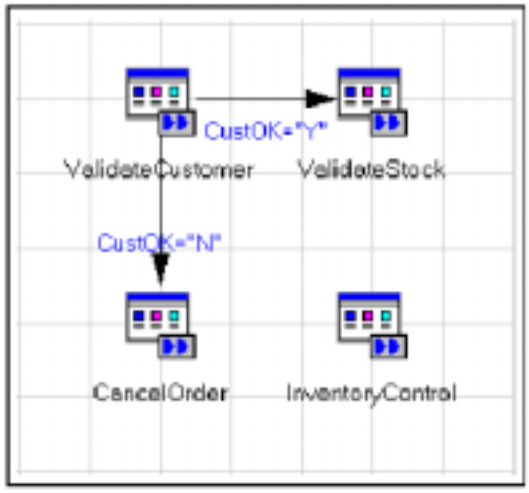

图 4-34 转换条件

添加剩余的控制连接器。完成后，将出现如图 4-35 所示的图表。

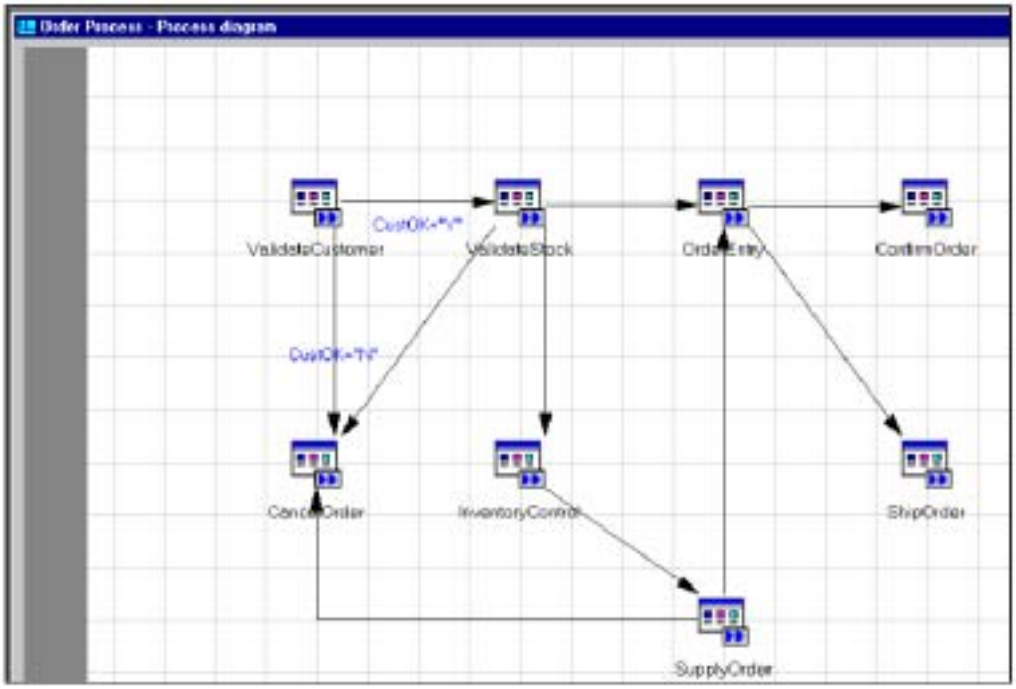

图 4-35 已完成的控制连接器

现在我们添加剩余转换条件。我们已经利用和/或以及保留字段(_RC)和新功能(_State)加入了多条件的使用。要获得转换条件可用选项的完整描述，请参考构建时联机帮助。

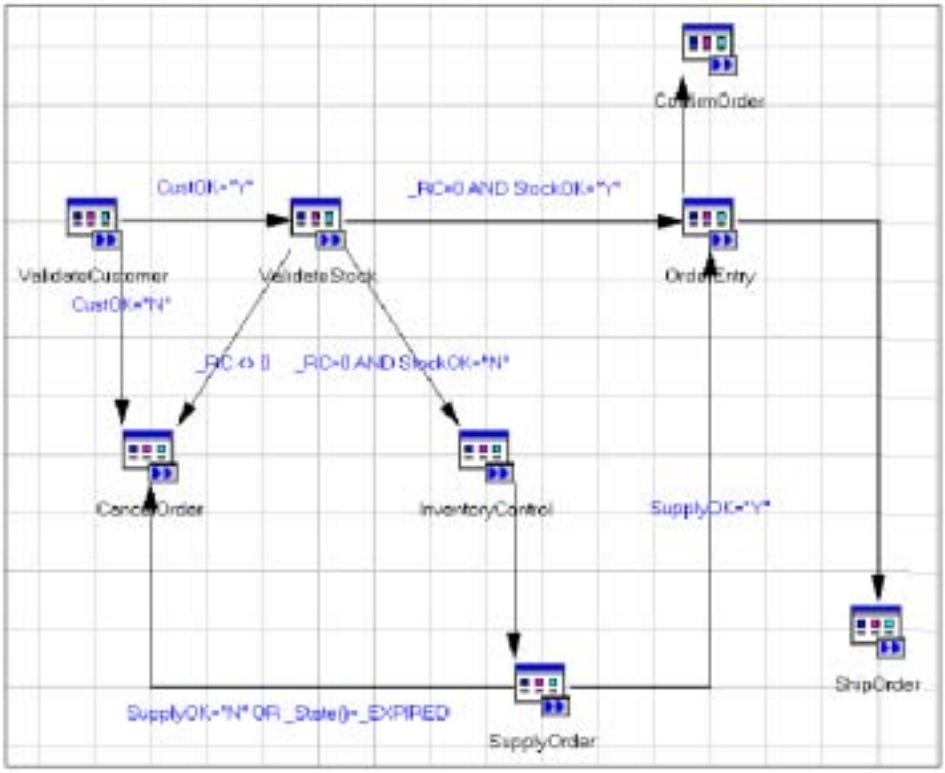

图 4-36 已完成的转换条件

数据流连接器

现在，我们将在该图表中添加数据流连接器。这些连接器确定数据怎样在活动之间传递。当活动完成时，数据从已完成活动的输出容器映射到需要该数据的任意活动的输入容器中。在我们的流程中，订单数据在流程开始时输入并流入流程的输入容器中。为了将数据传递给流程，您必须在图表中添加源节点。

1. 从制图工具栏中选择源节点(Source Node)工具。

图 4-37 源节点图标

2. 在图表的左上角单击鼠标。
3. 再次选择指针工具。
4. 从制图工具栏中选择数据连接器(Data Connector)工具。

图 4-38 数据连接器图标

5. 单击一次源节点(Source Node)，然后单击有效客户(ValidateCustomer)活动。这将在源节点和第一个活动之间插入数据流连接器。

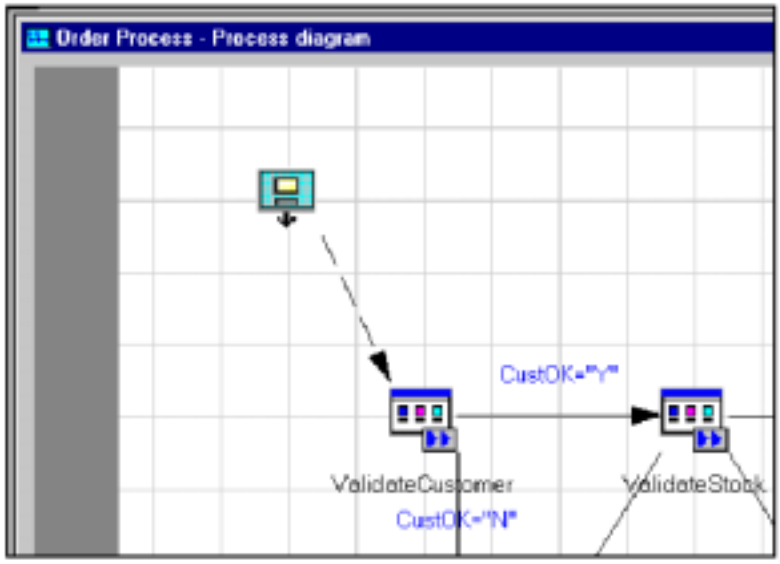

图 4-39 源节点数据流

6. 双击刚刚加入的数据流连接器，将出现如图 4-40 所示的数据映射窗口。

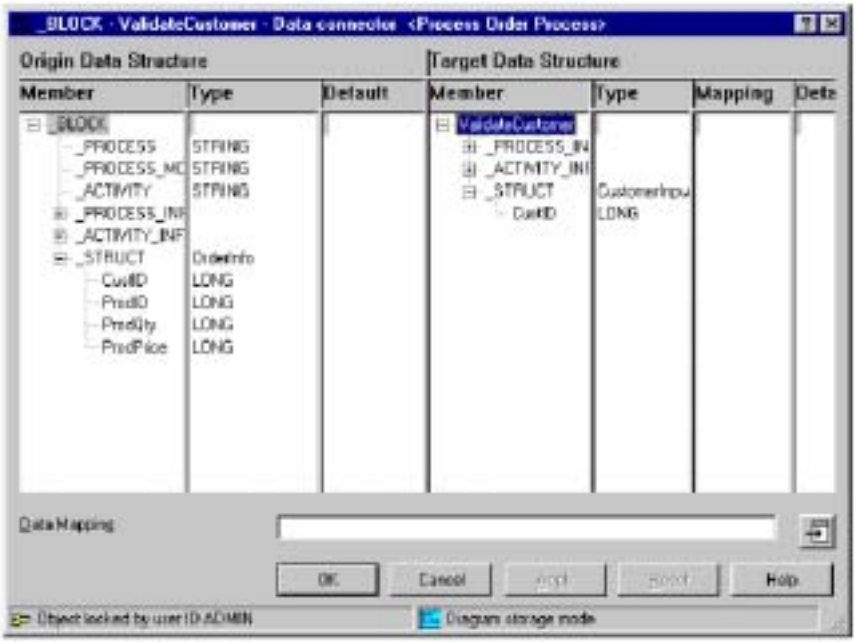

图 4-40 数据映射窗口

7. 展开_STRUCT 树。
8. 单击并抓住源数据结构窗格中 CustID，然后将其拖到目标数据结构窗格中的 CustID 上。

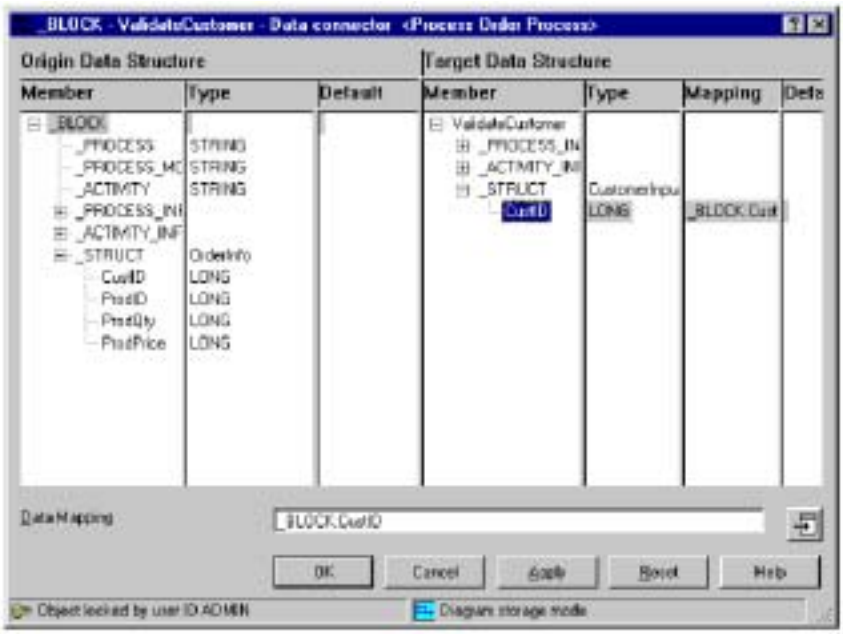

图 4-41 映射 CustID 字段到目标数据结构

9. 单击确定(OK)。

现在，我们将对剩余的数据流连接器重复这些步骤。表 4-2 显示了所有数据连接器和与其相关的映射。表中的每一个 From/To 组合都将有一个数据连接器与之对应。

来自	到达	源	目标
源节点	有效客户	CustID	CustID
源节点	有效库存	ProdID	ProdID
		ProdQty	ProdQty
源节点	取消订单	订单信息*	订单信息
源节点	订单条目	订单信息*	订单信息
源节点	库存控制	ProdID	ProdID
		ProdQty	ProdQty
有效客户	确认订单	CustName	CustName
有效客户	运送订单	CustName	CustName
		CustAddress	CustAddress
有效库存	确认订单	ProdDesc	ProdDesc
有效库存	库存控制	SupplierID	SupplierID
		ProdDesc	ProdDesc
库存控制	提供订单	提供输入*	提供输入
订单条目	运送订单	CustID	CustID
		ProdID	ProdID
		ProdQty	ProdQty
订单条目	确认订单	ProdQty	ProdQty
		ProdPrice	ProdPrice

提示：如果来自(From)和到达(To)数据结构相同，没有必要映射每个字段。通过默认，_BLOCK:_STRUCT 将自动映射。

现在，我们已经完成了建模练习。完整的图表如第 110 页的图 4-1 所示。

在构建时数据库中保存该模型，选择 **Process -> Save**。

通过选择 Process -> Verify 检验该模型。在成功检验该模型之后，准备将其导出到正常正常正常运行时环境。

4.1.5 从构建时环境导出模型到正常正常正常运行时环境

为了执行业务流程，我们必须从构建时环境中导出流定义语言（FDL）文件中的流程定义并将其导入正常正常正常运行时环境。

1. 选择构建时（Buildtime）->导出（Export）。
2. 在导出标记边框（Export flags frame）中选中深导出（Export deep）。
3. 单击确定（OK）。
4. 键入 FDL 文件的文件名并单击保存（Save）。

将在消息窗口中显示消息。一旦导出完成，将显示表明导出成功的消息。现在准备将 FDL 文件导入正常正常正常运行时环境。

1. 打开 DOS 命令提示符窗口。
2. 转到 FDL 文件保存的目录。
3. 采用下面命令导入 FDL 文件：

```
fmcibie -u ADMIN -p password -i xxxxxx.fdl -t –o
```

该导入实用程序将显示消息表明导入操作的结果。正常正常正常运行时服务器立刻知道了新业务流程，然后准备开始测试我们的流量。

4.2 使用哑元程序（dummy program）测试流程和条件

现在我们将能够使用哑元程序 fmcnshow.exe 测试所开发的模型。该程序使我们能够查看输入容器的内容和设置每个活动输出容器的内容。通过修改输出容器的值（我们想要测试的不同情况）可测试不同条件。

4.2.1 采用MQSeries Workflow客户机检验流程

最初，我们采用 MQSeries Workflow 客户机检验流程和条件，其是正常正常正常运行时前端图形用户接口。

1. 选择启动（Start） -> 程序（Programs） -> IBM MQSeries Workflow（IBM MQSeries Workflow） -> MQSeries Workflow 客户机（ MQSeries Workflow Client） – XYZ。

将出现登录窗口。

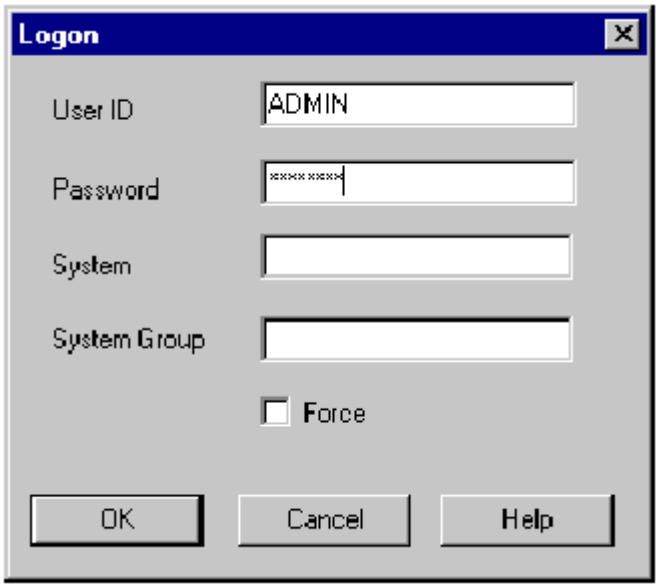

图 4-42 MQSeries Workflow 客户机登录

2. 输入 ADMIN 作为用户 ID，password 作为口令。

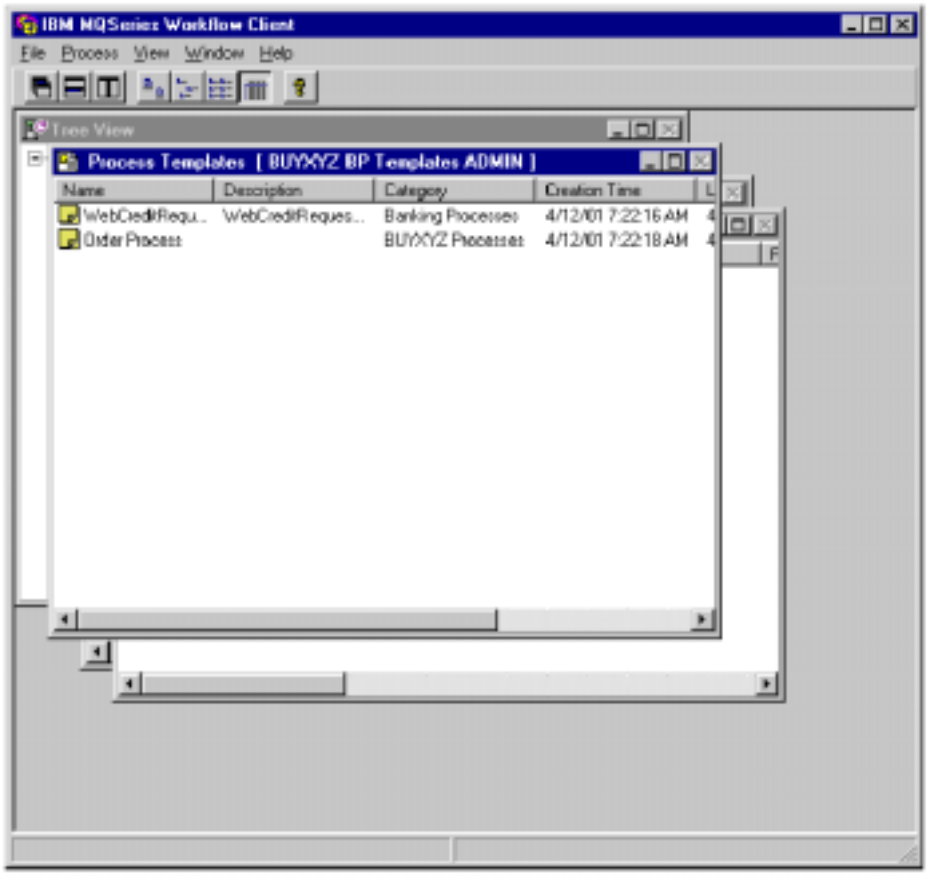

图 4-43 MQSeries Workflow 正常运行时客户机接口

3. 选择流程模板窗口。

4.2.2 检查所有条件和路径

1. 右键单击命令程序（Order Process）并选择创建和启动实例（Create and Start Instance）。MQSeries Workflow 客户机将显示窗口（如图 4-44 所示），允许在该实例的输入数据结构中提供数据。

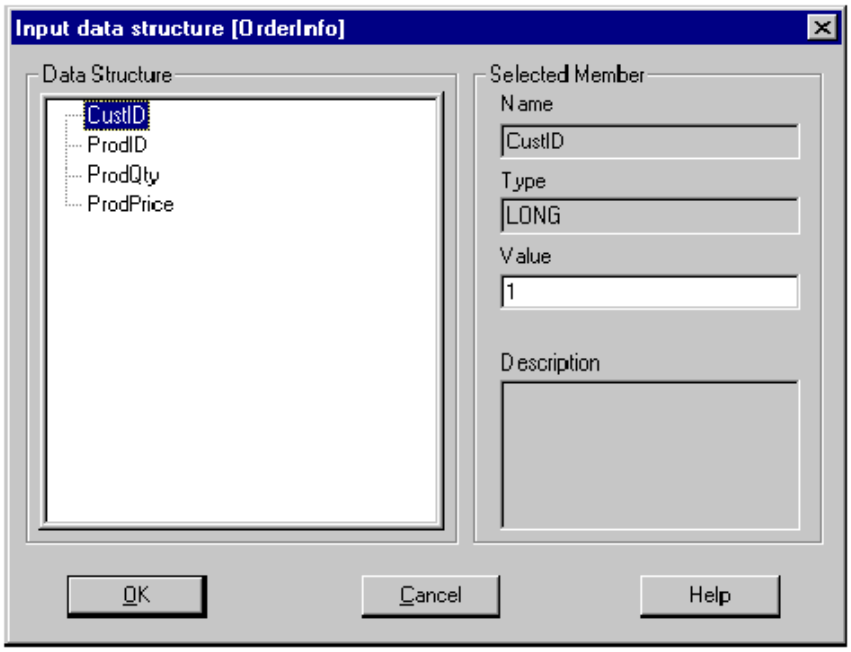

图 4-44 创建并启动——输入订单信息数据

2. 对于第一个测试，我们将使用默认值。
3. 单击确认（OK）。
4. 在 MQSeries Workflow 客户机中选择工作条款窗口。
5. 在窗口中右键单击并选择更新（Refresh）。

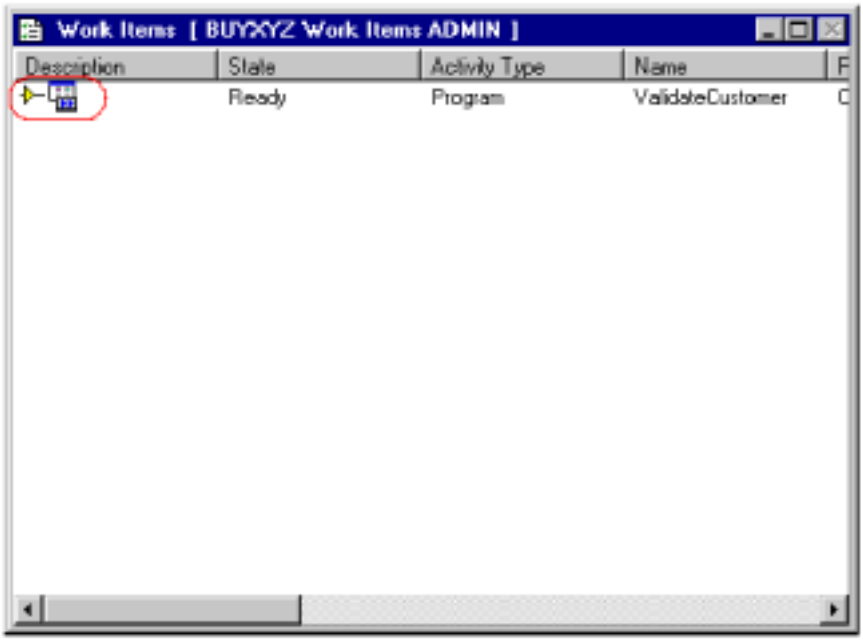

图 4-45 有效客户工作项目

您将在有效客户就绪状态的列表中看到一个项目。

6. 右键单击描述性标题(Description header)下的活动图标（如图 4-45 圈示）并选择启动(Start)。连接该活动的程序 fmcnshow 将启动。其主窗口如图 4-46 所示。

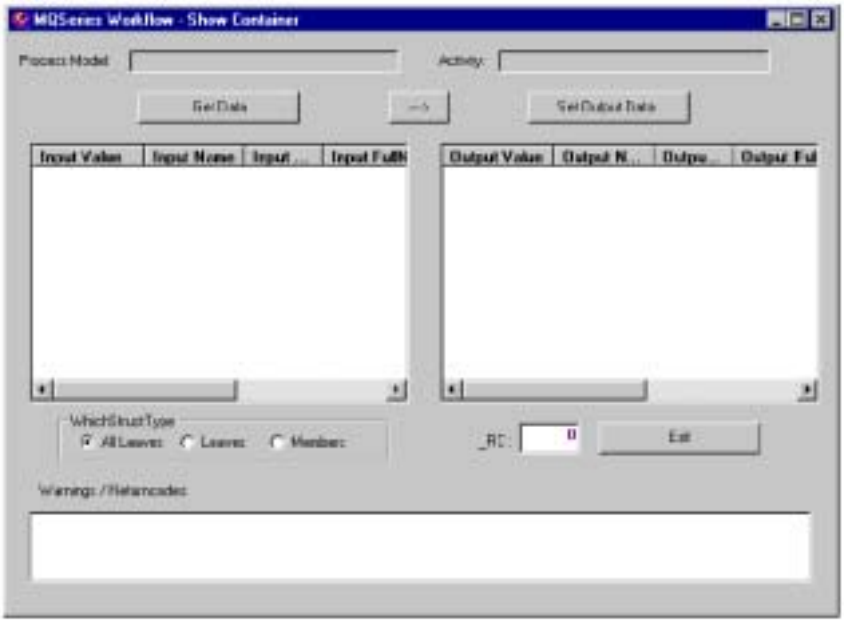

图 4-46 fmcnshow 应用程序窗口

fmcnshow 程序允许您视图输入和输出容器并且为输出容器更新数据。

7. 在输入容器中，单击**获取数据（Get Data）**按钮重新获得该值。选择 **Leaves** 或 **Members** 单选按钮仅能视图容器数据。

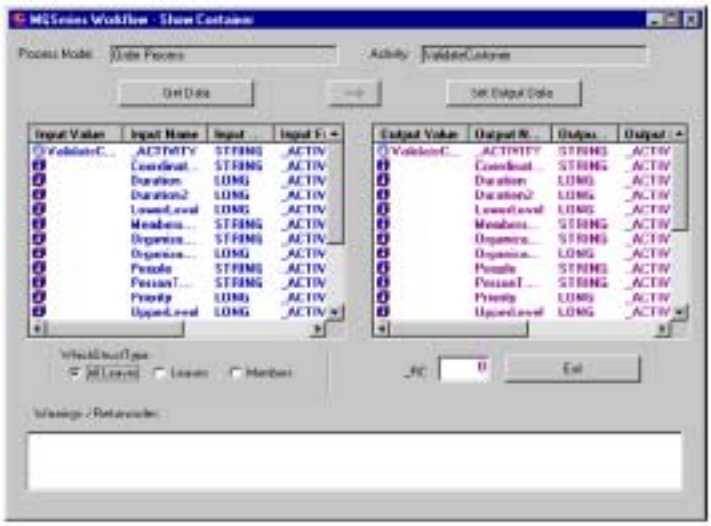

图 4-47 显示输入数据容器

8. 将如图 4-48 所示数据键入输出值列并单击**设置输出数据（Set Output Data）**按钮。

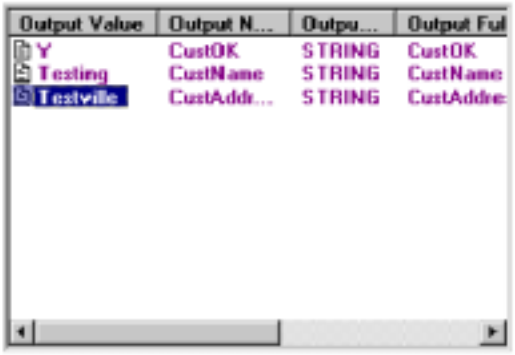

图 4-48 在输出容器中设置数据

9. 单击**退出（Exit）**按钮。

这将完成有效客户活动。右键单击并选择刷新（Refresh）按钮刷新工作项目列表。您将马上看到处于准备状态的有效库存活动项目。

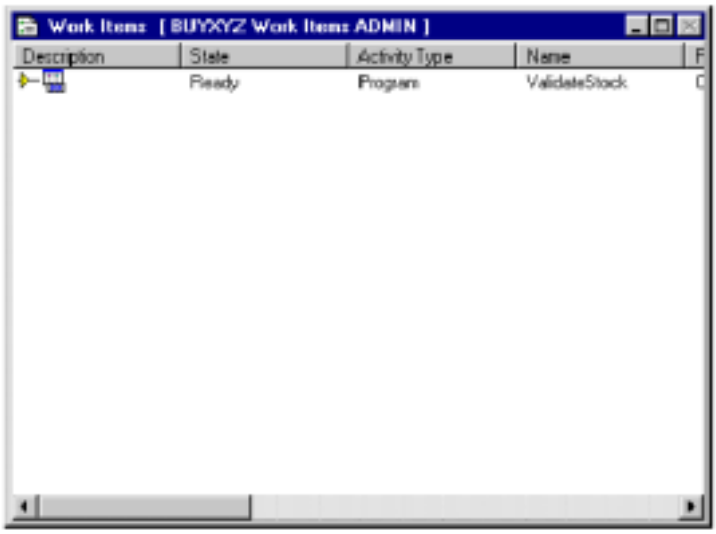

图 4-49 流程中的下一个活动

对于下一次测试，我们将重复先前的步骤以测试异常流程。这可以通过在输出容器的客户确定（CustOK）字段中输入否（N）完成。这将产生取消订单（CancelOrder）活动项目。

使用该方法，您可以测试 BUYXYZ 订单流程的所有控制流和数据流特性。如果希望看到给定流程所执行的步骤，您可以右键单击工作项目或流程实例并选择**监控流程实例（Monitor Process Instance）**。

特别测试考虑

当流程达到确认订单活动时，检验由适当活动传递的客户名和产品描述。

检验名称和地址的 ShipOrder 数据流。

如果在两分钟后还没有执行，检验 SupplyOrder 的活动有效期。如果有效期不存在，您可能需要检查调度服务器报告核实时间间隔。其可在构建时环境中找到，如下：

1. 单击**网络（Network）**标签。
2. 右键单击**域（Domain）**并选择**属性（Properties）**。

3. 选择**服务器（Server）**标签。
4. 单击**完成调度服务器设置（Complete scheduling server settings）**旁边的按钮。
5. 确保检查间隔值设置为 T1M。
6. 也要确保调度服务器设置为自动启动，默认为手动开启。从构建时环境中导出这些变化。在将已经导出的 FDL 文件导入正常运行时环境之后，您将需重启 MQSeries Workflow 服务以激活调度服务器的自动启动功能。

您可以检验调度服务器正在以适当的参数运行如下：

1. 选择**启动（Start）** -> **程序（Programs）** -> **IBM MQSeries Workflow** -> **MQSeries Workflow 管理实用程序（MQSeries Workflow Administration Utility） – XYZ。**

```
-    FMC16006I Administration Utility started.(启动 FMC16006I 管理实用程序)
     System group name(系统组名称)    :[BUYXYZ ]BUYXYZ
     System name(系统名称)           :[BUYXYZ01 ] BUYXYZ01
     Userid(用户 ID)                 :[ADMIN ] ADMIN
     Password(密码) :[]********
=    FMC16110I Receive thread for userID 'ADMIN'at system 'BUYXYZ01'
started.(在系统'BUYXYZ01'启动时，FMC16110I 为用户 ID'ADMIN'接收线程)
-    FMC16301I  UserID  'ADMIN'connected  to  system  'BUYXYZ01'.  （与连接系统相连的
     'BUYXYZ01'FMC16301I 用户 ID'ADMIN'）
      FMC15010I Main Menu（FMC15010I 主菜单）:
        s ...System Commands Menu（系统命令菜单）
        m ...Select Server Menu（选择服务器菜单）
        e ...Errorlog Commands Menu（错误日志命令菜单）
        l ...Systemlog Commands Menu（系统日志命令菜单）
        u ...User Commands Menu（用户命令菜单）
        x ...Exit Main Menu（退出主菜单）
m
    FMC15050I Select Server Menu（FMC15050I 选择服务器菜单）:
      a ...Administration Server Commands Menu（管理服务器命令菜单）
      e ...Execution Server Commands Menu（执行服务器命令菜单）
      s ...Scheduling Server Commands Menu（调度服务器命令菜单）
      c ...Cleanup Server Commands Menu（清除服务器命令菜单）
      x ...Exit Select Server Menu（退出选择服务器菜单）
s
    FMC15053I Scheduling Server Commands Menu（FMC15053I 调度服务器命令菜单）:
      i ...Info（信息）
      u ...Startup（启动）
      d ...Shutdown（关机）
      q ...Query（查询）
      w ...Wait（稍后）
      x ...Exit Scheduling Server Commands Menu（退出调度服务器命令菜单）
```

```
q
-    FMC16220I Scheduling Server is 'active'. (FMC16220I 调度服务器是“活跃的”)
    FMC15053I Scheduling Server Commands Menu (FMC15053I 调度服务器命令菜单):
    i ...Info (信息)
    u ...Startup (启动)
    d ...Shutdown (关机)
    q ...Query (查询)
    w ...Wait (稍后)
    x ...Exit Scheduling Server Commands Menu (退出调度服务器命令菜单)
i
-   Scheduling server settings (调度服务器设置)
-      Check interval (检查时间间隔)                                PT1M
-      Start mode  (启动模式)                                       Immediate
-      Notification check interval (通知检查时间间隔)               PT1M
-      Suspension check interval (中止检查时间间隔)                 PT1H
-      Create notification items threshold (创建通知项目阀值)       10
-      Delete notification items threshold  (删除通知项目阀值)      100
  FMC15053I Scheduling Server Commands Menu (FMC15053I 调度服务器命令菜单):
    i ...Info (信息)
    u ...Startup (启动)
    d ...Shutdown (关机)
    q ...Query (查询)
    w ...Wait (稍后)
    x ...Exit Scheduling Server Commands Menu (退出调度服务器命令菜单)
```

4.2.3 测试使用带有默认Web页的Web客户机的流程

现在，我们将使用 MQ 工作流 Web 客户机执行一些初始测试。Web 客户机将生成默认 Web 页以使您能够更新该活动输出容器中的数值。其允许您执行初始数据流与控制确认的方式在很大程序上与您执行 fmcnshow.exe 程序的方式相同。

登录 Web 客户机

1. 到 http://m23bzzyp/MQWFClient-XYZ1/RTC.html （参见图 4-50）。

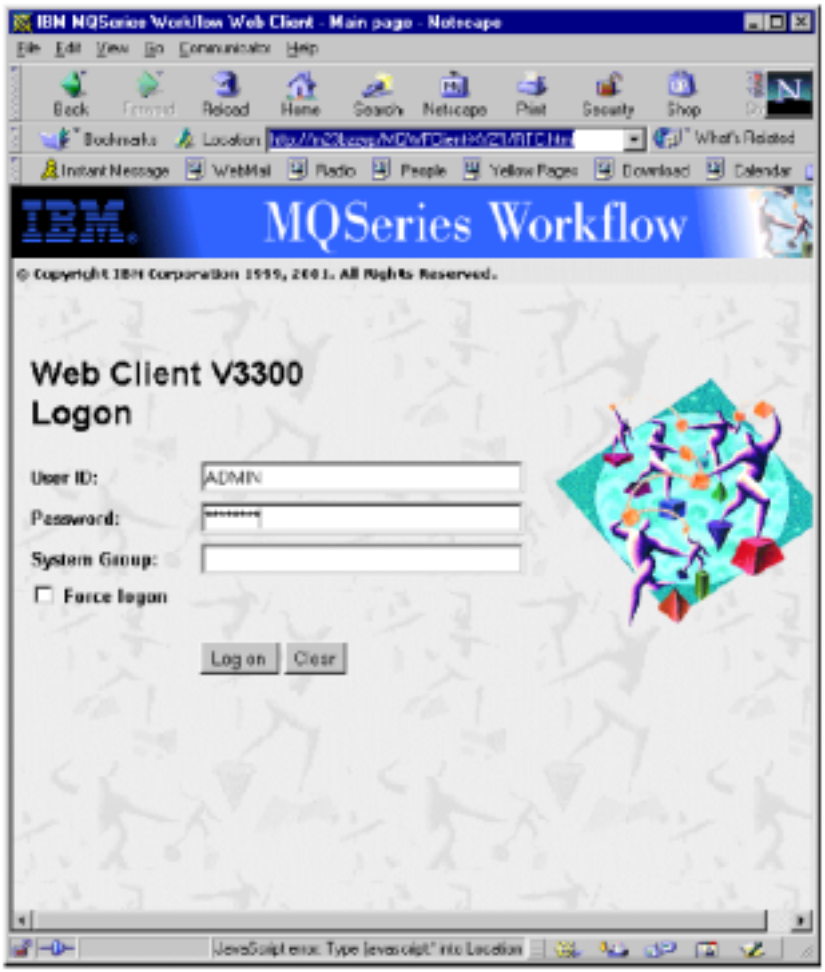

图 4-50 MQ 工作流 Web 客户机登录窗口

2. 输入 ADMIN 作为用户名，password 作为口令。
3. 单击登陆（Logon）按钮。

创建并启动流程实例

一旦登录到 Web 客户机，您将看到工作单元列表。

1. 下拉导航图标（Navigate）列表。
2. 选择 Template Li st - BP Templ ates，您将看到如图 4-51 所示的网页。

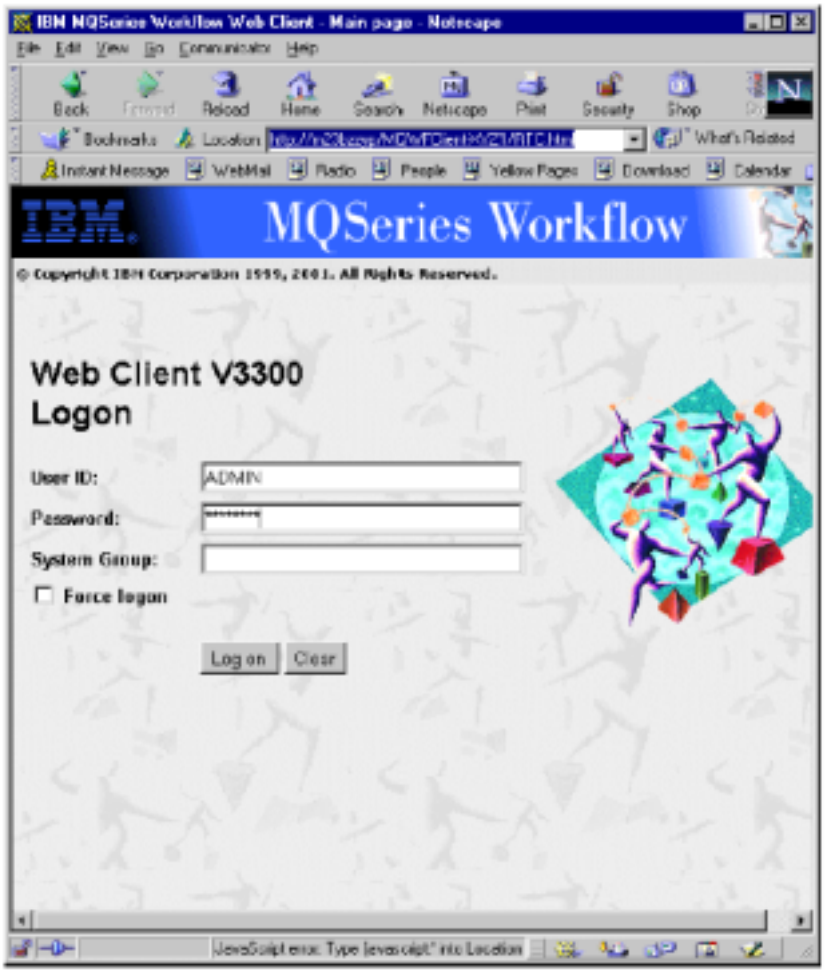

图 4-51 模板列表

3. 单击订单过程模板（Order Process template）下面的创建和启动实例（Create and start Instance）按钮。

图 4-52 创建和启动实例

4. 通过单击**创建和启动实例（Create and start Instance）**接受默认值。

这将启动新的过程实例，然后回到您的工作项目列表。

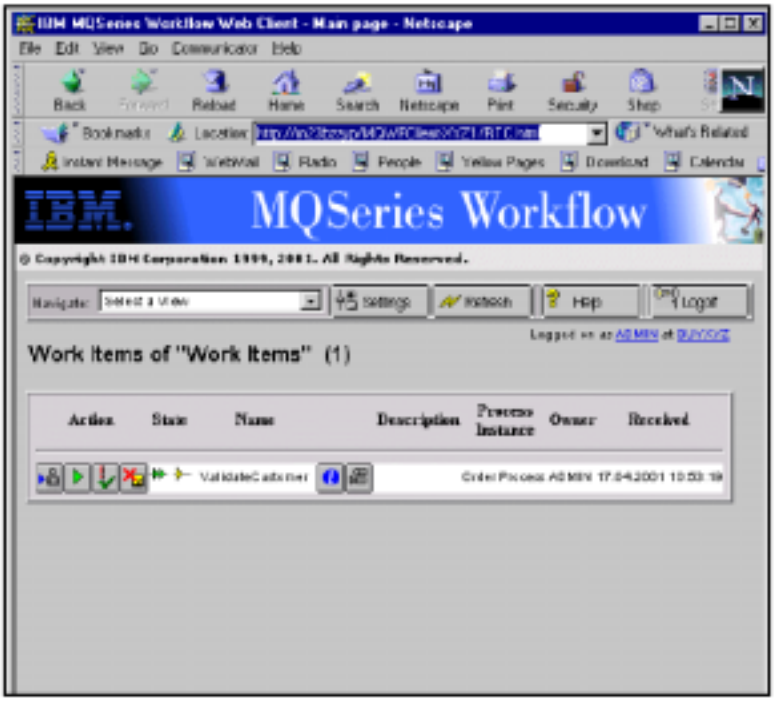

图 4-53 工作项目列表

测试流使用默认页的流程

当通过单击**检验工作项目（Check out Work Item）**按钮选择项目的时候，您将看到一个默认页（如图 4-54 所示）。该默认页显示了带有修改输出容器数据成员选项的输入和输出数据容器。

图 4-54 检验工作项目

一旦输入数据值，单击**登记工作项目（Check in Work Item）**按钮。这将再次回到您的工作列表。为了刷新工作列表，单击**刷新（Refresh）**按钮。

使用 Web 客户机接口，您可使用与工作流正常运行时客户机同样的方法完成初始数据流和控制流确认。

正如看到的那样，您可立刻拥有工作流的基于浏览器接口。在下一节中，我们将展示如何扩大默认 Web 接口。

4.3 开发手动活动的Web接口

使用 Websphere Studio 中的向导开发 Web 接口。但首先，我们需要对工作流作一些小的改变。

4.3.1 修改构建时

现在，我们将修改构建时定义以在执行手动活动时调用特殊的 Web 页。若需要关于定义和修改执行程序的介绍，请参考本书第 125 页的 4.1.3 节“ 创建执行程序 fmcnshow ”。

1. 创建手动活动的执行程序，该手动活动包括：取消命令（CancelOrder）、库存（InventoryControl）、和确认命令（ConfirmOrder）。确保数据标签使用定义程序活动的容器。
2. 修改程序活动以使用新定义的执行程序。
3. 导出已修改的 FDL 文件。

4.3.2 JSP快速应用向导

现在，我们将使用快速应用向导在应用程序中，生成一些定制的手动活动 Web 页。该向导由 SupportPac 提供，您可从以下地址下载：

http://www-4.ibm.com/software/ts/mqseries/txppacs/wa83.html

该 SupportPac 包含关于如何安装快速应用向导，以及如何在 WebSphere Studio 项目中使用它。正确安装之后，向导可通过在 WebSphere Studio 中单击工具（Tool） -> 向导（Wizards） -> 快速应用向导（Rapid deployment wizard）出现，如图 4-55 所示。

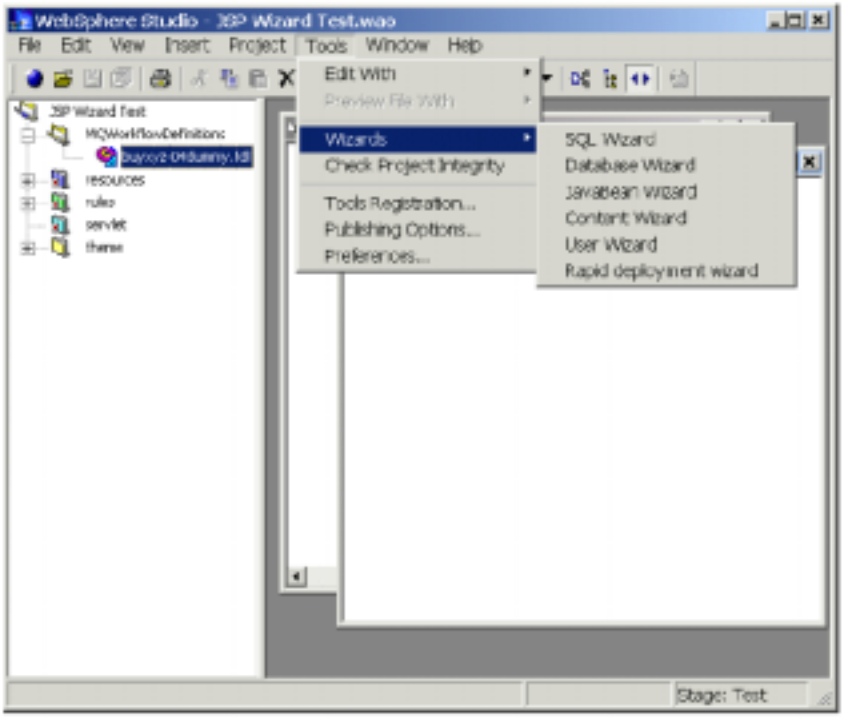

图 4-55 WebSphere Studio—快速应用向导

将 FDL 导入 WebSphere Studio

第一步是将 FDL 文件导入 WebSphere Studio。可通过如下步骤完成：

1. 打开 WebSphere Studio 并创建新工程。
2. 右键单击刚刚创建的工程，然后单击插入（Insert） ->文件夹（Folder）并给出 MQSeries Workflow 资源名。
3. 再次右键单击项目，然后单击插入（Insert） -> 文件（File）。
4. 选择使用现有（Use Existing）标签（如图 4-56 所示）。
5. 浏览已修改的 FDL 文件并选择它。

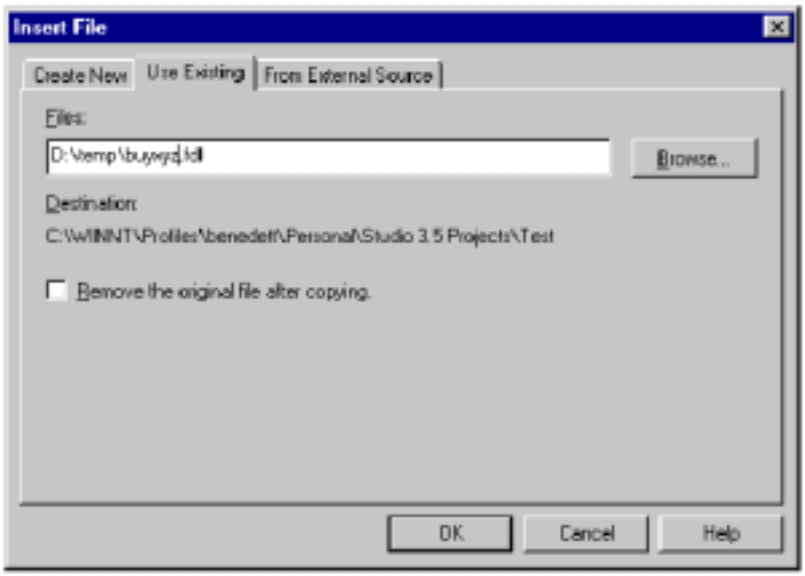

图 4-56 将FDL 文件导入 WebSphere Studio

6. 单击确定（OK）。

使用 JSP 快速应用向导，您可创建 JavaServer 页以创建和开启流程实例。您还可为每个手动活动创建 JavaServer 页。

7. 选择刚刚插入的 FDL 文件。从菜单栏中选择工具（Tools） - > 向导（Wizards） -> 快速应用向导（Rapid deployment wizard）。

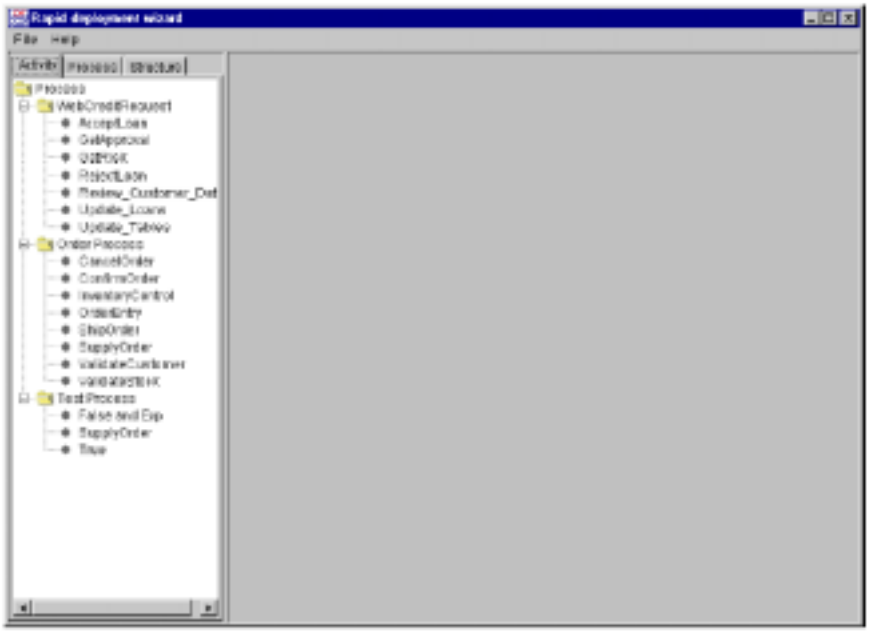

图 4-57 快速应用向导初始窗口

订单过程

首先，我们将创建 JSP 以创建和启动过程实例。该实例使用在订单信息数据结构中定义的字段。

1. 选择**过程（Process）**标签。
2. 双击**订单过程（Order Process）**。

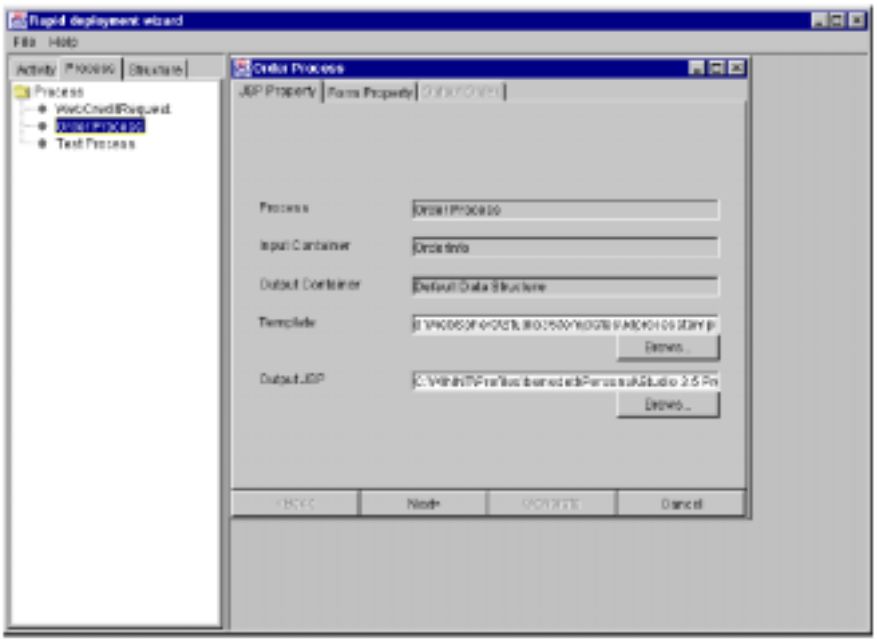

图 4-58 JSP 属性

将出现 JSP 属性窗口，您可以在其中指定模板文件的位置，以及您喜欢的输出 JSP 的位置。我们暂时保持默认设置。

3. 单击**下一步（Next>）**。
4. 选择第一个数据成员，本案例中为 CustID。

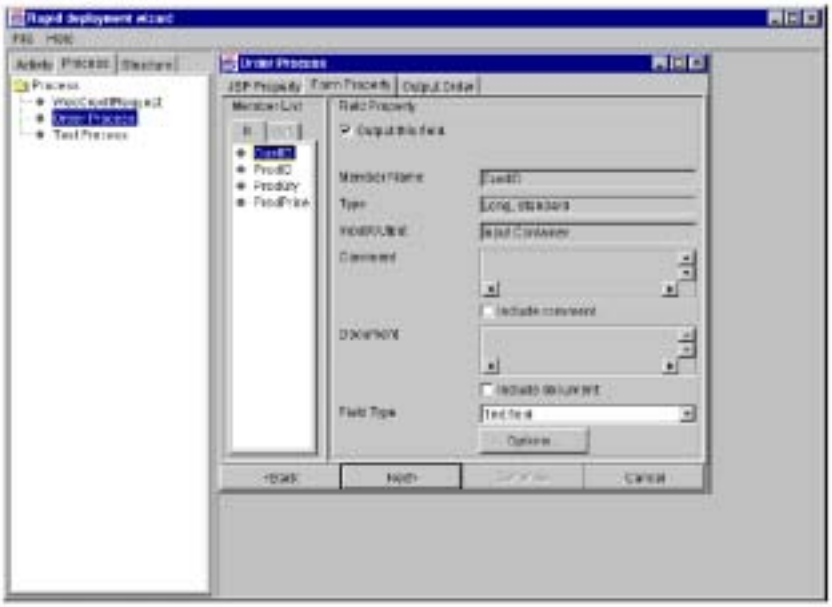

图 4-59 表格属性

5. 在此您可以指定字段特性。其它信息可通过单击**选项（Options）**指定。
6. 单击**下一步（Next>）**。

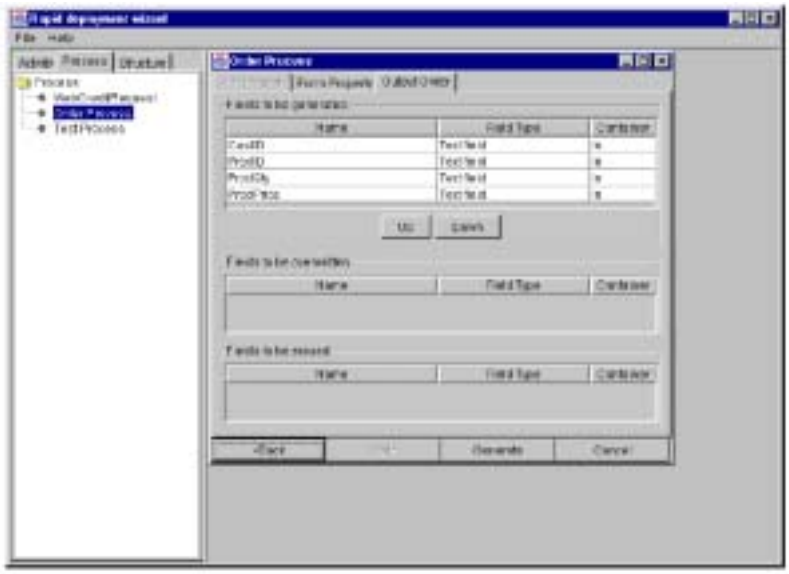

图 4-60 输出订单

在此您可以改变订单字段。

7. 单击**生成（Generate）**。

将提示您核实选择，单击是（Yes） ->确定（OK）。

现在创建和启动流程实例的 JSP 已保存在 WebSphere Studio 中的过程文件夹中。

取消订单（CancelOrder）

下一步，我们将使用快速应用向导创建取消订单（CancelOrder）活动的 JSP。这些步骤与以前练习非常相似。

1. 选择活动（Activity）标签。
2. 双击取消订单（CancelOrder）活动。

您将马上看到类似于本书第 163 页图 4-58 所示的 JSP 属性窗口。

3. 单击下一步（Next>）。

您将马上看到一个类似于 164 页图 4-59 所示的表格属性窗口。

4. 选择每个字段并检查选定输出该字段（Output this field）框。
5. 单击下一步（Next）>。
6. 单击生成（Generate）。
7. 单击是（Yes） -> 确定（OK）。

现在，取消订单（CancelOrder）活动的 JSP 已保存在 WebSphere Studio 中的程序文件夹中。

库存控制（InventoryControl）

重复在取消订单一节中确定的步骤。

确认命令（ConfirmOrder）

重复在取消订单一节中确定的步骤。

一旦生成了 JSP，就可进一步根据需要增强以包含使用 WebSphere Studio 的其他功能。

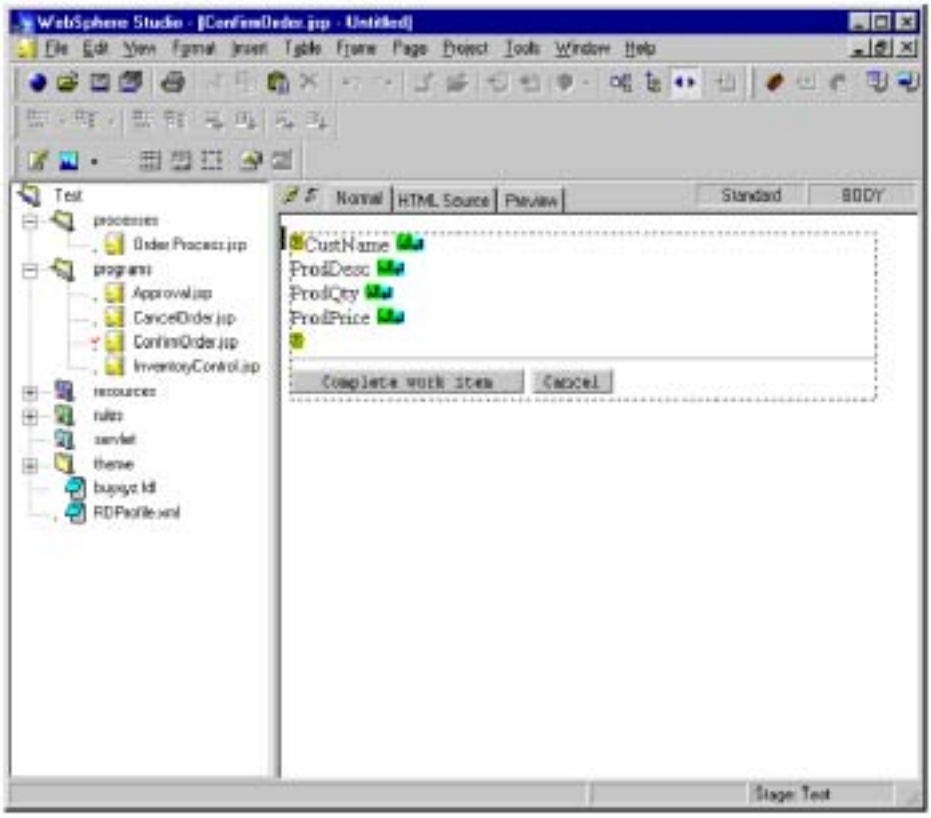

图4-61 在 WebSphere Studio 中查看 JSP

4.3.3 在Web服务器中应用JavaServer页

1. 复制文件到 MQ Workflow Web 客户机程序中(\mqwf\cfgs\xyz1\WebClient\webpages\processes)和程序(\mqwf\cfgs\xyz1\WebClient\webpages\programs)目录中。
2. 将已修改的 FDL 文件导入 MQ 工作流正常运行时环境。

4.3.4 采用Web客户机测试手动行为

在本书第 153 页的 4.2.3 节“ 测试使用带有默认 Web 页的 Web 客户机流程 ”中概述了以下步骤，重复 Web 客户机测试以检验生成的窗体运行正常。

在创建和启动新订单过程模板实例后，您将看到类似于图 4-62 所示的窗口。

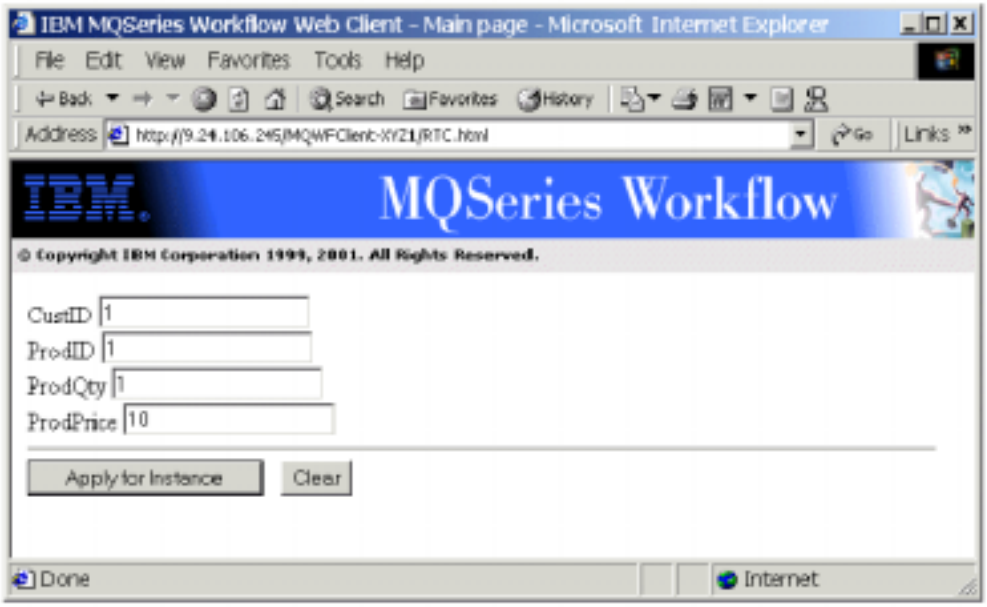

图 4-62 浏览已生成的 JSP 接口

当然，这并非人们所期待的富有想象力的接口。然而，从技术观点来说，唯一要做的事情只是加入更多的图片而已。JSP 包含了与 MQSeries Workflow 交互，以及运行业务流程所需的每件事物。

4.4 UPES活动的定义

在 MQSeries Workflow 的以前版本中，已经介绍过用户定义程序执行服务器（UPES）活动。基本上，UPES 活动即 MQSeries Workflow 在您选择的 MQSeries 队列中写 XML 消息。然后，该消息将由非 MQSeries Workflow 程序处理。它可以是用户写的程序或是 MQSeries Integrator 中的消息流。

UPES 活动可是同步的或异步的。对于同步的 UPES，意味着在收到回复之前，流程实例不能继续。在实例 4-1 中，您将看到该消息的格式化实例。在本书的第 179 页的第五章“ 在 MQSeries Integrator 中执行活动 ”中，我们将测试使用此类消息并为 MQSeries Workflow 构建回复消息的几个消息流。

实例 4-1 使用 MQSeries Workflow 的 XML 消息

```
<WfMessage>
<WfMessageHeader>
<ResponseRequired>Yes</ResponseRequired>
</WfMessageHeader>
<ActivityImplInvoke>

<ActImplCorrelID>RUEAAAABAE1AdQAAAAAAAAAAAAAACgAAAAEAUIAAAAAAAAAAAAAAAAAAKQQAAAAIADsASAAAAA
AAAAA
BF</ActImplCorrelID>
<Starter>ADMIN</Starter>
<ProgramID>
<ProcTemplID>AAAAAQBQgAAAAAAAAAAAAA==</ProcTemplID>
<ProgramName>UPES_Program</ProgramName>
</ProgramID>
<ImplementationData>
<ImplementationPlatform>WindowsNT</ImplementationPlatform>
<ProgramParameters></ProgramParameters>
<ExeOptions>
<PathAndFileName>fmcnshow.exe</PathAndFileName>
<InheritEnvironment>true</InheritEnvironment>
<StartInForeGround>true</StartInForeGround>
<WindowStyle>Visible</WindowStyle>
</ExeOptions>
</ImplementationData>
<ProgramInputData>
<_ACTIVITY>ShipOrder</_ACTIVITY>
<_PROCESS>Order Process$AAAAAQBNQHUAAAAAAAAAAA==</_PROCESS>
<_PROCESS_MODEL>Order Process</_PROCESS_MODEL>
<ShippingInput>
<CustID>1</CustID>
<ProdID>1</ProdID>
<ProdQty>150</ProdQty>
<ProdPrice>1</ProdPrice>
<CustName>Tester</CustName>
<CustAddress>100 Testing Lane</CustAddress>
</ShippingInput>
</ProgramInputData>
<ProgramOutputDataDefaults>
<_ACTIVITY>ShipOrder</_ACTIVITY>
<_PROCESS>Order Process$AAAAAQBNQHUAAAAAAAAAAA==</_PROCESS>
<_PROCESS_MODEL>Order Process</_PROCESS_MODEL>
```

```
<DefaultDataStructure>
</DefaultDataStructure>
</ProgramOutputDataDefaults>
</ActivityImplInvoke>
</WfMessage>
```

现在，让我们来研究如何在工作流中实现和配置 UPES 活动。

4.4.1 有效客户（ValidateCustomer）活动

现在，我们将用 UPES 活动代替有效客户活动中的哑元程序。该 UPES 活动将调用 MQSeries Integrator 消息询问客户数据库。其可通过将 XML 消息插入启动消息流的队列中完成。一旦完成消息流，回复消息将传回 MQ 工作流以表明活动完成。

创建 UPES

在构建时环境中：

1. 选择**网络（Network）**标签。
2. 展开**域（Domain）**树直看到系统为止。
3. 右键单击第一个系统并选择**新用户程序执行服务器（New User-Defined Program Execution Server）**。
4. 在如图 4-63 所示的窗口中，输入 UPES 名。

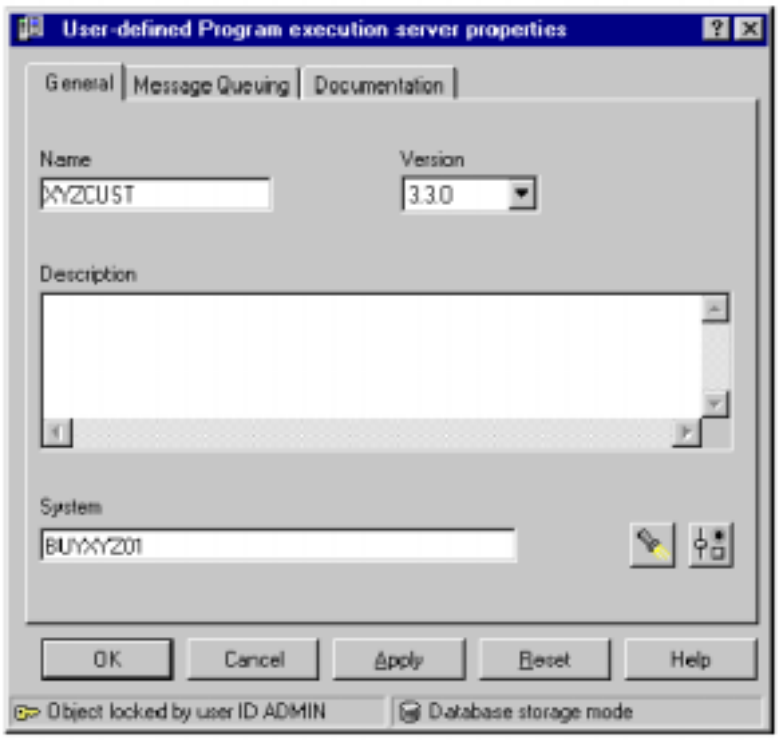

图 4-63 UPES 定义

5. 选择**消息查询（Message Queuing）**标签（如图 4-64 所示）。
6. 输入队列名。因为我们正在 MQSeries 配置中使用 MQSeries 集群，所以我们将不填写队列管理器名称字段。该队列将由两个 MQSeries Integrator 代理队列管理器提供。基于可用性和负载均衡的 MQSeries 集群技术将路由该消息到两个队列实例之一。

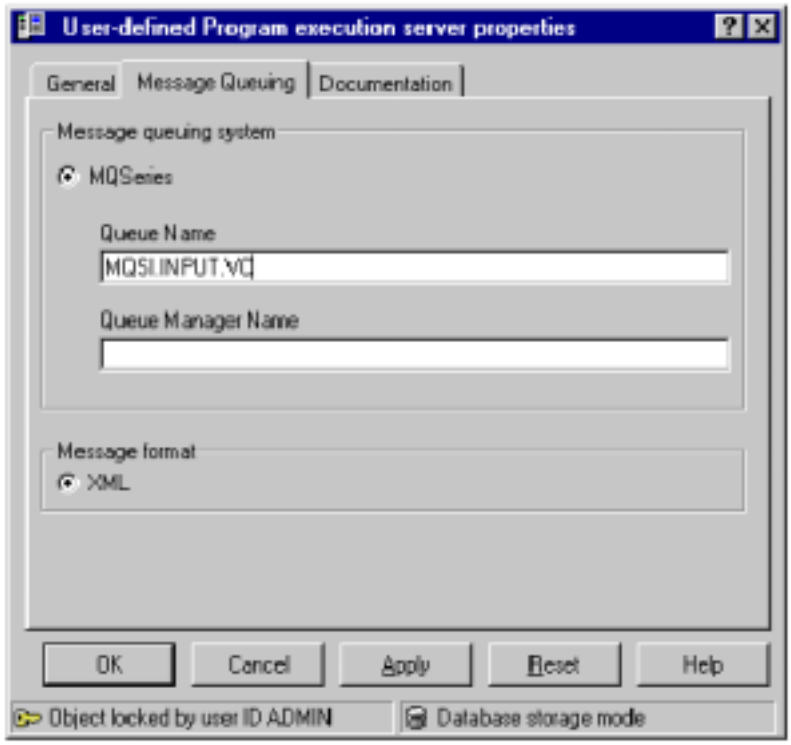

图 4-64 UPES 队列定义

7. 单击确定（OK）。

我们将马上在服务器列表中看到新的 UPES。

创建 UPES 执行程序

从构建时环境中：

1. 选择执行（Implementations）标签。
2. 右键单击程序（Programs）并选择新程序（New Program），将出现程序属性窗口（如图 4-65 所示）。

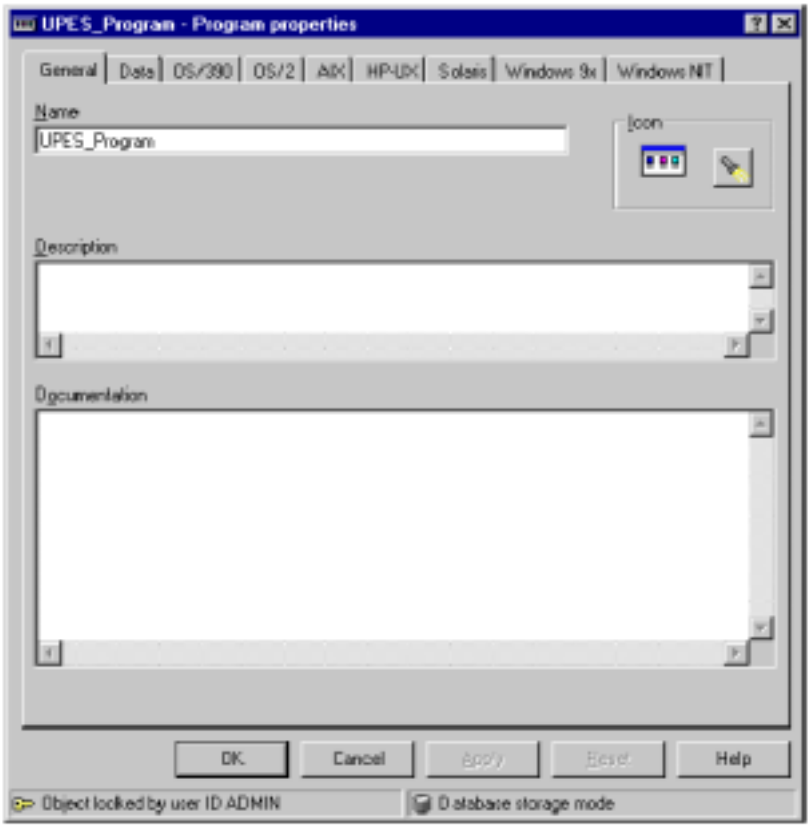

图 4-65 创建 UPES 执行程序

3. 输入程序名和可选描述。
4. 选择数据（Data）标签（如图 4-66 所示）。

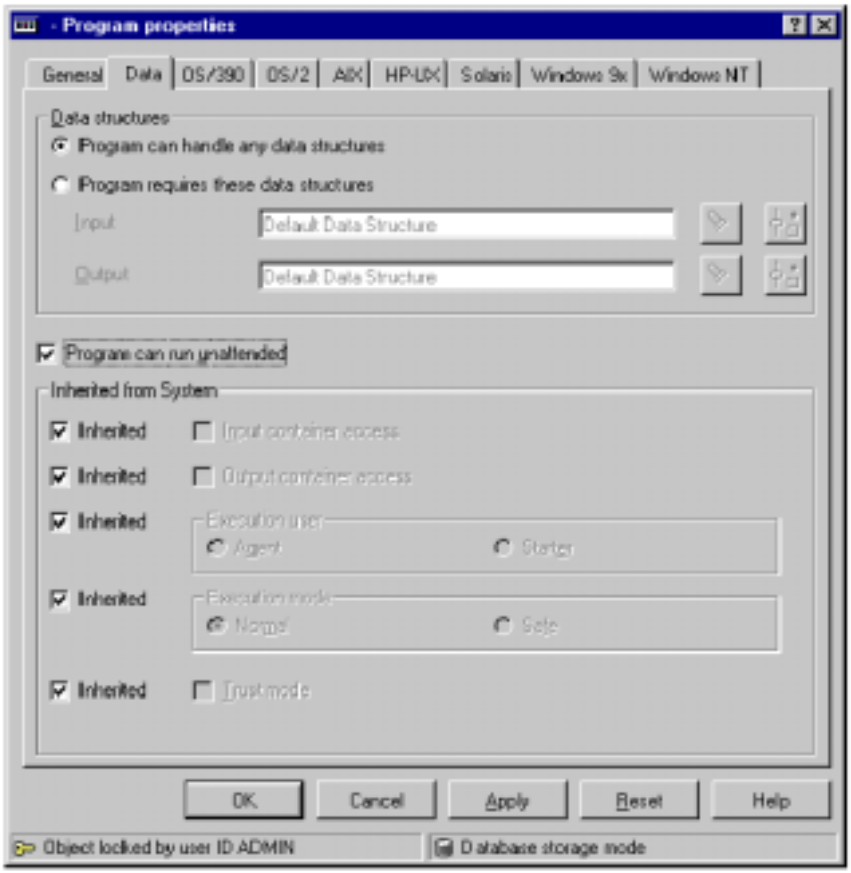

图 4-66 UPES 程序数据规范

5. 选择**程序处理所有数据结构（Program can handle any data structures）**。
6. 选定**程序自动运行（Program can run unattended）**。
7. 为了避免出现警告消息，您可以选择 **Windows NT** 标签并输入 fmcnshow.exe 作为路径和文件名。
8. 单击**确定（OK）**。

该执行程序将由所有 UPES 应用程序调用的自动活动使用。

更改活动定义

1. 打开有效客户（ValidateCustomer）活动的活动属性窗口（如图 4-67 所示）。

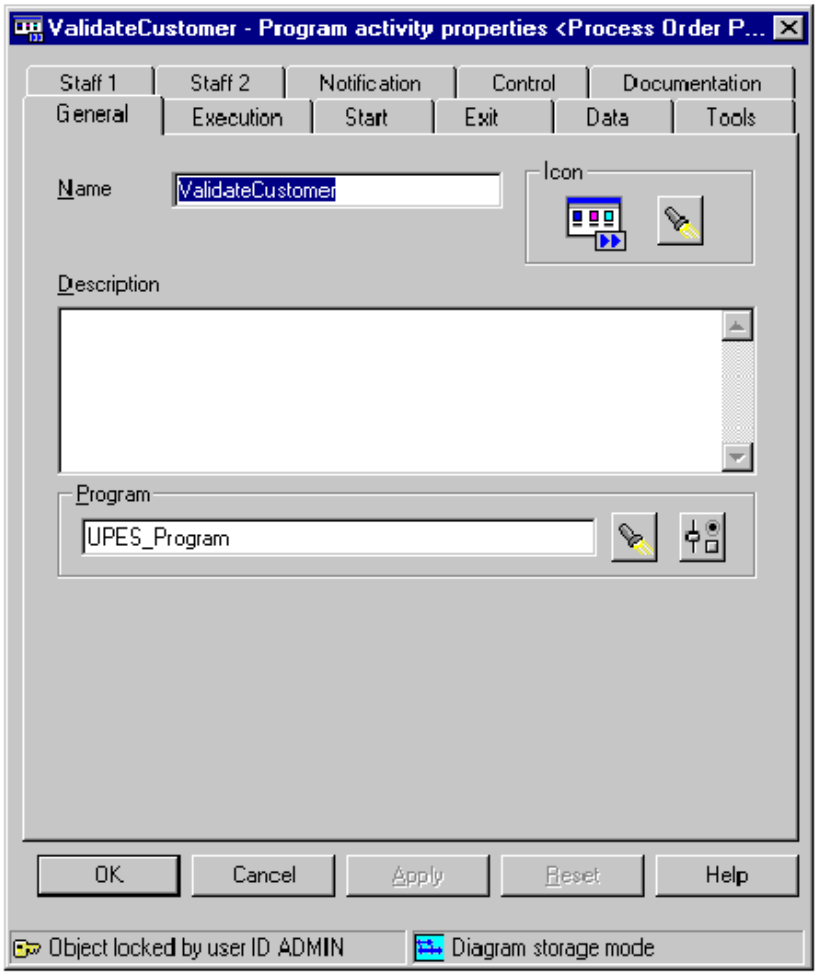

图 4-67 有效客户通用标签

2. 单击程序框中的手电筒图标。
3. 从列表中选择 **UPES_Program**。
4. 选择**执行（Execution）**标签。

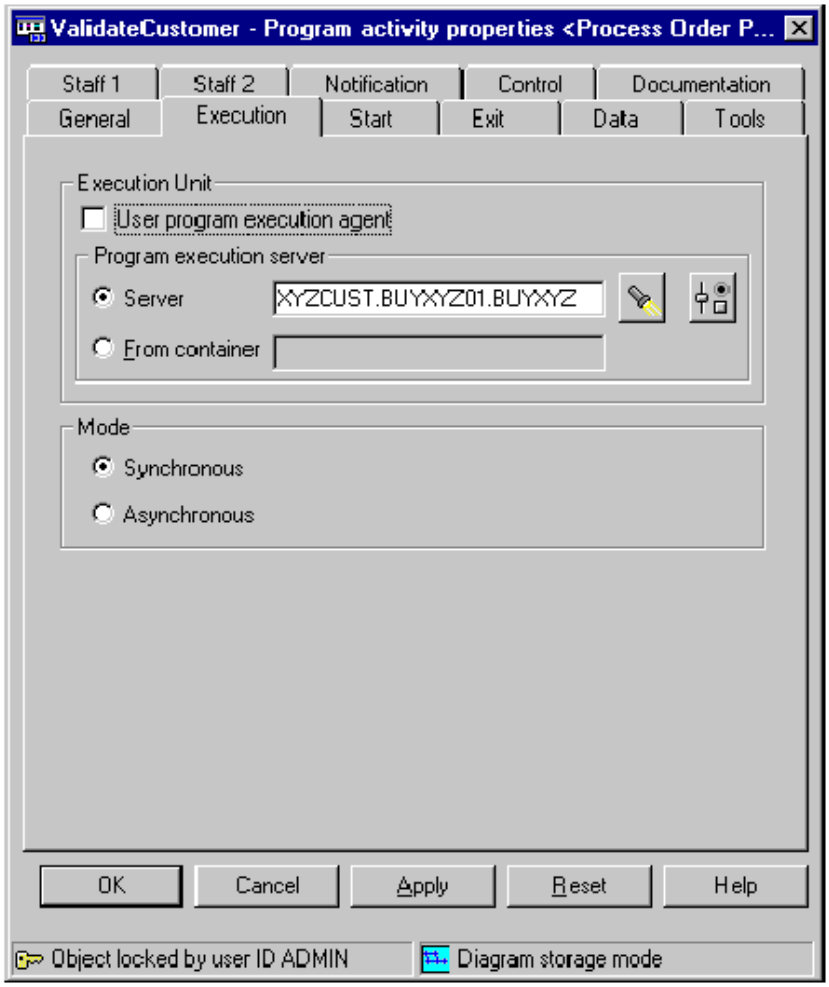

图 4-68 有效客户执行标签

5. 未选中用户程序执行代理程（User program execution agent）框。
6. 在程序执行服务器（Program execution server）边框中单击手电筒图标，将出现搜索程序执行服务器窗口。
7. 选择先前定义的 XYZCUST UPES。
8. 选择启动（Start）标签，您将看到如图 4-69 所示的窗口。

图 4-69 有效客户启动标签

9. 选择自动（Automatic）。
10. 单击确认（OK）。

这从工作流角度完成了有效客户活动的定义。MQSeries Integrator 消息流将是下一章的主题。如果确定该模式为同步的，MQ 工作流将在指定队列中插入 XML 消息并等待与该活动相关的 ID 匹配的回复以表明该活动已完成。

4.4.2 有效存储活动

现在，我们将创建类似活动以创建库存确认 UPES。该活动将由查询数据库的 MQSeries Integrator 消息流和执行计算组成。然而，从工作流角度来看，这些步骤是相同的。本书第 170 页的 4.4.1 节“ 有效客户活动 ”概述了以下步骤，我们只是指定了不同队列名而已。

4.4.3 供应订单活动

对于该活动，XML 消息将再次路由到 MQSeries Integrator 中。该消息流将预备消息，该消息是准备发送给供应商的。

使用 MQSI.INPUT.SO 作为输入队列创建 XYZSUPP UPES。

4.4.4 订单条目活动

对于该活动，XML 消息将路由到 MQSeries Integrator 中。该消息流将预备消息，该消息将从申请-回复模式中发送到 CICS 事务处理中。该事务处理是会计软件的条目事务处理。

使用 MQSI.INPUT.OE 作为输入队列创建 XYZORDER UPES。

4.4.5 运送订单活动

该活动将调用运行在 WebSphere 应用服务器中的 Enterprise JavaBean（EJB）。XML 消息将发送到 EJB 客户机程序。

使用 MQSI.INPUT.SH 作为输入队列创建 XYZSHIP UPES。

4.5 综述

此时，我们已经完成了工作流实施。后面我们将增强工作流模型以处理特殊的异常情况从而改善业务流程。您可以在正常运行时环境中导入当前的 FDL。然而，因为一些 UPES 活动是同步的，您所创建的每个实例将一直等待来自 UPES 的回复。

在本书的 179 页的第五章“在 MQSeries Integrator 中执行活动”中，我们将关注作为 MQSeries Integrator 消息流的 UPES 的执行。

5

在MQSeries Integrator中执行活动

在本章中，我们将详细解释用于支持订单处理应用程序的 MQSeries Integrator 中的消息流。消息流由 MQSeries Workflow 服务器请求开始。该消息流是 MQSeries Workflow 中活动的执行。在 MQSeries Workflow 术语中，有时称为 *call-out*。

该消息流由一些数据库操作、路由到供应商的消息，以及在历史格式和 XML 消息之间的格式化组成。

5.1 活动执行概述

在第一个方案中，我们使用如下案例。这些案例需要 MQSeries Integrator 与 MQSeries Workflow 共同工作。

- 客户确认

  该消息流在数据库中检查客户并检索一些关于客户的其他信息。

  该活动调用是同步的。

- 库存确认

  该消息流在数据库中检查产品并在订单数量超出库存时检索供应商信息。

  该活动调用是同步的。

- 供应商订单

  该消息流将入站消息转变成 CWF 格式并将订单发送给供应商。流程还存储数据存储数据，该数据是将回复消息发送给工作流流程实例所需的数据。

  另一个消息流处理订单回执，并创建 MQSeries Workflow 回复消息，然后更新产品库存并以订单数量增加库存。

  该活动调用是同步的。

- CICS 事件处理初始化

  该消息流将引入消息转换成 CICS 所使用的历史格式并将其发送给 CICS 程序。

  该活动调用是同步的。

在以下各节中，我们将介绍该消息流。首先，我们将以关于设计考虑、前提、先决条件的预备词汇开始。本章第二部分将给出对消息流元素进行详细描述。

我们可以在 Web 上获得完整的消息流 XML 输出和消息设置输出。请参考本书第 405 页的附录 C “ 其他资料 ” 获得关于如何检索这些输出的信息。本书的 397 页的附录 B “ 实例应用程序安装 ” 解释了 Web 资料包中每个文件的使用方法。

5.2 设计考虑

本节将描述 UPES 调用背景、应用数据库设计步骤，以及关于消息流设计的若干前提。

5.2.1 外部活动调用概述

我们将在实例中使用 MQSeries Workflow 用户定义程序执行服务器（UPES）工具。使用 UPES 意味着 MQSeries Workflow 将以 XML 格式发送调用请求消息给 MQSeries 队列（称为 UPES 输入队列）。MQSeries Integrator 信息流或 MQSeriesAdapter Offering 适配器（或其它自定义应用程序，例如 JMS 侦听器）将侦听适当的队列并开始处理调用请求。

MQSeries Workflow 的职责是把消息写入 UPES 输入队列。在此之后，处理消息和在 MQSeries Workflow 需要时发送回复消息将成为 UPES 的职责。

我们的应用程序将给出异步 UPES 调用模式的实例。所有 UPES 实例都将把带有字段“ ActivityImplInvoke ”的消息发送给 UPES。然后，我们将给出一个如何处理带有字段“ ActivityExpired ”消息的实例。我们不使用“ TerminateProgram ”消息。

在同步调用的案例中，UPES（例如消息流）必须记忆流入消息 ActivityImplInvoke 的 ActImplCorrelID 字段，以便可将其返回给 ActivityImplInvokeResponse 消息中的 MQSeries Workflow。

如果活动调用定义为异步方式，MQSeries Workflow 将不会等待一个回复。本案例中，ActImplCorrelID 字段将被设置为否（No）。

为了向先前的请求发送应答，UPES 必须执行以下步骤：

- 接收 XML 消息 ActivityImplInvoke。
- 如果需要回复，就需要保存活动调用上下文，即 ActImplCorrelID 和过程用户标识符，该过程代表已经启动的活动实例。
- 如有需要，在完成之后将以 ActivityImplInvokeResponse 消息格式发回带有适当的 ActImplCorrelID 和 UserIdentifier 的应答。

带有类型 ActivityExpired 的消息是异步的，MQSeries Workflow 将不会等待回复。UPES 唯一的职责就是在其环境中处理消息和执行必要的清除操作。

本书第 169 页中的实例 4-1 显示了 UPES 调用消息的实例。实例 5-1 显示了可能的响应消息。

实例 5-1 实例 MQSeries Workflow XML 响应消息

```
<WfMessage>
<WfMessageHeader>
<ResponseRequired>No</ResponseRequired>
</WfMessageHeader>
<ActivityImplInvokeResponse>
<ActImplCorrelID>RUEAAAACAAaABAAAAAAAAAAAAAAAACQAAAAEAGMAAAAAAAAAAAAAAAAAJQQAAAAIABoAFAAAAAAAAAA
BF</ActImplCorrelID>
<ProgramRC>0</ProgramRC>
<ProgramOutputData>
<CustomerValid>
<CustName>Tester</CustName>
<CustOK>Y</CustOK>
<CustAddress>100 Testing Lane</CustAddress>
</CustomerValid>
</ProgramOutputData>
</ActivityImplInvokeResponse>
</WfMessage>
```

关于 MQ 工作流 XML 消息格式或如何调用外部活动执行的更多信息，请参考《MQSeries Workflow3.3 编程指南》编号：SH12-6291-06 中的第五部分“ XML 消息=接口 ”。

入站和出站数据结构相当于在 MQSeries Workflow 中定义的输入和输出容器。两者都合乎 MQSeries Workflow XML 格式。

关于 MQSeries Workflow 输入和输出数据容器的定义，请参见本书的 113 页的 4.1.2 节“ 确定数据结构 ”。

5.2.2 消息流设计考虑

在消息流设计期间，我们决定保持流程尽可能简单并且只关注其基本功能。这意味着在许多情形下消息流功能可能不全。

另一个消息流设计考虑是保持透明的。为了达到该要求，在许多情形下我们将使用更多的节点。如果有最优化需求，那么下一步将是合并计算节点、数据库节点和过滤节点 ESQL，从而使节点数量减少。

我们设计消息流以使不同的消息流使用不同的输入队列。另一种方法是使用同一输入队列，过滤流入消息，然后将其传送给适当的消息流。

图 5-1 显示了全部的代理分配。

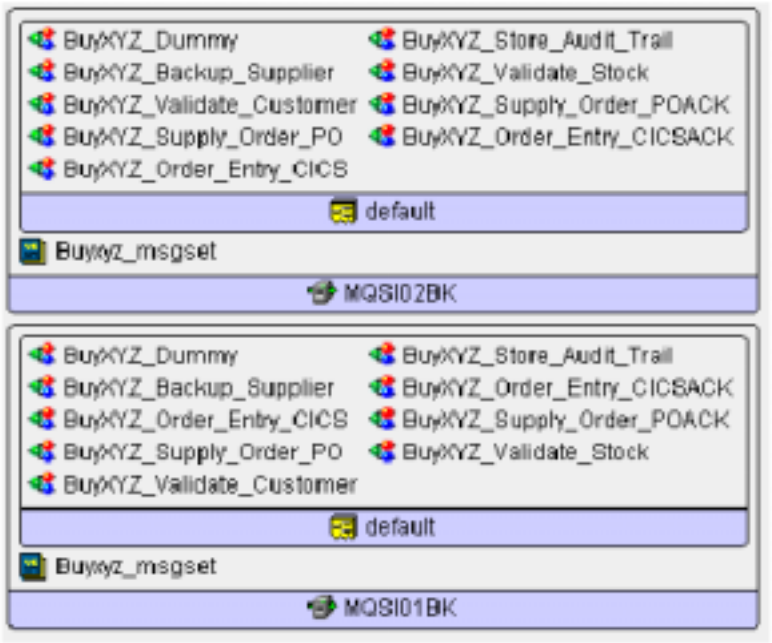

图 5-1 消息流代理分配

5.2.3 应用数据库设计

本节将描述应用数据库 CUSTOMER。该应用数据库是基于 WebCredit 实例的。在本书的 85 页的 3.5 节“确认工作流环境”中，该实例已经在 IVP 期间创建了 CUSTOMER 数据库。在我们的案例中，该数据库在 WebSphere 服务器存在的设备 M23BZZYP 上。

该应用程序使用表 CUSTOMER_DATA 和 CUSTOMER_ORDER 来存储关于客户及其订单的信息。另一些表将存储产品和供应商的详细资料。表 MESSAGELOG 将只保存临时用以创建 MQSeries Workflow 的回复信息。

在应用程序设计期间，我们将使用简化方案，其中只含有属于一个订单的一个产品项目。下面的数据库逻辑和输入/输出数据结构反映了此概念。

图 5-2 显示了 CUSTOMER 数据库的逻辑数据库设计。粗体（bold）显示的是主键，*斜体*显示的是外键。

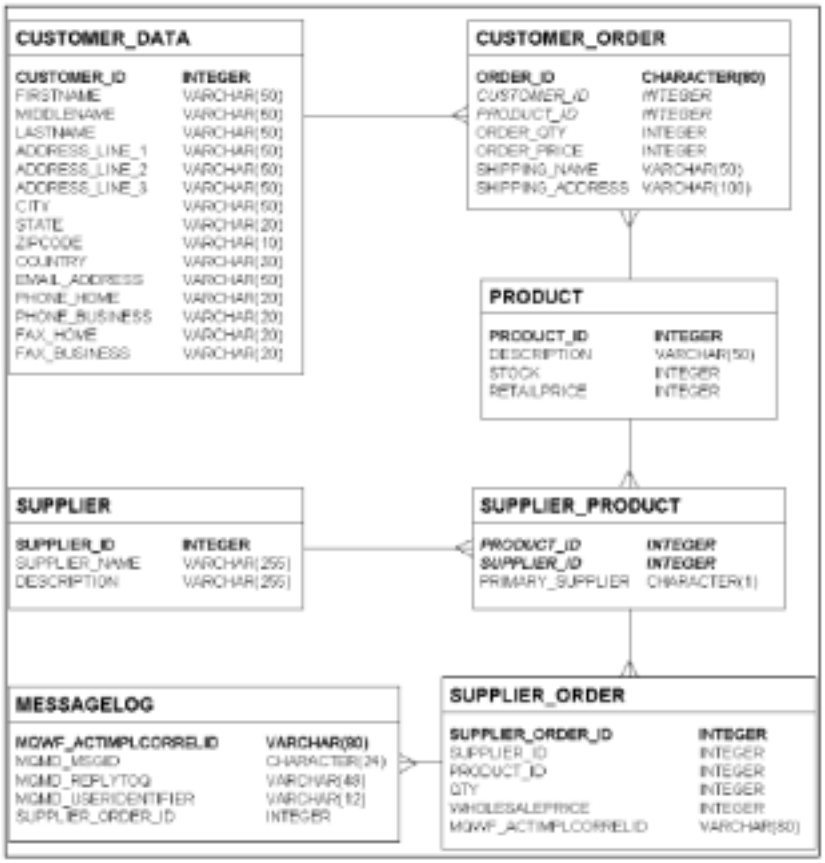

图5-2 逻辑数据库设计

在物理映射时，我们不使用任何外键约束或其它限制。其中一些表将使用 DB2 版本 7.1 的 IDENTITY 数据类型。

关于物理数据库和如何创建数据库的更多细节，请参考本书第 247 页的 5.9.2 节“ 创建应用数据库 ”。

5.3 创建BuyXYZ_Validate_Customer消息流

本节将描述消息流 BuyXYZ_Validate_Customer。第一部分将概述该消息流的功能。本节的剩余部分将用来描述每个节点的细节。这将包括入站和出站数据结构。

图 5-3 显示了完全的消息流。

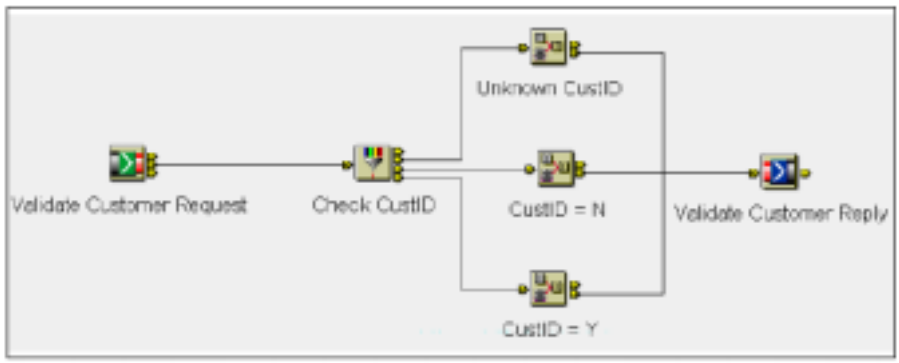

图5-3 确认客户信息流

消息流步骤如下：

- 确认客户。
- 如客户存在，从数据库中检索附带的数据。
- 生成回复消息。

该消息流以 XML 格式接收来自 MQSeries Workflow 服务器的客户确认请求。如果数据库中存在该客户，消息流将检索其他的客户数据，例如客户名和地址，然后创建带有肯定响应的回复消息。否则，消息流将创建否定响应并将客户确定（CustomerOK）字段设置成否（N）。原因码（ProgramRC）设置为 O，它意味着处理过程中没有出现错误。如果数据检索失败，该原因码将被设置为“ -1 ”。

5.3.1 输入和输出数据结构

本节将描述入站和出站消息结构。

我们已在工作流设计期间定义了该数据结构。关于如何创建使用 MQSeries Workflow 的数据结构的更多信息，请参考本书第 113 页的 4.1.2 节“ 确定数据结构 ”。

输入数据结构客户输入（CustomerInput）

输入数据由 MQSeries Workflow 生成。该输入消息的 ProgramInputData XML 字段只包含必须确认的客户标识符。

实例 XML 输入消息类似于实例 5-2。

实例 5-2 客户输入（CustomerInput）数据结构

```
<ProgramInputData>
<_ACTIVITY>ValidateCustomer</_ACTIVITY>
<_PROCESS>Order Process</_PROCESS>
<_PROCESS_MODEL>Order Process</_PROCESS_MODEL>
<CustomerInput>
<CustID>1</CustID>
</CustomerInput>
</ProgramInputData>
```

输出数据结构 CustomerValid

该回复消息由消息流生成，该消息流与 MQSeries Workflow 中所定义的结构相同。

实例 5-3 CustomerValid 数据结构

```
<ProgramOutputData>
<CustomerValid>
<CustName>Tester</CustName>
<CustOK>Y</CustOK>
<CustAddress>100 Testing Lane</CustAddress>
</CustomerValid>
</ProgramOutputData>
```

提示：该实例只显示了消息程序输入数据（ProgramInputData）和程序输出数据（ProgramOutputData）的文件夹。关于完全的 XML 消息的实例，请参考本书第 397 页的附录 B“ 实例程序安装 ”。

5.3.2 消息流细节

现在，我们将看一看消息流 BuyXYZ_Validate_Customer 的每个节点。

MQInput 节点有“效客户需求”（Validate Customer Request）

图 5-4 和图 5-5 显示了 MQInput 节点的配置窗口。

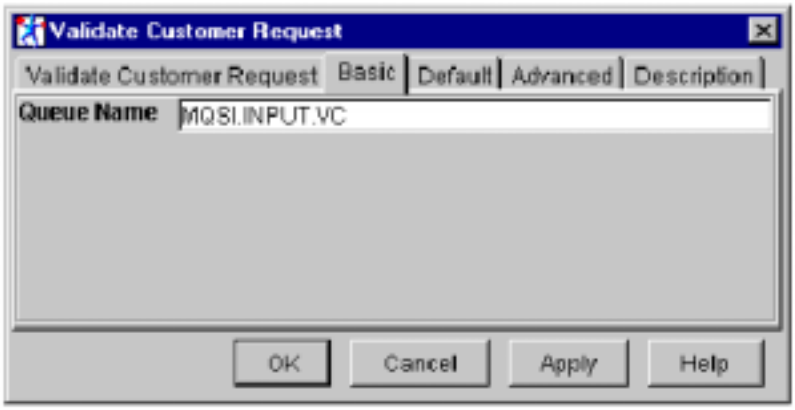

图 5-4 MQInput 节点细节

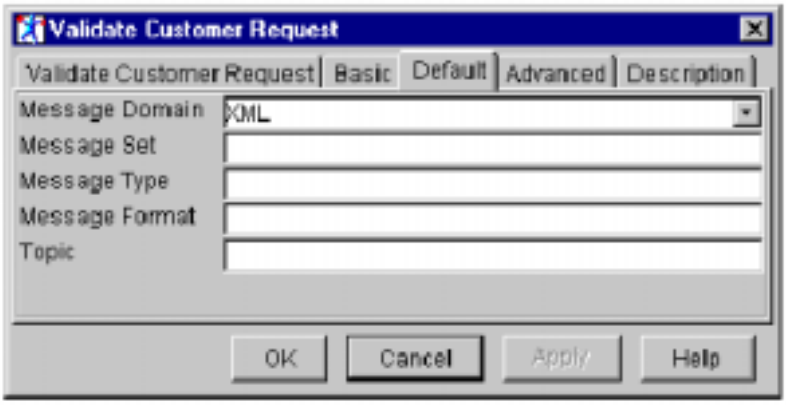

图 5-5 MQInput 节点细节

MQInput 节点只使用以下参数：

- 队列名设置为 MQSI.INPUT.VC。
- 默认消息域设置为 XML。

这意味着 MQSeries Workflow XML 消息将被解析成通用 XML。通用指的是 XML 消息没有在 MQSeries Integrator 的消息仓库管理器(MRM)中定义该事实。利用 MQSeries Workflow XML 工具包，您可生成基于 FDL 文件的 DTD。该工具包可从以下地址下载：

http://www-4.ibm.com/software/ts/mqseries/txppacs/wa05.html

然后，生成的 DTD 可使用 DTD 导入器导入 MRM，该导入器可从以下地址下载：

http://www-4.ibm.com/software/ts/mqseries/txppacs/id04.html

由于该 XML 消息相当简单，我们更喜欢使用 MQSeries Integrator 和 MQSeries Workflow 中的标准工具。

过滤节点“查看 CustID”

过滤节点的属性如图 5-6 所示。

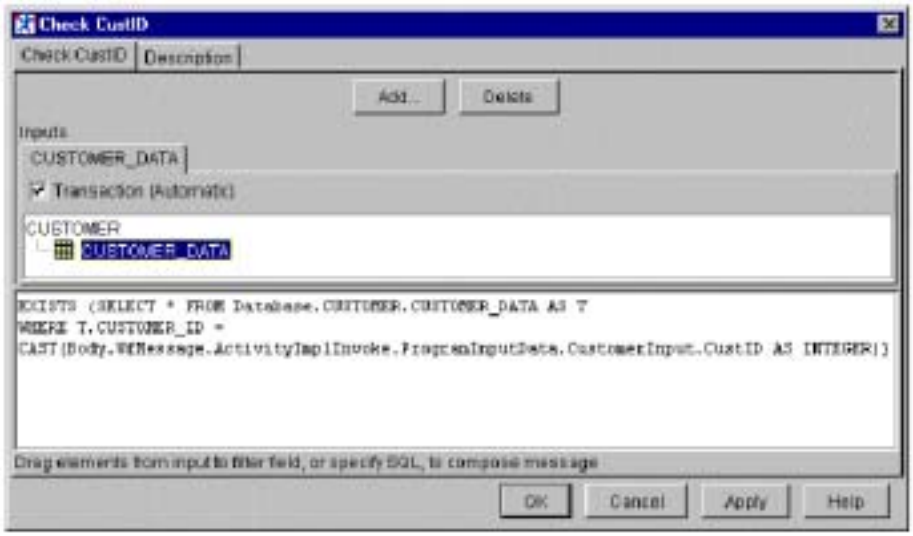

图 5-6 过滤节点细节

该过滤节点将查看数据库中是否存在客户。若存在，发送消息给真实终端。否则，发送消息给假终端。

计算节点“CustID = Y”

该计算节点从客户数据库中检索某些其他数据并在客户确认成功后创建输出消息。图 5-7 显示了该计算节点的属性。

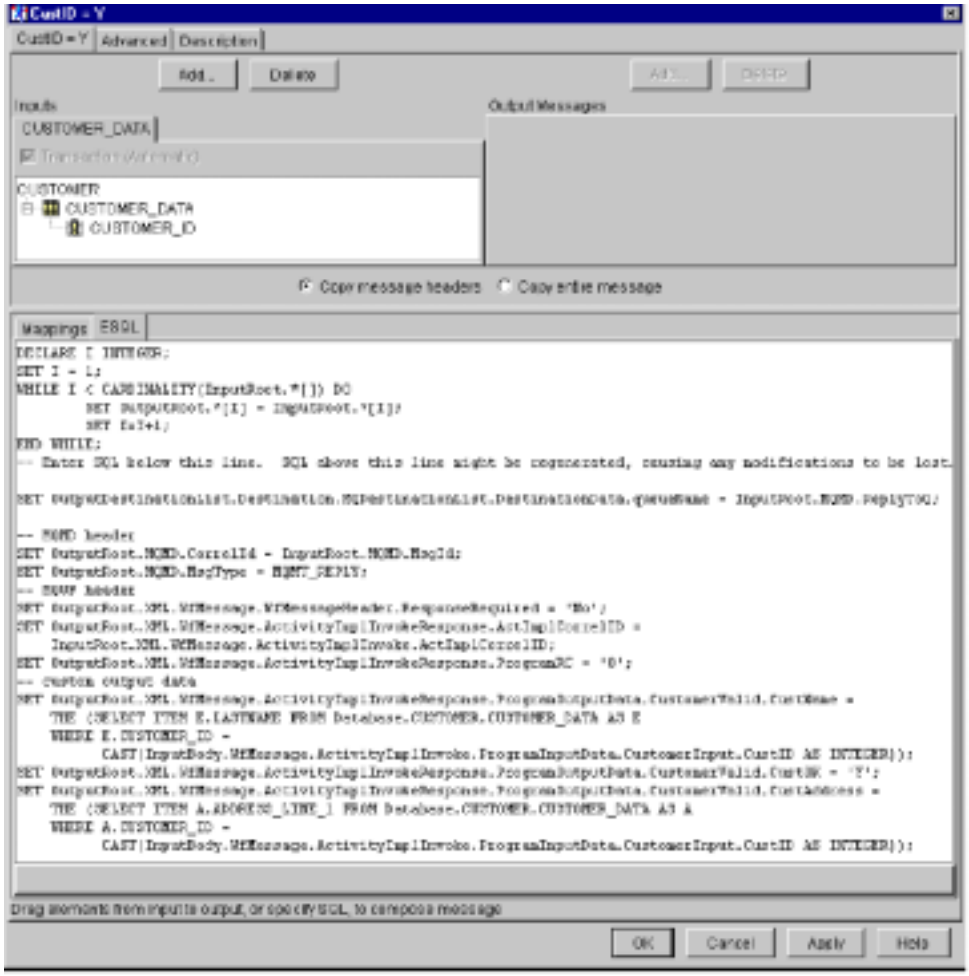
CustID = Y
CustID = Y
Advanced
Description
Add...
Delete
Inputs
Output Messages
CUSTOMER_DATA
CUSTOMER
CUSTOMER_DATA
CUSTOMER_ID
Copy message headers
Copy entire message
Mappings
ESQL
DECLARE I INTEGER;
SET I = 1;
WHILE I < CARDINALITY(InputRoot.*[]) DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I=I+1;
END WHILE;
-- Enter SQL below this line. SQL above this line might be regenerated, causing any modifications to be lost.
SET OutputDestinationList.Destination.MQPutDestinationList.DestinationData.queueName = InputRoot.MQMD.ReplyToQ;
-- MQMD header
SET OutputRoot.MQMD.CorrelId = InputRoot.MQMD.MsgId;
SET OutputRoot.MQMD.MsgType = MQMT_REPLY;
-- SOAP header
SET OutputRoot.XML.WSMessage.WSMessageHeader.ResponseRequired = 'No';
SET OutputRoot.XML.WSMessage.ActivityImplInvokeResponse.ActImplCorrelID =
InputRoot.XML.WSMessage.ActivityImplInvoke.ActImplCorrelID;
SET OutputRoot.XML.WSMessage.ActivityImplInvokeResponse.ProgramRC = '0';
-- Custom output data
SET OutputRoot.XML.WSMessage.ActivityImplInvokeResponse.ProgramOutputData.CustomerValid.CustName =
THE (SELECT ITEM E.LASTNAME FROM Database.CUSTOMER.CUSTOMER_DATA AS E
WHERE E.CUSTOMER_ID =
CAST(InputBody.WSMessage.ActivityImplInvoke.ProgramInputData.CustomerInput.CustID AS INTEGER));
SET OutputRoot.XML.WSMessage.ActivityImplInvokeResponse.ProgramOutputData.CustomerValid.CustOK = 'Y';
SET OutputRoot.XML.WSMessage.ActivityImplInvokeResponse.ProgramOutputData.CustomerValid.CustAddress =
THE (SELECT ITEM A.ADDRESS_LINE_1 FROM Database.CUSTOMER.CUSTOMER_DATA AS A
WHERE A.CUSTOMER_ID =
CAST(InputBody.WSMessage.ActivityImplInvoke.ProgramInputData.CustomerInput.CustID AS INTEGER));
Drag elements from inputs or output, or specify SQL, to compose message
OK
Cancel
Apply
Help

图5-7 计算节点细节

为了匹配 MQSeries Workflow 中的请求，输出消息中固有的 WFMessage XML 字段将从输入消息中映射。将入站 ActivityImplInvoke XML 文件夹映射到出站消息中的 ActivityImplInvokeResponse 文件夹。为实现这一点，实例 5-4 显示了所需的 ESQL 语句。

实例 5-4 计算节点“CustID=Y”

```
SET OutputRoot.XML.WfMessage.WfMessageHeader.ResponseRequired ='No';
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ActImplCorrelID =
InputRoot.XML.WfMessage.ActivityImplInvoke.ActImplCorrelID;
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramRC ='0';
```

对于回复，我们使用目的地列表。但是我们不指定目标队列管理器名，而让 MQSeries 决定将消息发送的地点。我们只在目标回复队列填写，该目标回复队列在 MQSeries 群集中共享。

图 5-8 显示计算节点高级设置。

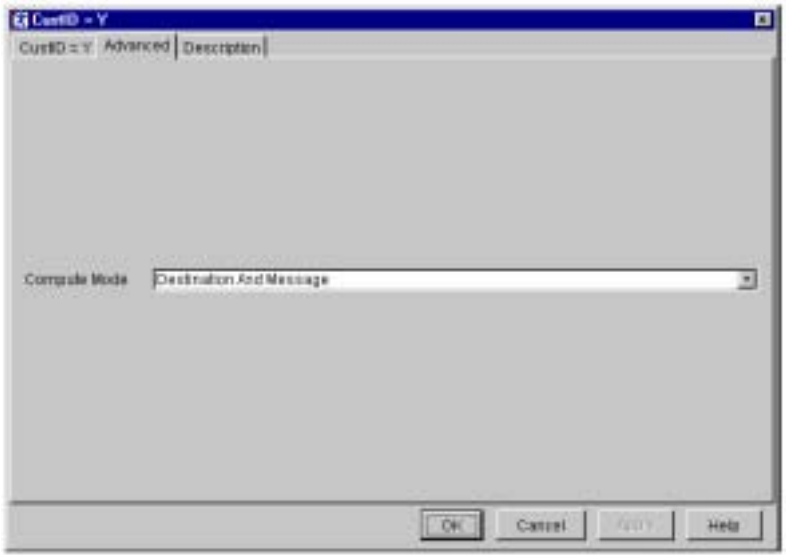

图 5-8 计算节点“CustID=Y”

将计算模式（Compute Mode）设置为目的地和消息（Destination and Message），从而不仅生成消息结构而且生成目的地列表结构。

计算节点“ CustID = N ”

如果客户确认失败，计算节点将以适当的格式创建回复消息。图 5-9 显示了计算节点属性。

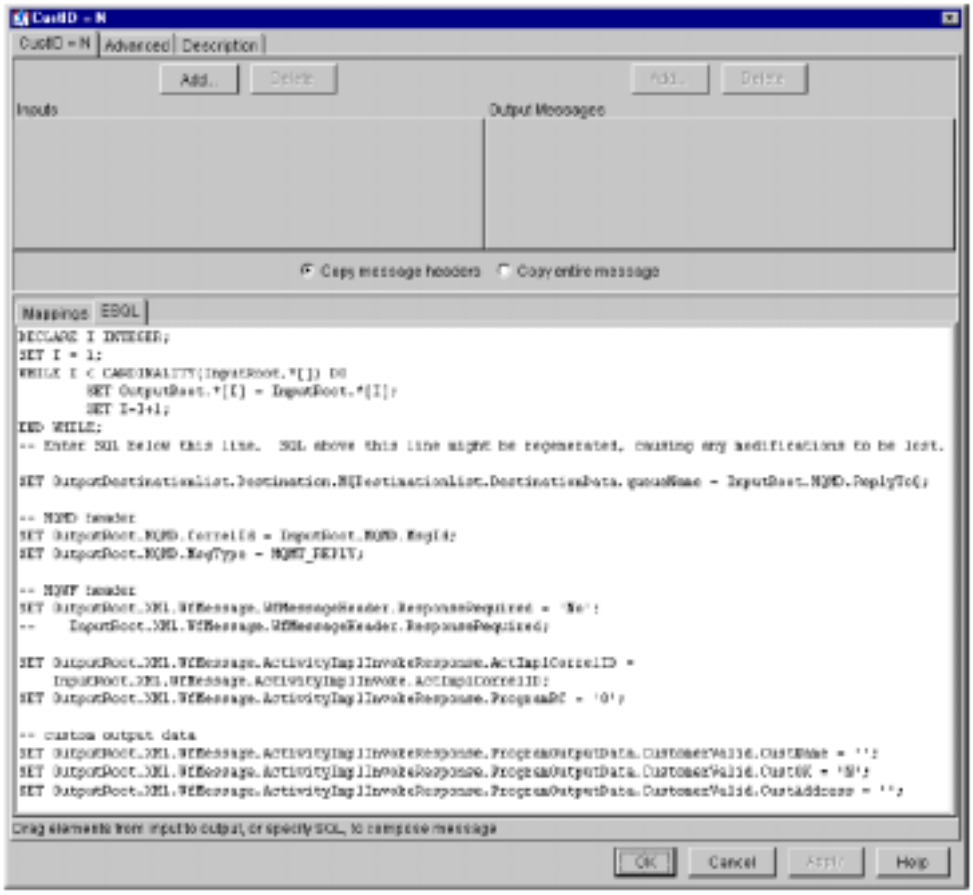

图 5-9 计算节点细节

在本案例中，输出消息的 CustOK 字段被设置为 N，其它回复字段保持空白。

计算节点的高级设置类似于计算节点“ CustID=Y ”的设置。

计算节点“ 未知客户 ID ”（“ Unknown CustID ”）

“ 未知客户 ID ”计算节点将被连接到“ 查看客户 ID ”过滤节点的未知终端，以便发生故障时发送空回复给正在等待的 MQSeries Workflow 流程。图 5-10 显示了该计算节点的属性。

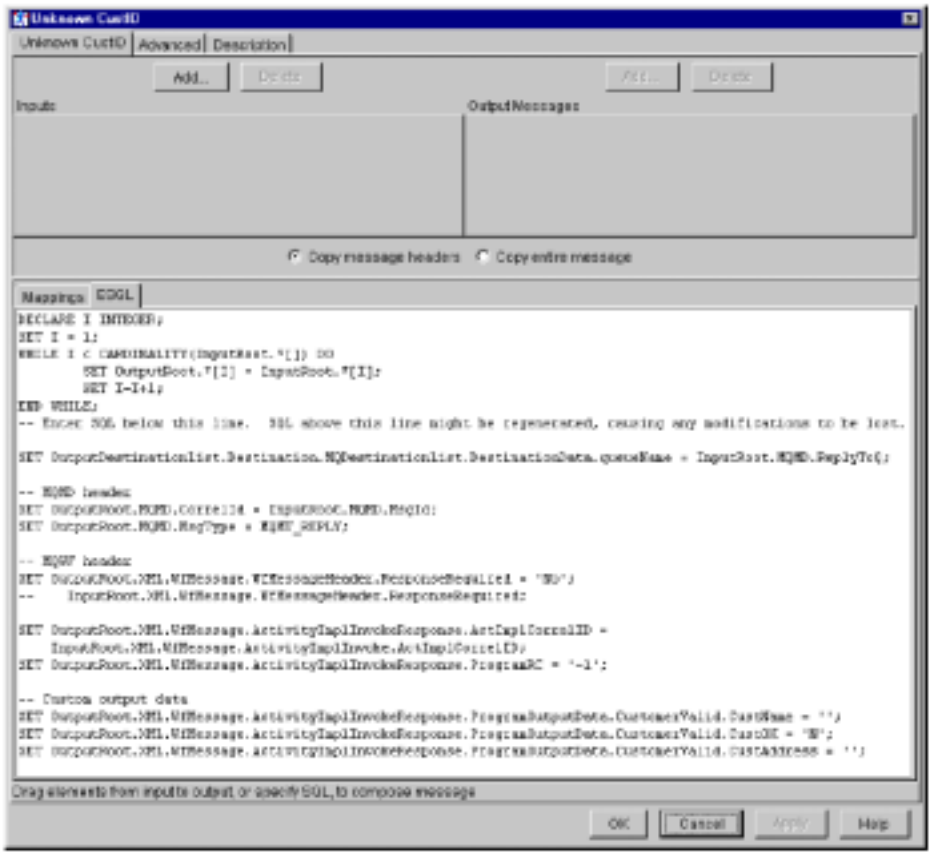

图5-10 计算节点细节

设置原因码（ProgramRC）为“ -1 ”，这意味着 UPES 执行失败。 设置 CustomerValid 输出容器数据为默认值。

提示：该解决方案（计算节点连接到未知过滤节点终端）仅在表示过滤节点表达式为未知时有效（例如，消息中的参考字段不存在）。在更复杂的方案中，建议使用其它的消息流设计方法。例如，使用异常列表与/或 TryCatch 节点。

MQOutput 节点“ 有效客户回复 ”（“ Validate Customer Reply ”）
图 5-11 和图 5-12 显示了 MQOutput 节点的设置。

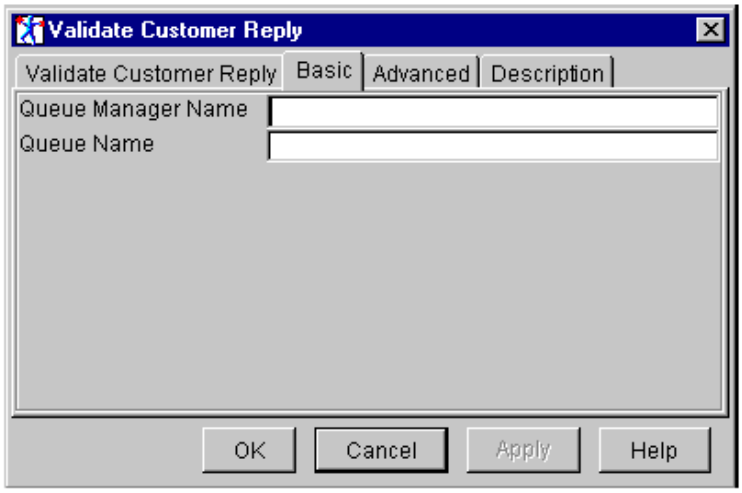

图 5-11 MQOutput 节点细节

不填写队列管理器名和队列名参数。在计算节点中把目的地模式（“ Destination Mode ”）参数设置为“ 目的地列表 ”。因此，该消息将被发送到在目的地列表中命名的队列。目的地列表不包含任何关于队列管理器的信息。选择适当的队列管理器是 MQSeries 的任务。

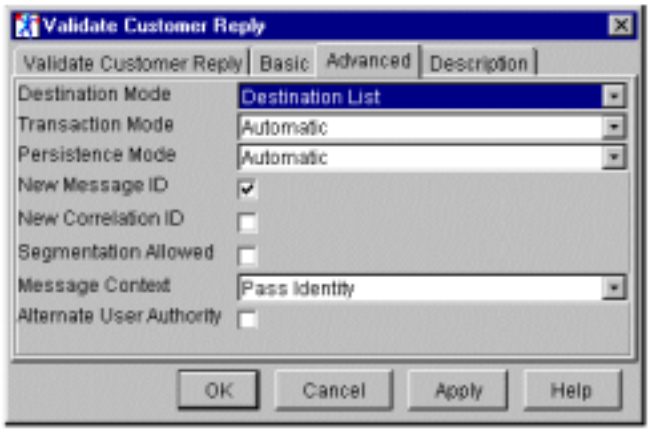

图 5-12　MQOutput 节点细节

将消息上下文（Message Context）参数设置为 Pass Identity，这是创建 MQSeries Workflow 确认消息的关键。除了该选项之外，回复消息将把 MQMD 字段 UserIdentifier 设置为代理用户 ID。该用户 ID 一般不是已知的或已授权的 MQSeries Workflow 用户 ID。

当您开发自己的 MQSeries Integrator 消息流与 MQSeries Workflow 交互时，您的 MQSeries Workflow XML 消息可能被执行服务器拒绝。为了获得更多细节或理解为什么消息被执行服务器拒绝，您可以使用 MQSeries Workflow 管理员工具。单击 Start -> Programs -> IBM MQSeries Workflow （IBM MQSeries Workflow）->MQSeries Workflow 管理实用程序（MQSeries Workflow Administration Utility） - XYZ 并以 ADMIN 用户 ID 登录。选择 l 作为系统日志然后再次单击 l 列出的条目。

实例 5-5 显示了管理实用程序的运行，其中有两个 XML 消息被拒绝。您也可参考错误日志（主目录中的 e 选项）在执行服务器中找出您 XML 消息的问题。

实例 5-5 管理实用程序的输出

```
- FMC16006I Administration Utility started. (启动 FMC16006I 管理实用程序)
System group name (系统组名称)  :[BUYXYZ ] BUYXYZ
System name (系统名称)  :[BUYXYZ02 ]BUYXYZ02
Userid (用户 ID)  :[ADMIN ] ADMIN
Password (密码)  :[] *********
- FMC16301I UserID 'ADMIN'connected to system 'BUYXYZ02'. (将 FMC16301I 用户 ID'ADMIN'连接到系统'BUYXYZ02')
FMC15010I Main Menu (FMC15010I 主菜单) :
s ...System Commands Menu (系统命令菜单)
m ...Select Server Menu (选择服务器菜单)
e ...Errorlog Commands Menu (错误日志命令菜单)
l ...Systemlog Commands Menu (系统日志命令菜单)
u ...User Commands Menu (用户命令菜单)
```

```
x ...Exit Main Menu（退出主菜单）
=FMC16110I Receive thread for userID 'ADMIN'at system 'BUYXYZ02'started. (在系统'BUYXYZ02'
启动时，FMC16110I 为用户 ID'ADMIN'接收线程)
l
FMC15070I Systemlog Commands Menu（FMC15070I 系统日志命令菜单）：
i ...Info
l ...List（列表）
p ...Purge（清除）
x ...Exit Systemlog Commands Menu（退出系统日志命令菜单）
l
-4/30/01 6:36:17 PM BUYXYZ02:FMC01100E Incorrect XML document(FMC01100E 错误 XML 文档).The
message that is returned(返回的消息)
by the XML parser is:Fatal Error at (line 1,char 327):
-4/30/01 6:42:49 PM BUYXYZ02:FMC10120I Administration server stopping.( FMC10120I 管理服务器
停止)
-4/30/01 6:42:49 PM BUYXYZ02:FMC10030W System is being shutdown.( FMC10030W 系统关闭)
-4/30/01 6:42:49 PM BUYXYZ02:FMC10510I Execution server instance stopped.( FMC10510I 执行服
务器实例停止)
-4/30/01 6:42:49 PM BUYXYZ02:FMC10210I Execution server for system BUYXYZ02 stopped.(系统
BUYXYZ02 的 FMC10210I 执行服务器停止)
-4/30/01 6:42:54 PM BUYXYZ02:FMC10020I System BUYXYZ02 in system group BUYXYZ stopped.(系统
组 BUYXYZ 中的 FMC10020I 系统 BUYXYZ02 停止)
-4/30/01 6:42:54 PM BUYXYZ02:FMC10130I Administration server for system BUYXY
Z02 stopped.( 系统 BUYXYZ02 的 FMC10130I 管理服务器停止)
-5/1/01 9:00:55 AM BUYXYZ02:FMC10100I Administration server starting.(启动 FMC10100I 管理服
务器)
-5/1/01 9:00:55 AM BUYXYZ02:FMC10110I Administration server for system BUYXYZ
02 started.( 启动系统 BUYXYZ02 的 FMC10110I 管理服务器)
-5/1/01 9:01:00 AM BUYXYZ02:FMC10200I Execution server for system BUYXYZ02 st
arted.(启动系统 BUYXYZ02 的 FMC10200I 执行服务器)
-5/1/01 9:01:01 AM BUYXYZ02:FMC10500I Execution server instance started. (启动 FMC10500I 执
行服务器实例)
-5/1/01 9:01:02 AM BUYXYZ02:FMC10000I System startup complete(完全启动 FMC10000I 系统).System
BUYXYZ0
2 in system group BUYXYZ is now running.(正在运行系统组 BUYXYZ 中的 2)
-5/1/01 3:22:21 PM BUYXYZ02:FMC10100I Administration server starting.(启动 FMC10100I 管理服
务器)
-5/1/01 3:22:26 PM BUYXYZ02:FMC10200I Execution server for system BUYXYZ02 started.(启动系
统 BUYXYZ02 的 FMC10200I 执行服务器)
-5/1/01 3:22:26 PM BUYXYZ02:FMC10500I Execution server instance started.(启动 FMC10500I 执行
服务器实例)
-5/1/01 3:22:27 PM BUYXYZ02:FMC10500I Execution server instance started. (启动 FMC10500I 执
行服务器实例)
-5/1/01 3:22:27 PM BUYXYZ02:FMC10000I System startup complete(完全启动 FMC10000I 程序).System
BUYXYZ0
2 in system group BUYXYZ is now running. (正在运行系统组 BUYXYZ 中的 2)
-5/1/01 3:22:27 PM BUYXYZ02:FMC10500I Execution server instance started.(启动 FMC10500I 执行
服务器实例)
-5/2/01 10:14:53 AM BUYXYZ02:FMC12200I The completion of the program for activity
'QQAAAAEATUA+AAAAAAAAAAAAAAALAAAAAQBQAAAAAAAAAAAAAAAAAAAtB'was ignored.
```

5.4 创建BuyXYZ_Validate_Stock消息流

本节将阐述消息流 BuyXYZ_Validate_Stock。图 5-13 显示了完全的消息流。

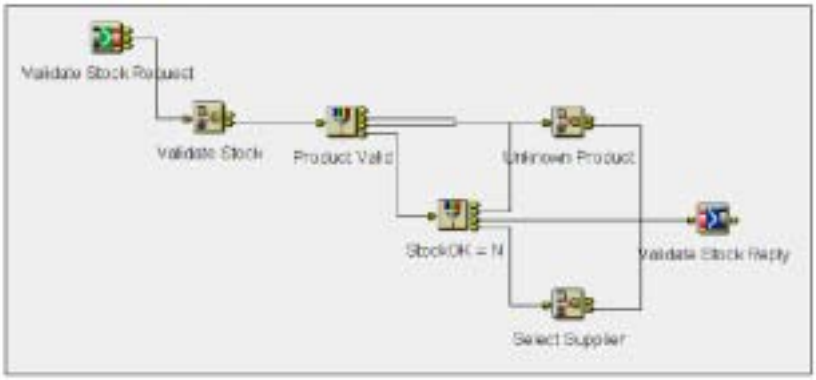

图5-13 确认库存消息流

该消息流的目的是确认产品和检查库存。作为初始步骤，该消息流将确认数据库中的产品。如果产品列表中存在该产品，那么该消息流将检查库存并与订单数量相比较。最后如需要，将从数据库中检索供应商标识符。如果出现错误，回复消息包含相应的 MQSeries Workflow 原因码（ProgramRC）。

5.4.1 输入和输出数据结构

输入和输出数据结构与在 MQSeries Workflow 构建时环境中分配给有效库存活动的数据结构相同。

输入数据结构库存输入

实例库存输入 XML 输入消息的 ProgramInputData 字段类似实例 5-6 所示。

实例5-6 库存输入数据结构

```
<ProgramInputData>
<_ACTIVITY>ValidateStock</_ACTIVITY>
<_PROCESS>Order Process</_PROCESS>
<_PROCESS_MODEL>Order Process</_PROCESS_MODEL>
<StockInput>
<ProdID>1</ProdID>
<ProdQty>1</ProdQty>
</StockInput>
</ProgramInputData>
```

输出数据结构 StockValid

该输出数据结构包含库存确认的结果和其他关于产品和供应商的数据。

实例 5-7StockValid 数据结构

```
<ProgramOutputData>
<StockValid>
<StockOK>Y</StockOK>
<ProdDesc>CD1</ProdDesc>
<SupplierID>0</SupplierID>
</StockValid>
</ProgramOutputData>
```

5.4.2 消息流细节

本节将阐述消息流 the BuyXYZ_Validate_Stock 中每个节点的细节。

MQInput 节点“ 有效库存需求 ”（“ 有效库存 Request ”）

图 5-14 和图 5-15 显示了“ MQInput ”节点的配置。

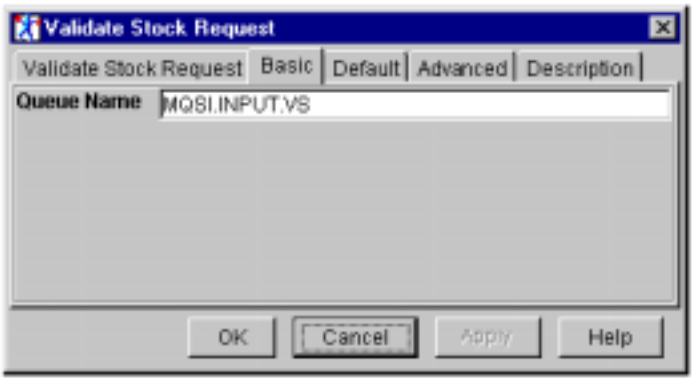

图 5-14 “MQInput ”节点细节

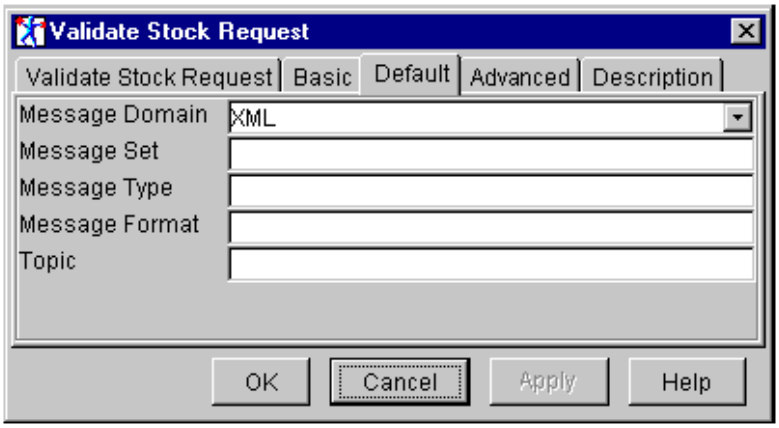

图5-15“MQInput”节点细节

将“消息域”设置为 XML，因为“MQInput”节点从 MQSeries Workflow 接收标准格式的 XML 消息。

计算节点“有效库存”

计算节点将执行该消息流的大部分工作。图 5-16 显示了计算节点的属性。

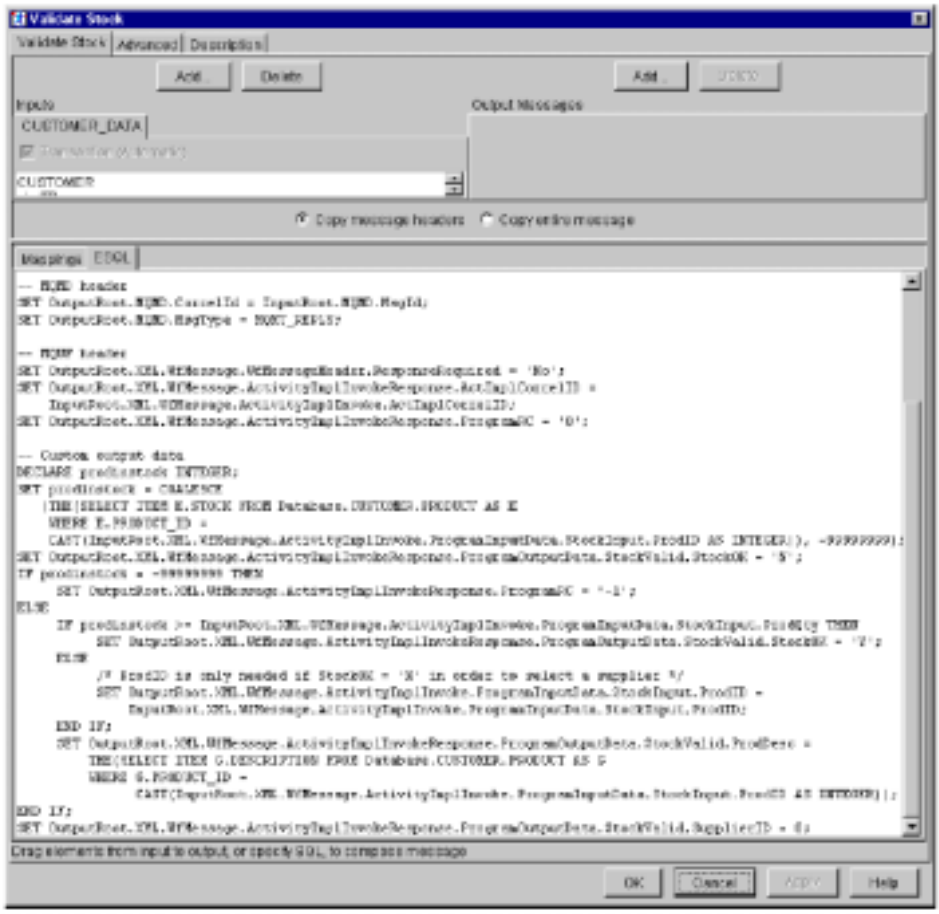

图5-16 计算节点细节

在复制消息标题之后，消息流将检查入站产品标识符以确定产品是否正确、是否有库存。实例 5-8 显示了读数据库 ESQL 语句。

实例 5-8 计算节点“有效库存”

```
SET prodinstock =COALESCE(THE(SELECT ITEM E.STOCK FROM
Database.CUSTOMER.PRODUCT AS E WHERE E.PRODUCT_ID =
CAST(InputRoot.XML.WfMessage.ActivityImplInvoke.ProgramInputData.
StockInput.ProdID AS INTEGER)),-9999999);
```

数据库查询将返回产品库存信息。如果数据库中没有该产品的信息，查询将返回空值，“COALESCE”函数将返回-99999999。因为在我们的实例应用程序中可以有负产品库存，所以我们可以使用该数值。然后，我们将选择一个足够大的数字用作无效号码。

如果数据库中不存在该产品信息，这意味着定单无效，程序将把原因码（ProgramRC）置为‘-1’。参看实例 5-9 中相应的 ESQL 语句。这将触发调用 MQSeries Workflow 活动来取消定单。

实例5-9 计算节点“有效库存”

```
IF prodinstock =-99999999 THEN
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramRC ='-1';
```

最后，该节点将与实际库存比较引入产品数量，如实例 5-10 所示。

实例5-10　计算节点“有效库存”

```
IF prodinstock >=InputRoot.XML.WfMessage.ActivityImplInvoke.ProgramInputData.
StockInput.ProdQty THEN
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.
ProgramOutputData.StockValid.StockOK ='Y';
ELSE
/*ProdID is only needed if StockOK ='N'in order to select a supplier */
SET OutputRoot.XML.WfMessage.ActivityImplInvoke.
ProgramInputData.StockInput.ProdID =InputRoot.XML.WfMessage.
ActivityImplInvoke.ProgramInputData.StockInput.ProdID;
```

如果定单数量没有超过库存，无需生成供应商定单，出站“ProgramOutputDataXML”区域的返回字段“StockOK”将被设置为‘Y’。

假定需要供应商定单（ELSE 语句），另一个节点将选择适当的供应商。为了达到该要求，必须将产品标识符复制到该节点输出中去。

过滤节点“ Product Valid ”

该过滤节点检验产品确认的结果。它将执行前面的计算节点以便使在产品标识符无效时结果码（ProgramRC 字段）包含‘ -1 ’。该过滤节点的属性如图 5-17 所示。

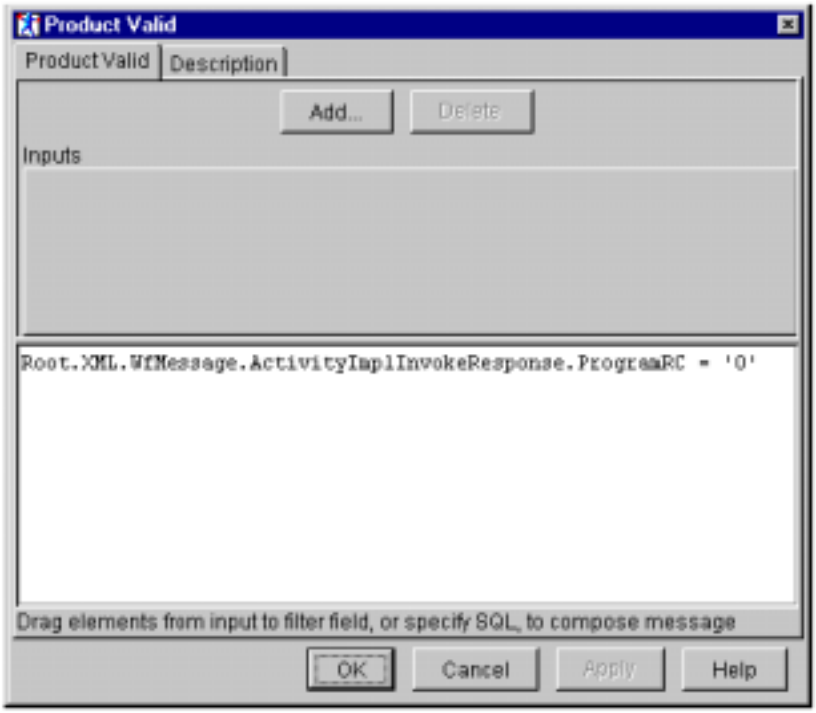

图 5-17　过滤节点细节

过滤节点“ StockOK = N ”

该过滤节点检查库存确认的结果并决定是否选择一个供应商。该过滤节点属性如图 5-18 所示。

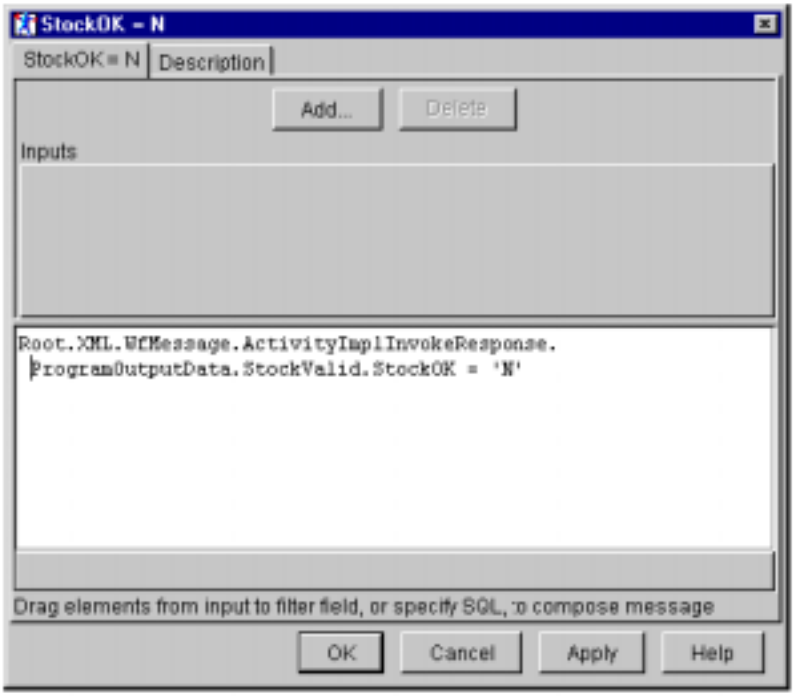

图 5-18　过滤节点细节

计算节点“ 选择供应商 ”

该计算节点连接到过滤节点“ StockOK = N ”的真实终端。如果库存中没有足够产品以履行定单，MQSeries Workflow 将发送一个采购定单给适当的供应商。为此，供应商标识符是必需的。

图 5-19 显示了计算节点“ 选择供应商 ”的属性。

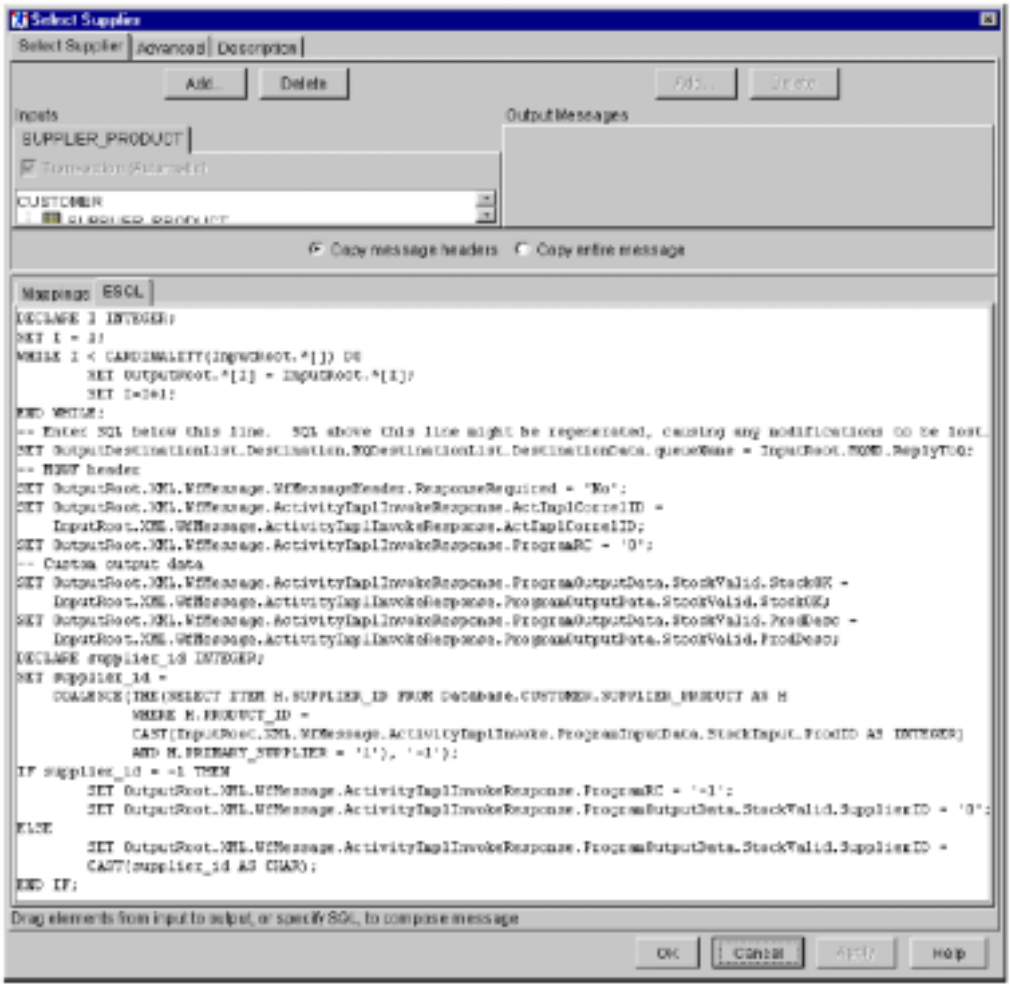
Select Supplier
Select Supplier
Advanced
Description
Add...
Delete
Inputs
SUPPLIER_PRODUCT
CUSTOMER
Output Messages
Copy message headers
Copy entire message
Mappings
ESQL
DECLARE I INTEGER;
SET I = 3;
WHILE I < CARDINALITY(InputRoot.*[]) DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I=I+1;
END WHILE;
-- Enter SQL below this line. SQL above this line might be regenerated, causing any modifications to be lost.
-- MQMT header
SET OutputRoot.MQMD.ReplyToQ = InputRoot.MQMD.ReplyToQ;
SET OutputRoot.XML.WfMessage.WfMessageHeader.ResponseRequired = 'No';
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ActImplCorrelID =
InputRoot.XML.WfMessage.ActivityImplInvokeResponse.ActImplCorrelID;
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramRC = '0';
-- Custom output data
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.StockValid.StockOK =
InputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.StockValid.StockOK;
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.StockValid.ProdDesc =
InputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.StockValid.ProdDesc;
DECLARE supplier_id INTEGER;
SET supplier_id =
COALESCE((SELECT ITEM H.SUPPLIER_ID FROM Database.CUSTOMER.SUPPLIER_PRODUCT AS H
WHERE H.PRODUCT_ID =
CAST(InputRoot.XML.WfMessage.ActivityImplInvoke.ProgramInputData.StockInput.ProdID AS INTEGER)
AND H.PRIMARY_SUPPLIER = '1'), '-1');
IF supplier_id = -1 THEN
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramRC = '-1';
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.StockValid.SupplierID = '0';
ELSE
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.StockValid.SupplierID =
CAST(supplier_id AS CHAR);
END IF;
Drag elements from input to output, or specify SQL, to compose message
OK
Cancel
Apply
Help

图5-19 计算节点细节

该计算节点将从数据库中检索相应的供应商标识符。实例 5-11 显示了确定供应商 ID 的 ESQL 语句。

实例 5-11 计算节点“选择供应商”

```
SET supplier_id =COALESCE(THE(SELECT ITEM H.SUPPLIER_ID FROM
Database.CUSTOMER.SUPPLIER_PRODUCT AS H
WHERE H.PRODUCT_ID =
CAST(InputRoot.XML.WfMessage.ActivityImplInvoke.ProgramInputData.
StockInput.ProdID AS INTEGER)AND H.PRIMARY_SUPPLIER ='1'),'-1');
```

该算法与产品确认相同。数据库查询将返回供应商标识符，或在不能发现有效供应商时返回空值，“ COALESCE ”函数将返回‘ -1 ’。

该节点将检查查询结果并设置相应的输出消息字段。所需的 ESQL 语句如实例 5-12 所示。

实例 5-12　计算节点“选择供应商”

```
IF supplier_id =-1 THEN
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.
ProgramRC ='-1';
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.
ProgramOutputData.StockValid.SupplierID ='0';
ELSE
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.
ProgramOutputData.StockValid.SupplierID =
CAST(supplier_id AS CHAR);
END IF;
```

计算节点“未知产品”

如果由于一个错误的产品标识符而使产品定单无效时，该计算节点将创建 MQSeries Workflow 回复消息。

“ ProgramOutputData ”字段被设置为默认值。原因码（ProgramRC）将被设置为‘ -1 ’以表明处理过程中出现了一个错误。图 5-20 显示了该计算节点的属性。

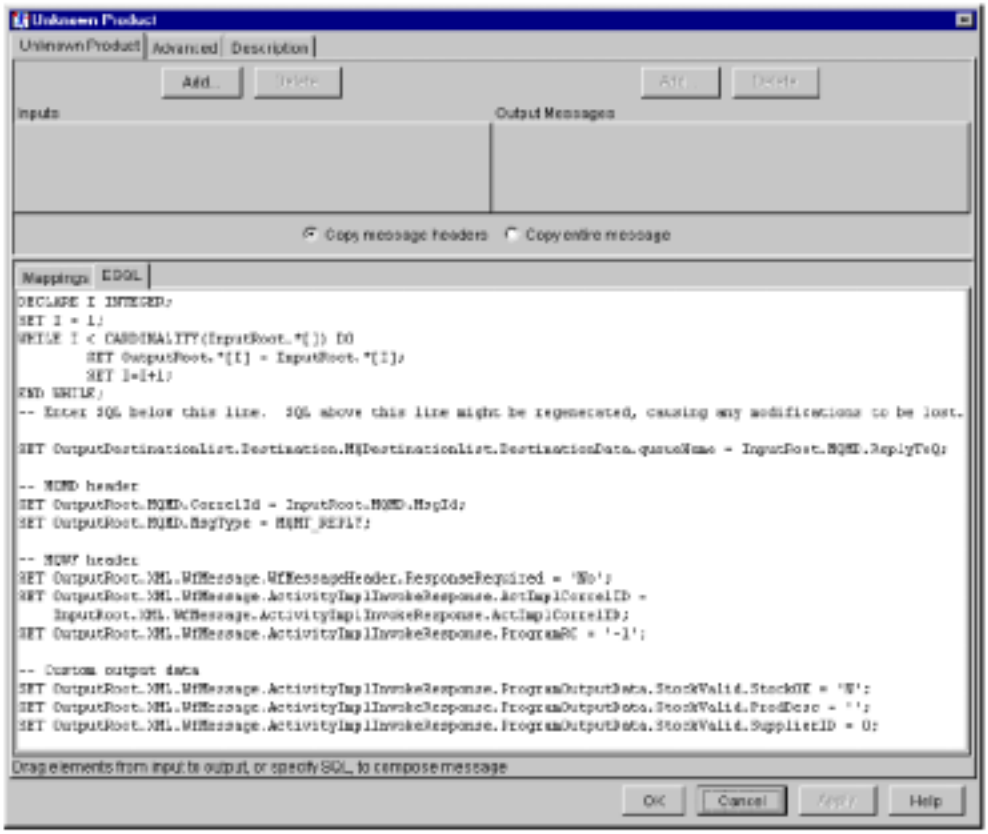

图5-20　计算节点细节

MQOutput 节点“ 有效库存回复 ”（“Validate Stock Reply”）

该“ MQOutput ”节点的设置与第 193 页“ MQOutput 节点“ 有效客户回复 ”小节中的“ MQOutput ”节点 “ 有效客户回复 ”相同。消息将被写入使用目的地列表的队列。队列管理器不在该目的地列表中指定，而只指定队列——MQSeries 群集队列。MQSeries 群集将决定在哪一个队列中写入给定的消息。

5.5 创建BuyXYZ_Supply_Order_PO消息流

本节将描述消息流 BuyXYZ_Supply_Order_PO。

图 5-21 显示了完全的消息流。

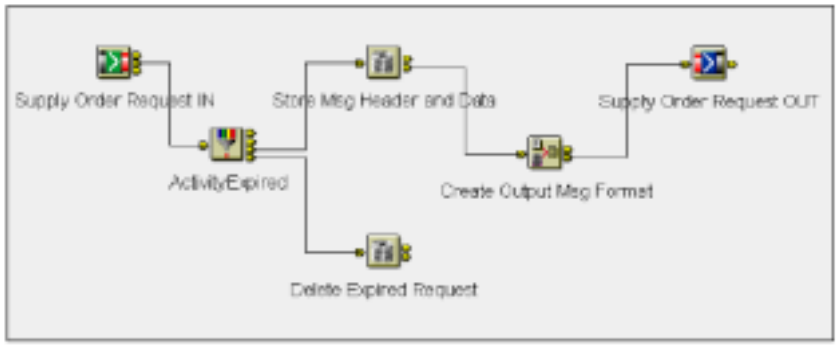

图 5-21 BuyXYZ_Supply_Order_PO 消息流

该消息流接收两种不同的消息：

- 带有“ ActivityImplInvoke ”字段的工作流 XML 消息。在本案例中，该消息流的输出是一条准备发送给供应商的消息。
- 带有“ ActivityExpired ”字段的工作流 XML 消息。

当收到来自 MQSeries Workflow 的“ ActivityImplInvoke ”请求时，该流程将在数据库中保存必要的字段。该步骤是随后创建回复消息给 MQSeries Workflow 的关键。在保存标题信息之后，该流程将以 CWF（历史电报格式）创建输出消息。

在本案例中，当收到来自 MQSeries Workflow 的“ ActivityExpired ”消息（该消息包含来自先前已过期活动的 ActImplCorrelID ）时，该流程将从数据库中删除“ ActImplCorrelID ”，因为它与活动已不再相关。该活动终止意味着我们已等待比留给供应商回复消息更久的时间。最后，当供应商发送回复消息时，在 5.6 节“ 创建 BuyXYZ_Supply_Order_POACK 消息流 ”（215 页）中所讨论的消息流将处理此异常情况。

5.5.1 输入和输出数据结构

该 XML 消息中的输入和输出数据结构又来源于构建时环境中所做的定义。

输入数据结构“ SupplyOrder ”

该输入数据结构与 MQSeries Workflow 中的“ SupplyInput ”数据容器相同。实例“ StockInput ” XML 输入消息的“ ProgramInputData ”字段如实例 5-13 所示。

实例 5-13“ SupplyOrder ”数据结构

```
<ProgramInputData>
<_ACTIVITY>SupplyOrder</_ACTIVITY>
<_PROCESS>Order Process</_PROCESS>
<_PROCESS_MODEL>Order Process</_PROCESS_MODEL>
<SupplyInput>
<SupplierID>1</SupplierID>
<ProdID>1</ProdID>
<ProdQty>10</ProdQty>
</SupplyInput>
</ProgramInputData>
```

输入数据结构“ ActivityExpired ”

实例 5-14 显示了带有“ ActivityExpired ”字段的消息内容。

实例 5-14“ ActivityExpired ”数据结构

```
<WfMessage>
  <WfMessageHeader>
    <ResponseRequired>No</ResponseRequired>
  </WfMessageHeader>
  <ActivityExpired>
    <ActImplCorrelID>RUEAAAABAE1AWwAAAAAAAAAAAAAAAwAAAAEAUIAAAAAAAAAAAAAAAAA
    DQQAAAAEATYBmAAAAAAAAAABF</ActImplCorrelID>
  </ActivityExpired>
</WfMessage>
```

输出数据结构“ Supply_Order_PO ”

图 5-22 显示了与 MRM 数据仓库的相关部分。该消息通过导入 C 语言结构生成。

图5-22 消息类型定义

关于完全的消息设置，请参考附录 B“ 实例程序安装 ”（397 页）。

5.5.2 消息流细节

本节将描述消息流 BuyXYZ_Supply_Order_PO 中每个节点的细节。

MQInput 节点“ 供应定单需求 IN ”

该“ MQInput ”节点将接收 XML 消息并从给定的输入队列中读取信息。该设置与另一个消息流的设置相同，该消息流处理从 MQSeries Workflow 以 XML 格式发来的入站消息。图 5-23 和图 5-24 显示了该“ MQInput ”节点的属性。

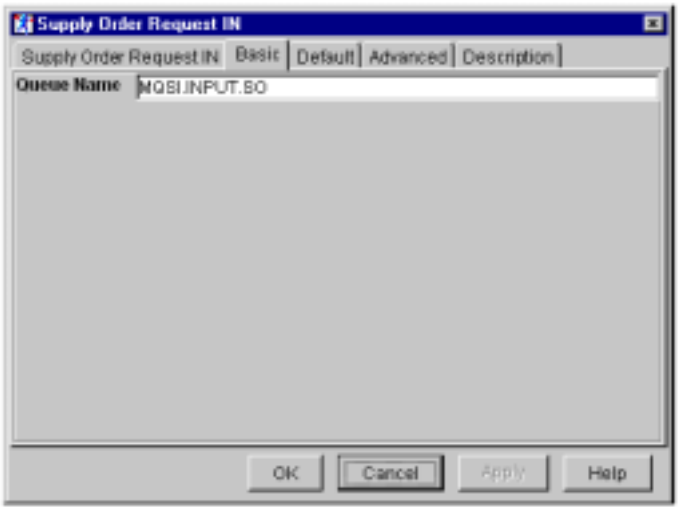

图5-23“MQInput”节点细节

在“默认”标签中（如图 5-24 所示），“消息域”被设置为 XML。

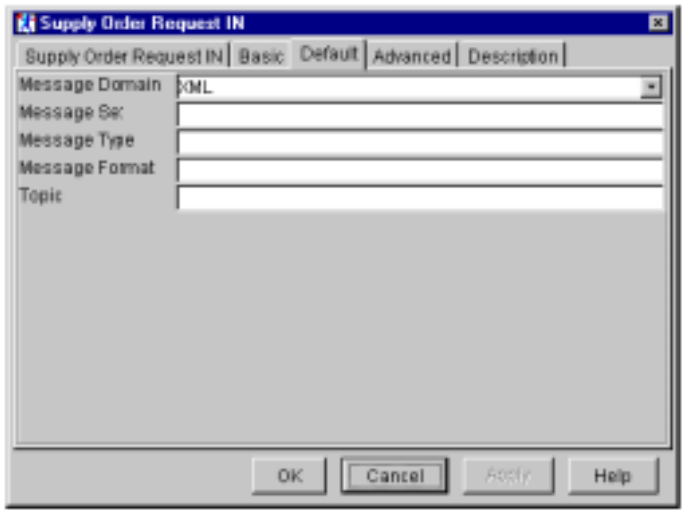

图5-24“MQInput”节点细节

过滤节点“ActivityExpired”

该过滤节点用于区分我们所期望的两个不同的消息。图 5-25 显示了该过滤节点的属性。

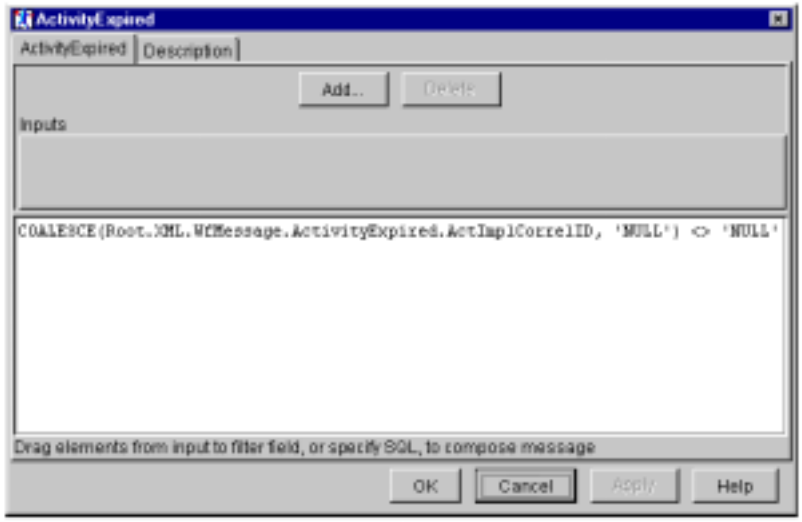

图5-25 过滤节点细节

该过滤节点将检查入站消息的类型并将其路由给流程的适当部分。如果该消息有“ ActivityExpired ”字段，它将被传送给真实输出终端。

数据库节点“ 存储 Msg 标题和数据 ”

该数据库节点将两条不同的信息分别保存在不同的表中：

- 供应商定单细节保存在表 CUSTOMER.SUPPLIER_ORDER 中。
- 消息标题信息保存在表 CUSTOMER.MESSAGELOG 中。

图 5-26 显示了该数据库节点的属性。

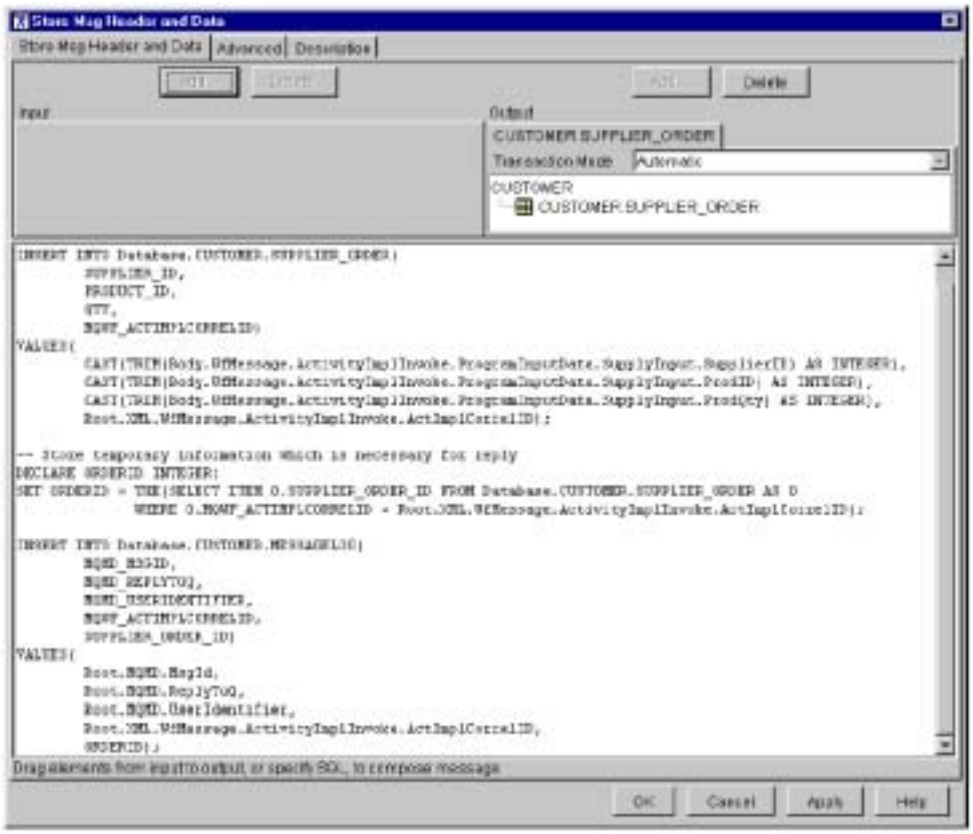

图 5-26 数据库节点细节

对于第二个“ 插入 ”语句，我们需要由第一个“ 插入 ”语句创建的“ ORDER_ID ”。为此，我们将使用“ ActImplCorrelID ”（如实例 5-15 所示）。

*实例 5-15 数据*库节点“ 存储 Msg 标题和数据 ”

```
SET ORDERID =THE(SELECT ITEM O.SUPPLIER_ORDER_ID FROM Database.
CUSTOMER.SUPPLIER_ORDER AS O WHERE O.MQWF_ACTIMPLCORRELID =
Root.XML.WfMessage.ActivityImplInvoke.ActImplCorrelID);
```

另一个选项应使用“ ActImplCorrelID ”作为表“ SUPPLIER_ORDER ”的一个主键。

计算节点“ Create Output Msg Format ”

该计算节点的属性如图 5-27 所示。

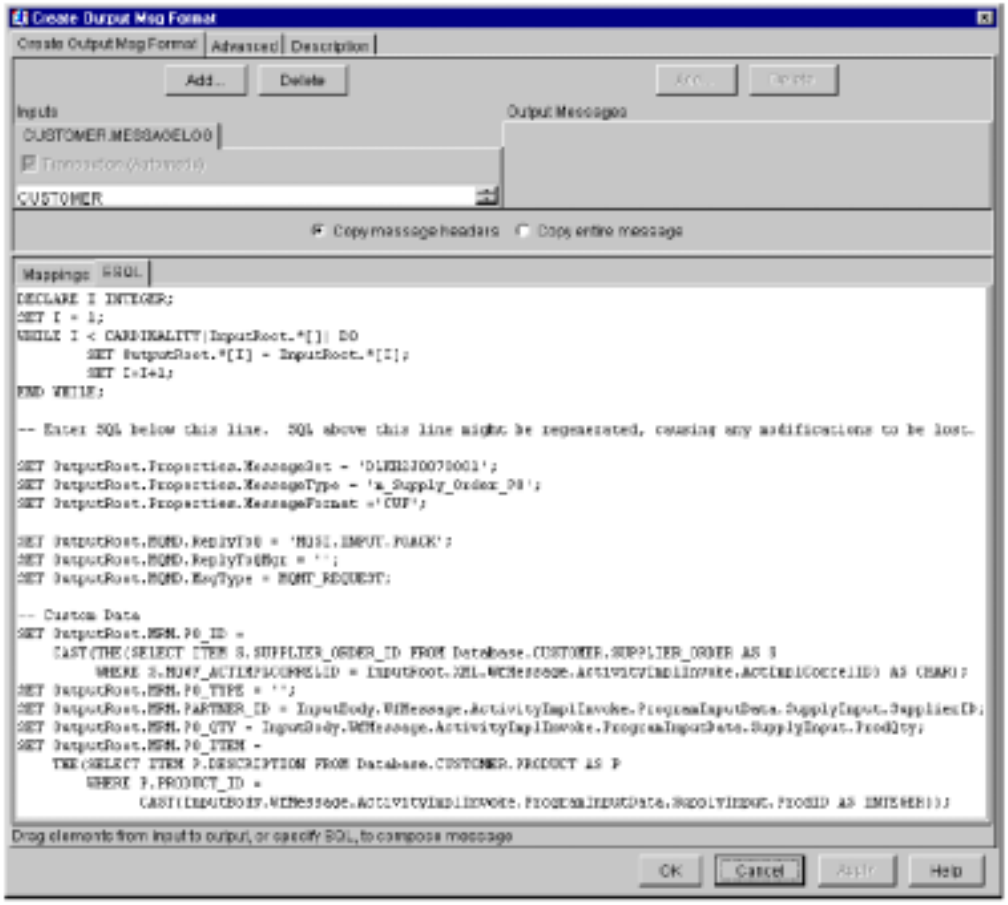

图5-27 计算节点细节

该计算节点将入站 XML 消息转变为 CWF 格式。为此，我们要先设置“ 属性 ”字段以指定 MRM 消息。(见实例 5-16)。

实例5-16 计算节点“ 创建 Msg 格式”

```
SET OutputRoot.Properties.MessageSet ='DLKH2JO07O001';
SET OutputRoot.Properties.MessageType ='m_Supply_Order_PO';
```

```
SET OutputRoot.Properties.MessageFormat ='CWF';
```

该节点将为适当的 MQMD 字段赋值。队列 MQSI.INPUT.POACK 是由在 5.6 节“ 创建 BuyXYZ_Supply_Order_POACK 流程 ”（第 215 页）中讨论的消息流使用的队列名。

实例 5-17 计算节点“ 创建输出 Msg 格式 ”（Create Output Msg Format）

```
SET OutputRoot.MQMD.ReplyToQ ='MQSI.INPUT.POACK';
SET OutputRoot.MQMD.ReplyToQMgr ='';
SET OutputRoot.MQMD.MsgType =MQMT_REQUEST;
```

该计算节点将使用如实例 5-18 所示的 ESQL 语句填写 MRM 文件夹。

实例 5-18 计算节点“ 创建输出 Msg 格式 ”（Create Output Msg Format）

```
SET OutputRoot.MRM.PO_ID =
  CAST(THE(SELECT ITEM S.SUPPLIER_ORDER_ID FROM Database.CUSTOMER.
  SUPPLIER_ORDER AS SWHERE S.MQWF_ACTIMPLCORRELID =
  InputRoot.XML.WfMessage.ActivityImplInvoke.ActImplCorrelID)AS CHAR);
SET OutputRoot.MRM.PO_TYPE ='';
SET OutputRoot.MRM.PARTNER_ID =
  InputBody.WfMessage.ActivityImplInvoke.ProgramInputData.
  SupplyInput.SupplierID;
SET OutputRoot.MRM.PO_QTY =
  InputBody.WfMessage.ActivityImplInvoke.ProgramInputData.
  SupplyInput.ProdQty;
SET OutputRoot.MRM.PO_ITEM =
  THE(SELECT ITEM P.DESCRIPTION FROM Database.CUSTOMER.PRODUCT AS P
  WHERE P.PRODUCT_ID =CAST(InputBody.WfMessage.ActivityImplInvoke.
  ProgramInputData.SupplyInput.ProdID AS INTEGER));
```

MQOutput 节点“ 供应订单请求 OUT ”（Supply Order Request OUT）

图 5-28 和图 5-29 显示了该 MQOutput 节点的基本属性和高级属性。

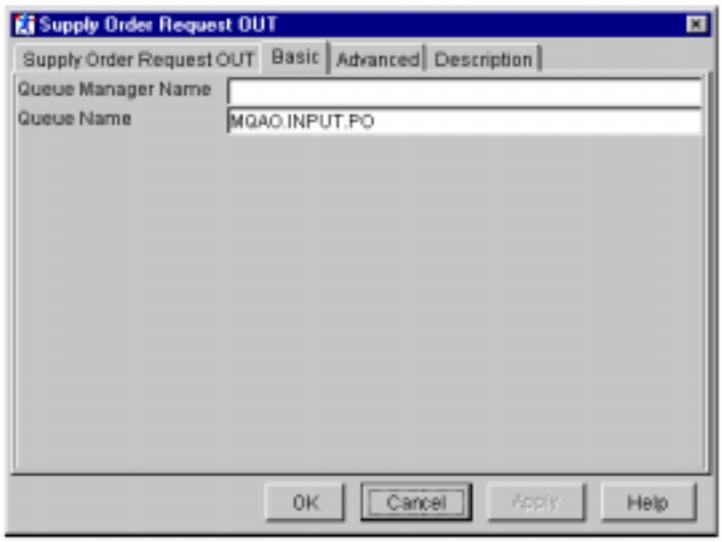

图 5-28MQOutput 节点细节

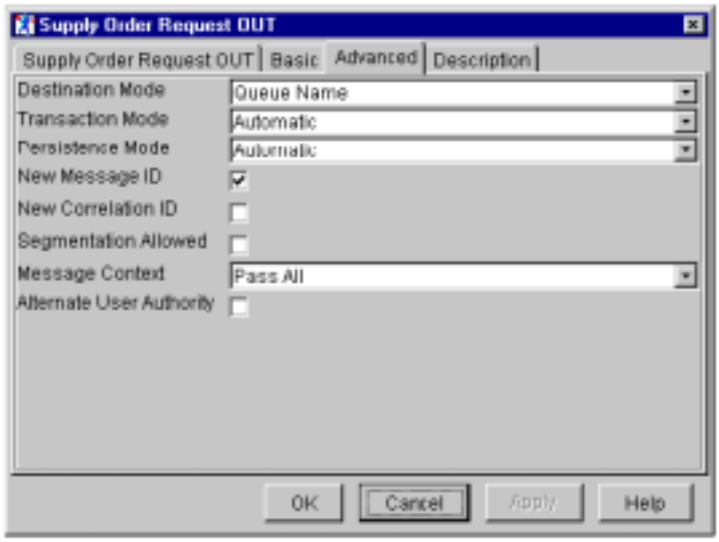

图 5-29MQOutput 节点细节

该节点将发送消息到群集队列。因此，我们只需指定队列名即可。

数据库节点“ 删除到期请求 ”（Delete Expired Request）

图 5-30 显示了该数据库节点的属性。

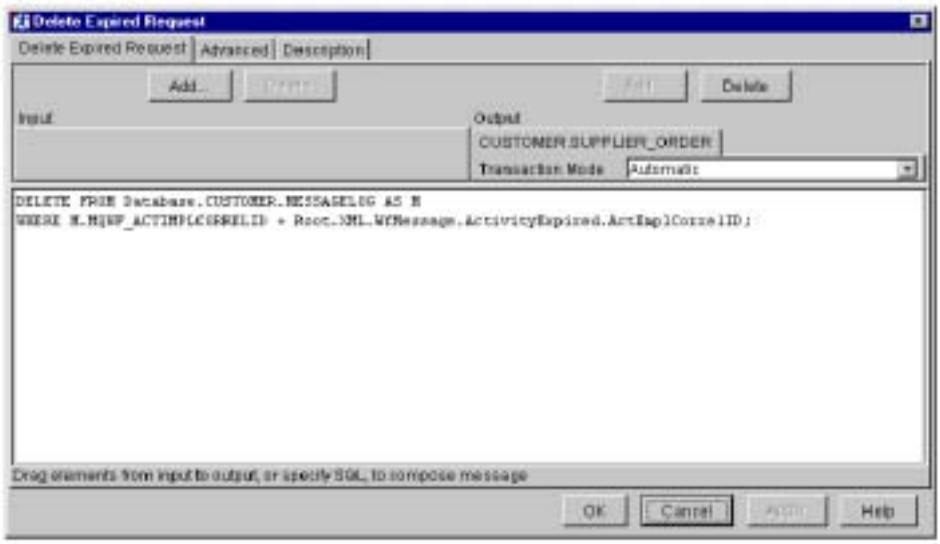

图5-30 计算节点细节

该数据库节点将从表 CUSTOMER.MESSAGELOG 中删除到期记录。该节点将使用 ActImplCorrelID 作为主键以在数据库中找到该记录。

5.6 创建BuyXYZ_Supply_Order_POACK消息流

本节将描述消息流 BuyXYZ_Supply_Order_POACK。它将处理由供应商发送的返回消息。该消息流将创建 MQSeries Workflow 回复消息以表明活动 SupplyOrder 已经完成。

图 5-31 显示了完全的消息流。

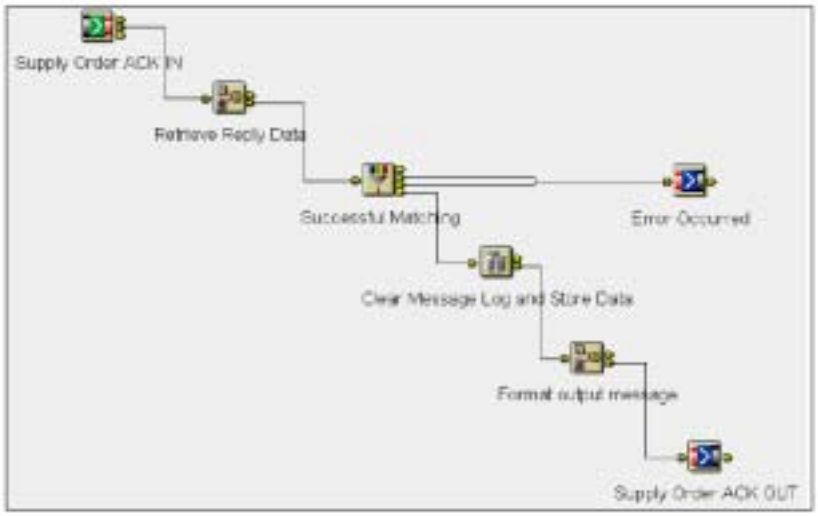

图5-31 消息流概览

该消息流将接收来自历史消息格式中供应商的回复消息。消息流步骤如下：

- 寻找适当的请求消息。
- 检索必要的数据字段。
- 将引入 MRM 字段映射成适当的 XML 元素。
- 在成功映射后，将包含回复信息的临时数据从数据库中删除，以保持表尽可能小。
- 保存入站价格，并在供应订单成功执行后增加库存。

5.6.1 输入和输出数据结构

在本节中，我们将描述引入和流出消息格式。

输入数据结构“ Supply_Order_POACK ”

入站消息是历史格式。我们将定义它使用 MRM 格式。您也可以使用已导出的 XML 文件来创建消息设置。

图 5-32 显示了该消息的消息区域和消息长度。

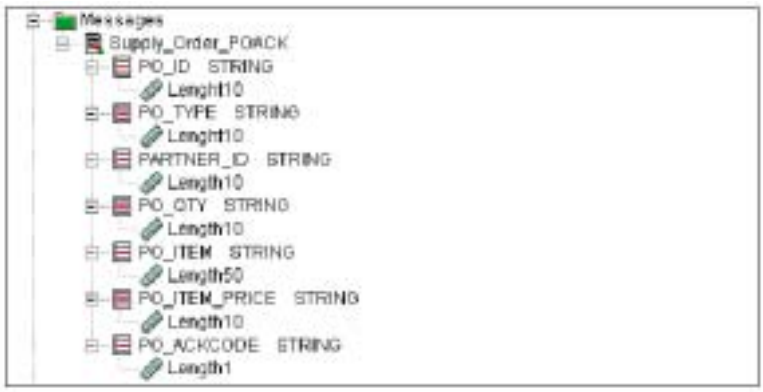

图 5-32 消息类型定义

输出数据结构 SupplyValid

输出数据结构包含供应订单的结果和实际供应价格（参见图 5-19）。

实例 5-19StockValid 数据结构

```
<ProgramOutputData>
  <SupplyValid>
    <SupplyOK>Y</SupplyOK>
    <SupplierPrice>100</SupplierPrice>
  </SupplyValid>
</ProgramOutputData>
```

5.6.2 消息流细节

本节将描述消息流 BuyXYZ_Supply_Order_POACK 中每个节点的细节。

MQInput 节点“ 供应订单 ACK IN ”（Supply Order ACK IN）

图 5-33 和图 5-34 显示了 MQInput 节点的配置窗口。

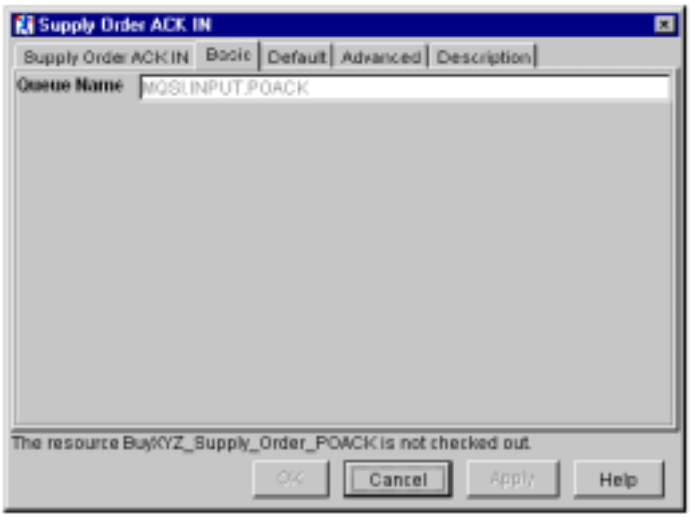

图 5-33MQInput 节点细节

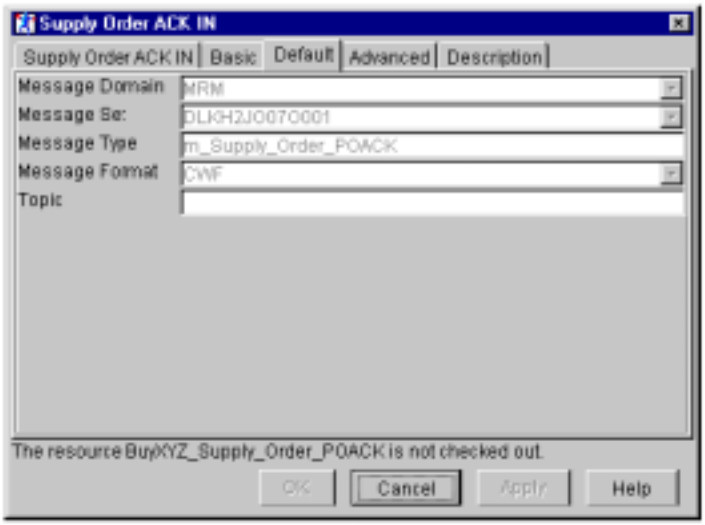

图 5-34MQInput 节点细节

引入消息是 CWF 格式。在节点参数中设置所需默认消息参数。没有这些设置，该节点将不能够解析引入消息，因为该消息没有 MQRFH2 部分。

计算节点“ 检索回复数据 ”（Retrieve Reply Data）

图 5-35 显示了该计算节点的属性。

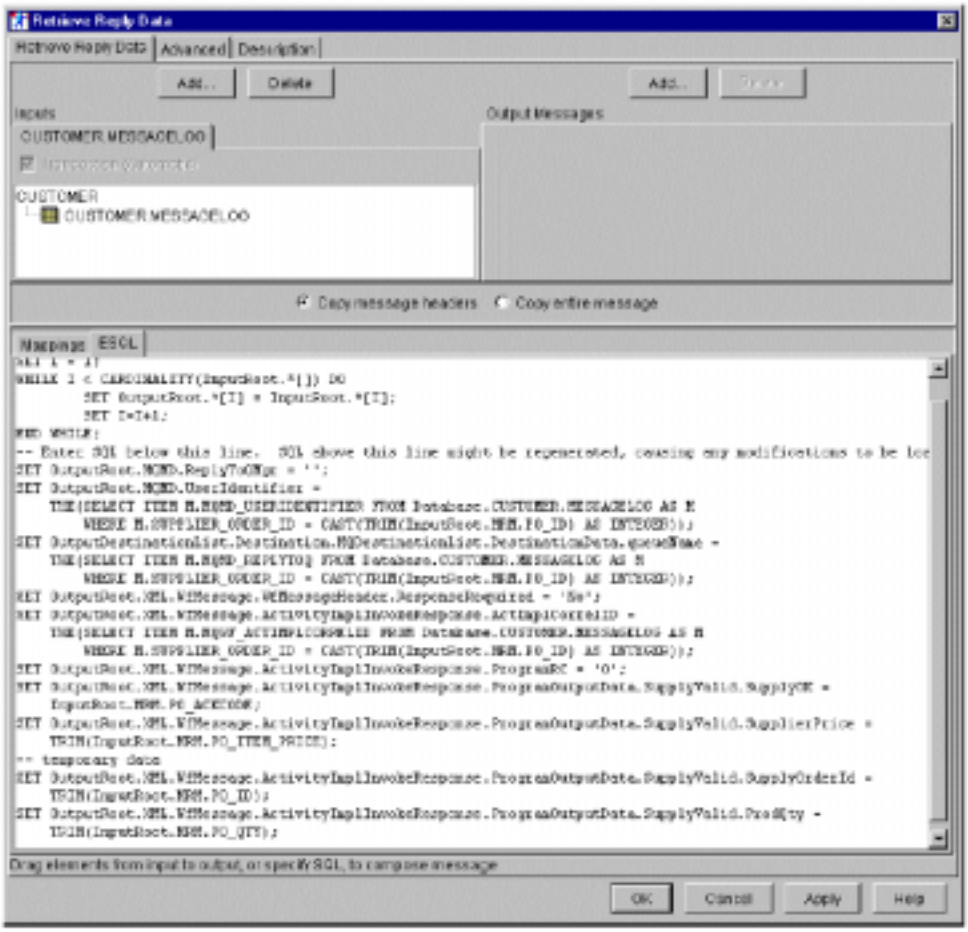

图5-35 计算节点细节

该计算节点将检索回复消息的必需字段并将其复制到输出终端。

如实例 5-20 所示，我们将使用不同查询检索数据并为每个相应的输出字段指定 SELECT 语句。

另一个选项将使用计算节点中的临时 XML 文件夹保存查询结果（作为一列）。我们将在 5.8 节“ 创建 BuyXYZ_Order_Entry_CICSACK 消息流 ”（第 233 页）中给出如何使用该方法的实例。

实例 5-20 计算节点“ 检索回复数据 ”（Retrieve Reply Data）

```
SET OutputRoot.MQMD.UserIdentifier =
  THE(SELECT ITEM M.MQMD_USERIDENTIFIER FROM
  Database.CUSTOMER.MESSAGELOG AS M WHERE M.SUPPLIER_ORDER_ID =
  CAST(TRIM(InputRoot.MRM.PO_ID)AS INTEGER));
SET OutputDestinationList.Destination.MQDestinationList.DestinationData.
  queueName =
  THE(SELECT ITEM M.MQMD_REPLYTOQ FROM Database.CUSTOMER.MESSAGELOG AS M
  WHERE M.SUPPLIER_ORDER_ID =CAST(TRIM(InputRoot.MRM.PO_ID)AS INTEGER));
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ActImplCorrelID =
  THE(SELECT ITEM M.MQWF_ACTIMPLCORRELID FROM
  Database.CUSTOMER.MESSAGELOG AS M WHERE M.SUPPLIER_ORDER_ID =
  CAST(TRIM(InputRoot.MRM.PO_ID)AS INTEGER));
```

最后，该节点将把引入数据字段映射到流出消息中。

有些字段是更新数据库所必须的，但却不是输出消息的一部分。我们也将映射这些字段，并在数据库操作使用另一个计算节点之后将其删除。

将计算节点 Advanced 参数设置为目的地和消息。

过滤节点“ 产品有效 ”（Product Valid）

该过滤节点的属性如图 5-36 所示。

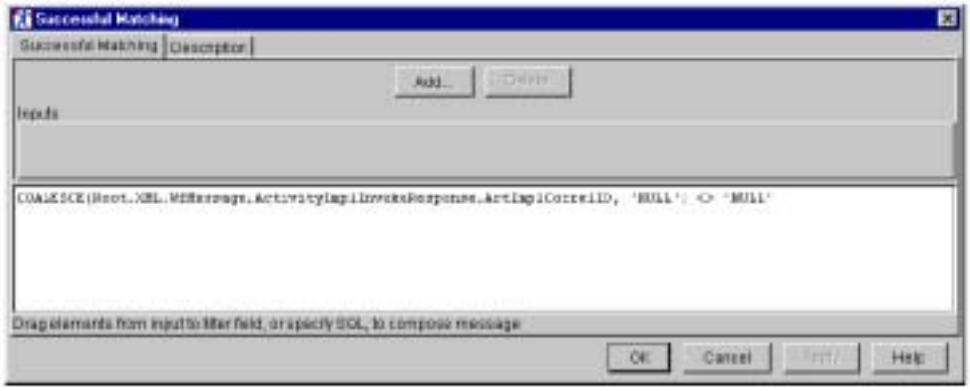

图 5-36 过滤节点细节

该过滤节点将检查计算节点“ 检索回复数据 ”（Retrieve Reply Data）中数据检索的结果。如果查询失败，SELECT 的结果将为空，这意味着该作业将删除 ActImplCorrelID 字段。我们可以使用 COALESCE 函数过滤这种情况。

过滤节点“ 清除消息日志和储存数据 ”（Clear Message Log and Store Data）

图 5-37 显示了该数据库节点属性。

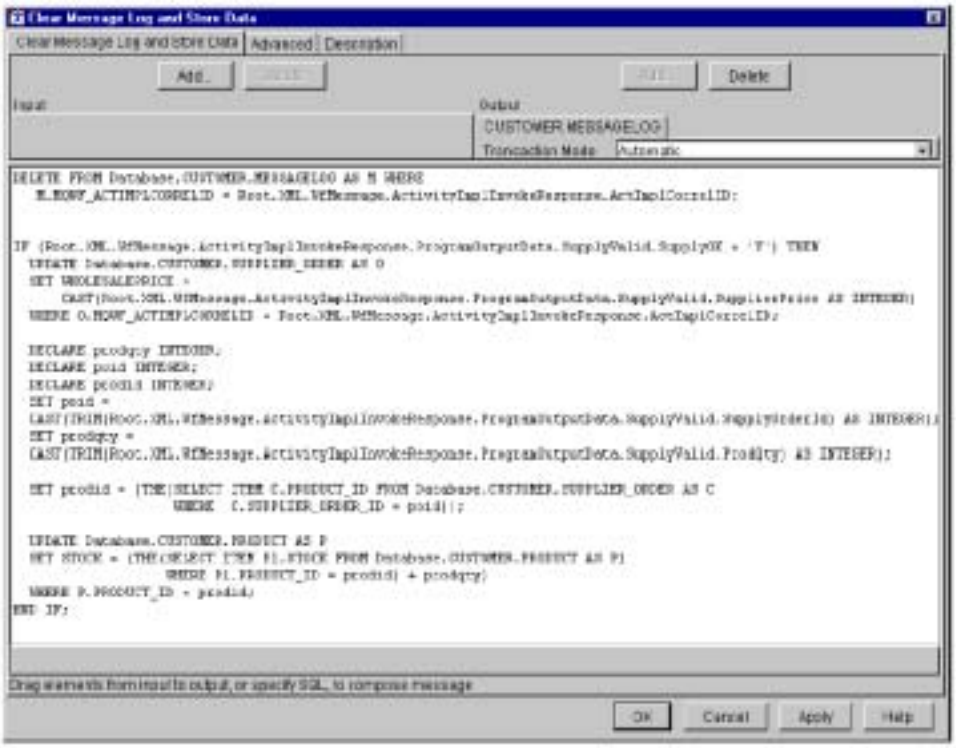

图 5-37 数据库节点细节

在成功回复之后，MESSAGELOG 表中的条目就是多余的了；因而，我们可以从数据库中删除这些信息。该数据库节点将清除相应的记录（参见实例 5-21）。

实例 5-21 数据库节点“清除消息日志和储存数据”(Clear Message Log and Store Data)

```
DELETE FROM Database.CUSTOMER.MESSAGELOG AS M WHERE M.MQWF_ACTIMPLCORRELID =
Root.XML.WfMessage.ActivityImplInvokeResponse.ActImplCorrelID;
```

单独执行该操作，数据库节点将在数据库中保存引入批发价格，并在订单回复成功（StockOK = ‘Y’）时增加产品库存。实例 5-22 显示了所需的 ESQL 语句。

实例 5-22 数据库节点“清除消息日志和储存数据”(Clear Message Log and Store Data)

```
UPDATE Database.CUSTOMER.SUPPLIER_ORDER AS O
  SET WHOLESALEPRICE =CAST(Root.XML.WfMessage.ActivityImplInvokeResponse.
  ProgramOutputData.SupplyOutput.SupplierPrice AS INTEGER)
WHERE O.MQWF_ACTIMPLCORRELID =
  Root.XML.WfMessage.ActivityImplInvokeResponse.ActImplCorrelID;


UPDATE Database.CUSTOMER.PRODUCT AS P
  SET STOCK =(THE(SELECT ITEM P1.STOCK FROM Database.CUSTOMER.PRODUCT AS P1
              WHERE P1.PRODUCT_ID =prodid)+prodqty)
WHERE P.PRODUCT_ID =prodid;
```

以后的操作将完全独立于其他操作。我们已将两个操作合并在一个数据库节点中，但我们也可以使用两个单独的节点。

计算节点“格式输出消息”（Format output message）

图 5-38 显示了该计算节点属性。

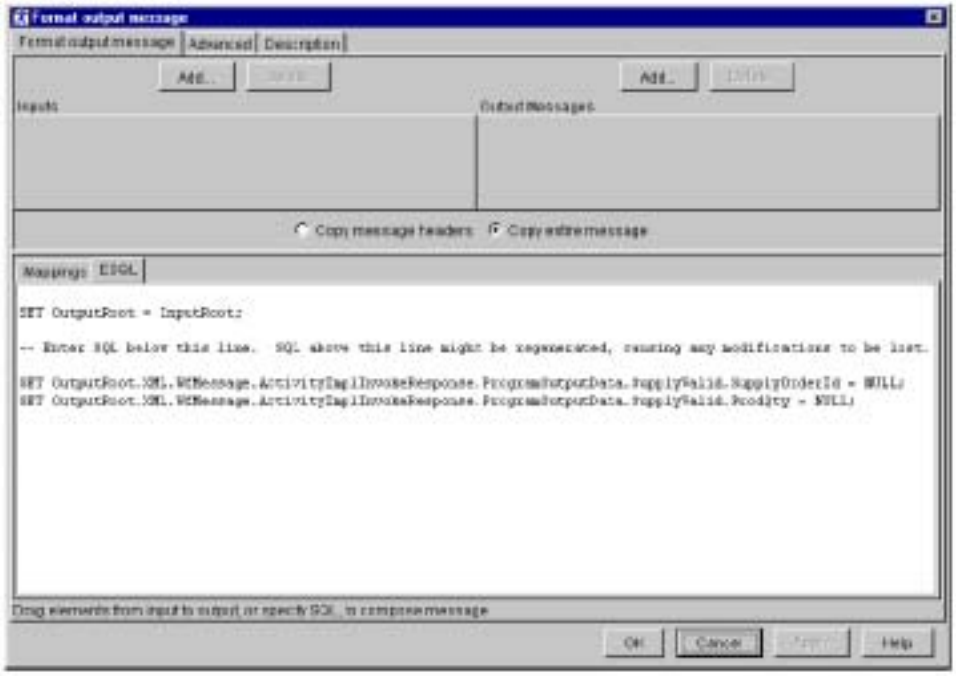

图 5-38 计算节点细节

该计算节点将输入消息复制到输出消息中，并删除临时字段 SupplierOrderId 和 ProdQty。这些字段不是流出消息的一部分。

MQOutput 节点“ 供应订单 ACK OUT ”（Supply Order ACK OUT）

图 5-39 和图 5-40 显示了该 MQOutput 节点的设置。

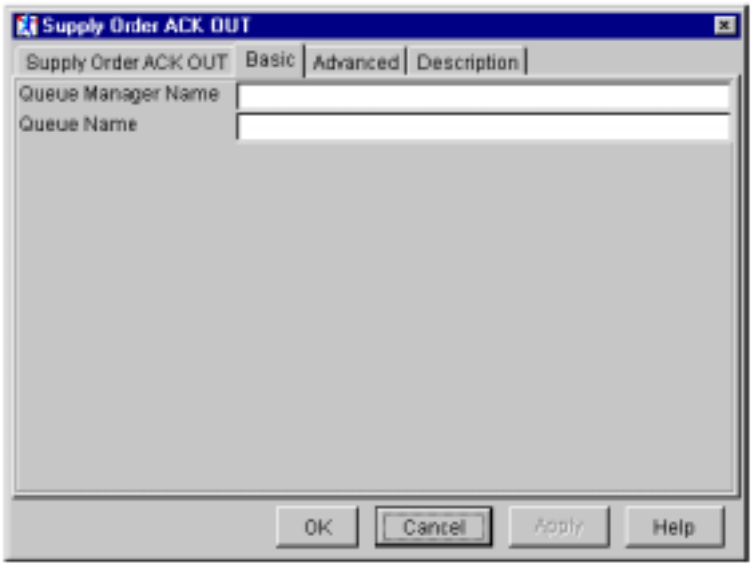

图5-39MQOutput 节点细节

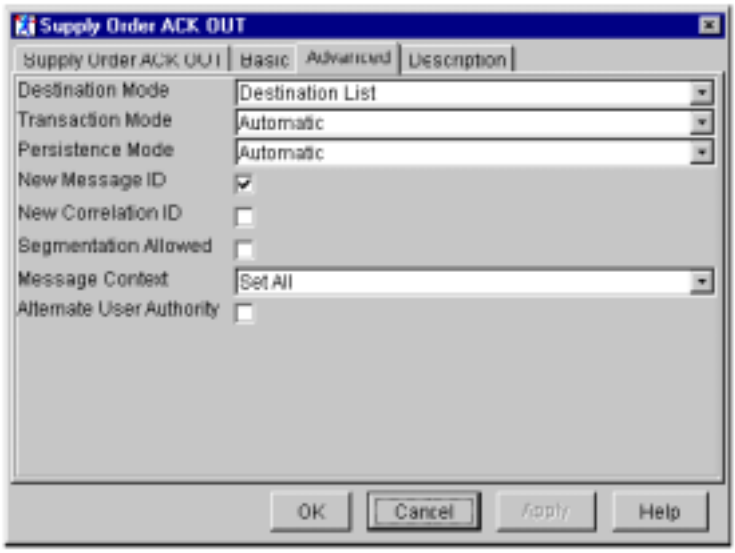

图5-40MQOutput 节点细节

我们使用目的地列表作为目的地模式。队列名和队列管理器名将不在 Basic 标签中指定。我们在消息上下文字段中使用 Set All 参数来重新设置所有 MQMD 字段。

提示：虽然使用的 MQOutput 节点中的 ReplyToQueue 工具似乎是很好的解决方案，但这并不可行。当 MQSeries Workflow 正在 MQMD 标题和 ReplyToQueue 选项中查找原始用户标识符时，MQSeries 将不会设置 MQMD 字段，即使该字段已在计算节点中更新。

MQOutput 节点“ 错误出现 ”（Error Occurred）

图 5-41 和图 5-42 显示了该 MQOutput 节点的基本属性和高级属性。

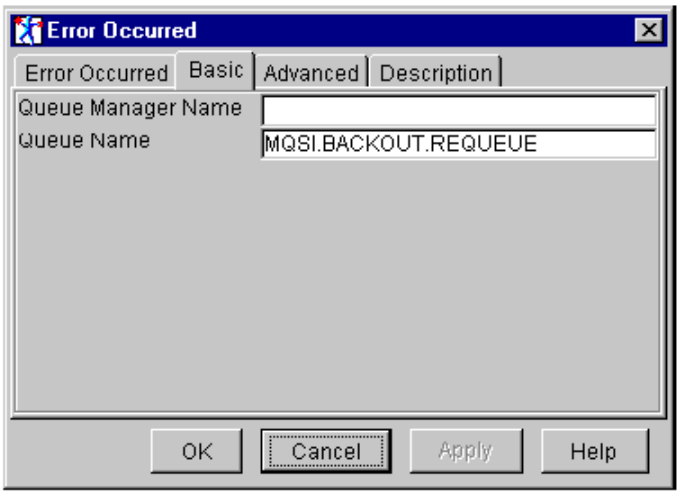

图 5-41MQOutput 节点细节

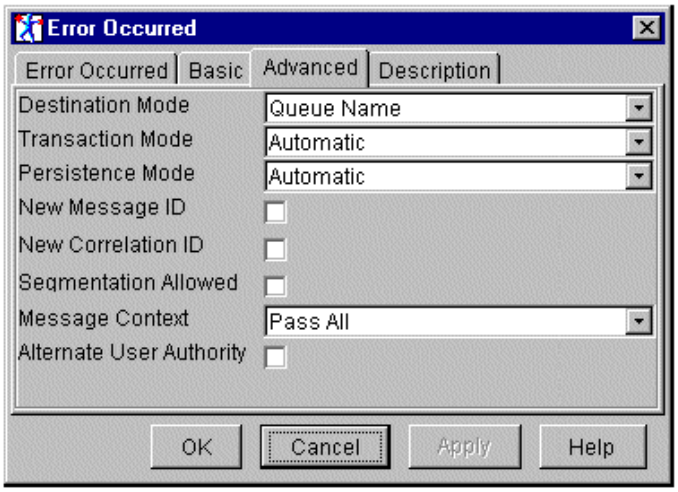

图 5-42MQOutput 节点细节

当创建回复消息失败时，该节点将把这一消息加入错误队列。

5.7 创建BuyXYZ_Order_Entry_CICS消息流

本节将描述消息流 BuyXYZ_Order_Entry_CICS。首先，我们将给出该消息流的功能概览，本节的其余部分将用于描述每个节点的细节，其中包括入站和出站数据结构。

图 5-43 显示了完全的消息流。

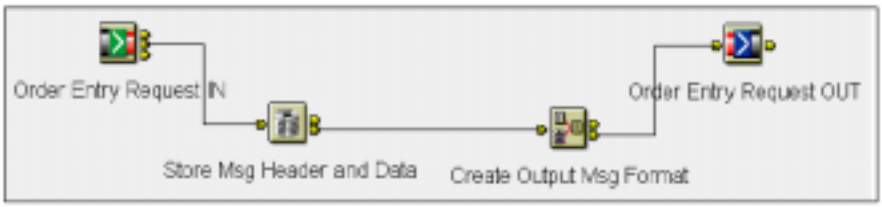

图5-43 消息流概览

消息流步骤如下：

- 从队列 MQSI.INPUT.OE 中读取 MQSeries Workflow XML 消息。
- 将必要的回复信息存储到数据库 CUSTOMER 的表 MESSAGELOG 中。
- 以历史格式（CWF）为 CICS 创建消息。
- 将该消息加入远程队列 SYSTEM.CICS.BRIDGE.QUEUE。

该消息流将以 XML 格式接收来自 MQSeries Workflow 服务器的客户订单条目请求。消息流必须将 XML 消息转换为 CICS 消息历史格式。转换后的消息将由 OS/390 的后端系统接收。

CICS 应用程序是模似程序。它将接受该消息并加入完成代码，然后将其传回队列 MQSI.INPUT.OEACK。它不更新数据库。CICS 程序通过 MQSeries-CICS 桥工具执行。

5.7.1 输入和输出数据结构

XML 消息中的输入和输出数据结构又来源于构建时环境中所做的定义。

输入数据结构 OrderInfo

输入数据由 MQSeries Workflow 生成。输入消息中 ProgramInputData XML 字段包含客户标识符、产品标识符、产品数量、产品价格和产品描述。

XML 输入消息类似实例 5-23 所示。

实例 5-23StockInput 数据结构

```
<ProgramInputData>
  <_ACTIVITY>ValidateCustomer</_ACTIVITY>
  <_PROCESS>Order Process</_PROCESS>
  <_PROCESS_MODEL>Order Process</_PROCESS_MODEL>
  <OrderInfo>
    <CustID>1</CustID>
    <ProdID>1</ProdID>
    <ProdQty>1</ProdQty>
    <ProdPrice>12</ProdPrice>
    <ProdDesc>CD1</ProdDesc>
  </OrderInfo>
</ProgramInputData>
```

提示：该实例仅显示了整个消息中的 ProgramInputData 文件夹。

输出数据结构 Order_Entry

MQSeries Integrator 将把该消息从 XML 格式转换为历史格式。我们需要在 MRM 中定义 Order_Entry 消息以支持这一转换。

图 5-44 显示了 Order_Entry 消息细节。

图 5-44 消息类型定义

提示：您可以导入 Buyxyz_msgset.mrp，它包含本红本书中使用的所有消息。导入命令如下：

```
mqsimrmimpexp -i MRMDataSourceName MRMDataSourceUserId MRMDataSour
cePassword Buyxyz_msgset.mrp
```

5.7.2 消息流细节

本节将描述消息流 BuyXYZ_Order_Entry_CICS 中每个节点的细节。

MQInput 节点“ 订单条目请求 IN ”（Order Entry Request IN）

图 5-45 和图 5-46 显示了该 MQInput 节点的配置窗口。

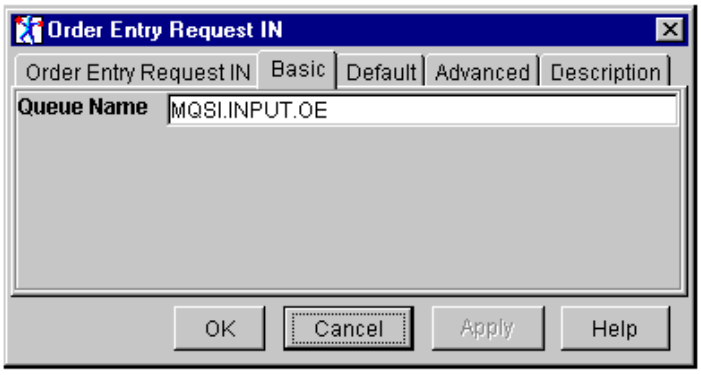

图 5-45MQInput 节点细节

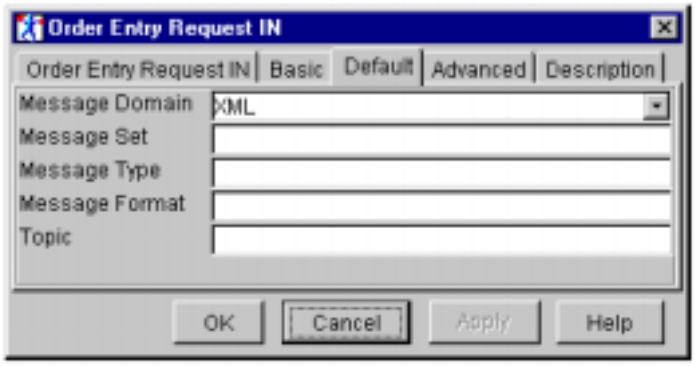

图 5-46MQInput 节点细节

在 MQInput 节点中，我们仅设置以下两个参数：

- 队列名设置为 MQSI.INPUT.OE。
- 默认消息域（Default Message Domain）设置为 XML。

数据库节点“ 储存 Msg 标题和数据 ”（Store Msg Header and Data）

图 5-47 显示了该数据库节点的配置窗口。

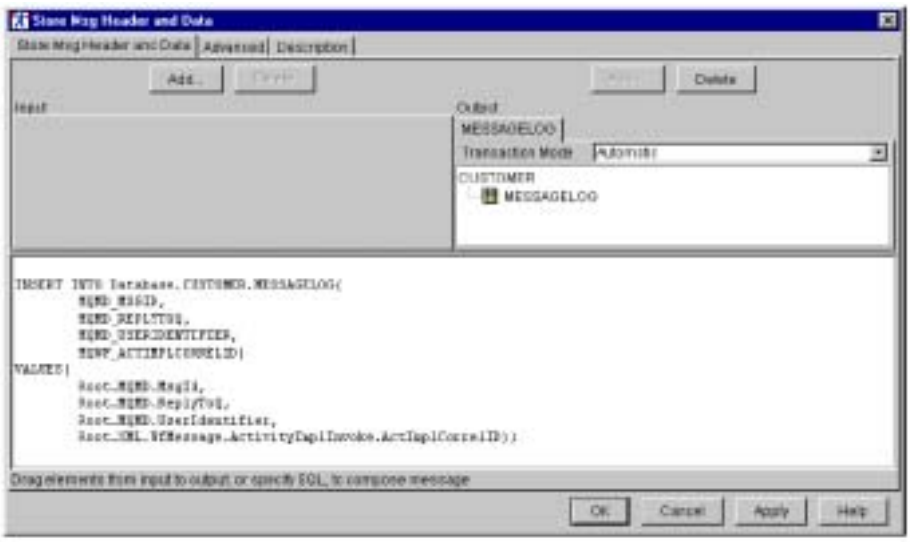

图 5-47 数据库节点细节

该数据库节点将在数据库 CUSTOMER 的 MESSAGELOG 表中插入处理回复消息（来自 CICS 应用程序）所需的数值并生成 MQSeries Workflow 回复消息。存入数据库的数值如下：

- 消息描述符标题数值：

  MsgID： 回复消息流必需的消息标识符，用以匹配和检索表 MESSAGELOG 中相应的消息数据。

  ReplyToQ： 回复队列。这些队列都是群集队列。因此我们仅需存储 reply-to-queue 数值，而不需存储 reply-to-queue 管理器数值。

  UserIdentifier：为 MQSeries Workflow 回复消息所需的数值。

- MQSeries Workflow 标题数值：

  ActImplCorrelId：活动执行相关标识符， UPES 确认消息所需的数值。

实例 5-24 显示了 ESQL 语句：

实例 5-24 数据库节点“储存 Msg 标题和数据”（Store Msg Header and Data）

```
INSERT INTO Database.CUSTOMER.MESSAGELOG(
  MQMD_MSGID,
  MQMD_REPLYTOQ,
  MQMD_USERIDENTIFIER,
  MQWF_ACTIMPLCORRELID)
VALUES(
  Root.MQMD.MsgId,
  Root.MQMD.ReplyToQ,
  Root.MQMD.UserIdentifier,
  Root.XML.WfMessage.ActivityImplInvoke.ActImplCorrelID);
```

我们必须将数据库 CUSTOMER 和表 MESSAGELOG 加入该数据库节点。

计算节点“创建输出 Msg 节点”（Create Output Msg Format）

图 5-48 显示了该计算节点的配置窗口。

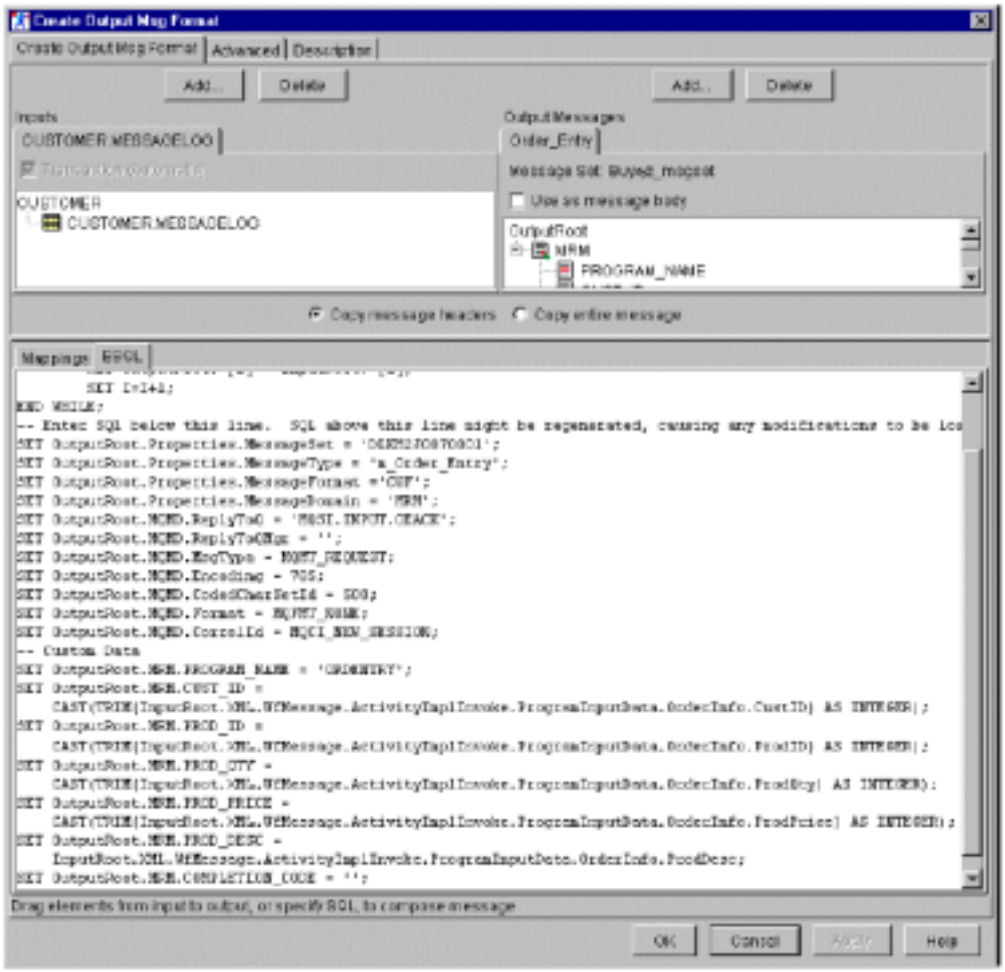

图5-48 计算节点细节

在该计算节点中，我们将设置：

- 消息设置标识符和消息标识符：

实例5-25 计算节点“创建输出Msg 节点”(Create Output Msg Format)

```
SET OutputRoot.Properties.MessageSet ='DLKH2JO07O001';
SET OutputRoot.Properties.MessageType ='m_Order_Entry';
```

- 消息格式和消息域：

实例5-26 计算节点“创建输出Msg 格式”(Create Output tMsg Format)

```
SET OutputRoot.Properties.MessageFormat ='CWF';
SET OutputRoot.Properties.MessageDomain ='MRM';
```

- 回复至队列和回复至队列管理器：

实例5-27　计算节点“Create Output Msg Format ”

```
SET OutputRoot.MQMD.ReplyToQ ='MQSI.INPUT.OEACK';
SET OutputRoot.MQMD.ReplyToQMgr ='';
```

- CICS 接受事务所需的 M QMD 和 MRM 字段。在此，我们强行将数据转换为代码页 500。相关 ID 必须含有数值 MQCI_NEW_SESSION，以便通知 CICS 桥该消息不是现有事务的一部分。CICS 桥可以接收两种消息类型。第一种消息类型以 CICS 桥标题开始；第二种类型没有该标题。如果是第二种类型，需要在消息的前八个字节中指定 CICS 桥必须运行的程序名。

实例5-28 计算节点“创建输出Msg 格式”(Create Output Msg Format)

```
SET OutputRoot.MQMD.MsgType =MQMT_REQUEST;
SET OutputRoot.MQMD.Encoding =785;
SET OutputRoot.MQMD.CodedCharSetId =500;
SET OutputRoot.MQMD.Format =MQFMT_NONE;
SET OutputRoot.MQMD.CorrelId =MQCI_NEW_SESSION;
SET OutputRoot.MRM.PROGRAM_NAME ='ORDENTRY';
```

在 ESQL 的第二部分，我们将指定转换字段。我们将添加 COMPLETION_CODE 字段，该字段由 CICS 应用程序设置，用以描述事务处理成功。

实例5-29 计算节点“创建输出Msg 格式”(Create Output Msg Format)

```
SET OutputRoot.MRM.CUST_ID =
  CAST(TRIM(InputRoot.XML.WfMessage.ActivityImplInvoke.
  ProgramInputData.OrderInfo.CustID)AS INTEGER);
SET OutputRoot.MRM.PROD_ID =
  CAST(TRIM(InputRoot.XML.WfMessage.ActivityImplInvoke.
  ProgramInputData.OrderInfo.ProdID)AS INTEGER);
SET OutputRoot.MRM.PROD_QTY =
  CAST(TRIM(InputRoot.XML.WfMessage.ActivityImplInvoke.
  ProgramInputData.OrderInfo.ProdQty)AS INTEGER);
SET OutputRoot.MRM.PROD_PRICE =
  CAST(TRIM(InputRoot.XML.WfMessage.ActivityImplInvoke.
  ProgramInputData.OrderInfo.ProdPrice)AS INTEGER);
```

```
SET OutputRoot.MRM.PROD_DESC =
  InputRoot.XML.WfMessage.ActivityImplInvoke.ProgramInputData.
  OrderInfo.ProdDesc;
SET OutputRoot.MRM.COMPLETION_CODE ='';
```

请注意，我们添加了数据库 CUSTOMER 的表 MESSAGELOG 作为输入 ODBC 源，然后添加了 Order_EntryMRM 定义作为输出。

MQOutput 节点“ Order Entry Request OUT ”

图 5-49 显示了该 MQOutput 节点的配置窗口。

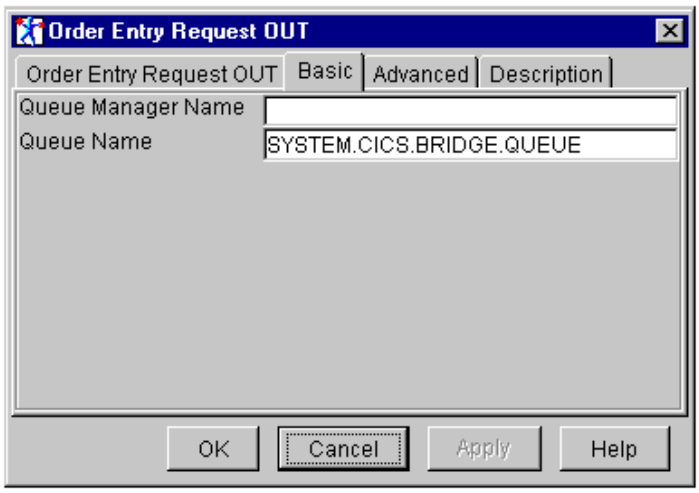

图 5-49 过滤节点细节

在该 MQOutput 节点中，我们将只把队列名设置 SYSTEM.CICS.BRIDGE.QUEUE。这事实上是队列管理器 MQGV（在 OS/390 系统上）上的一个远程队列定义。

5.8 创建BuyXYZ_Order_Entry_CICSACK消息流

本节将描述消息流 BuyXYZ_Order_Entry_CICSACK。第一部分将给出该消息流的功能概览，其余部分用于描述每个节点的细节，其中包括入站和出站数据结构。

图 5-50 显示了完全的消息流。

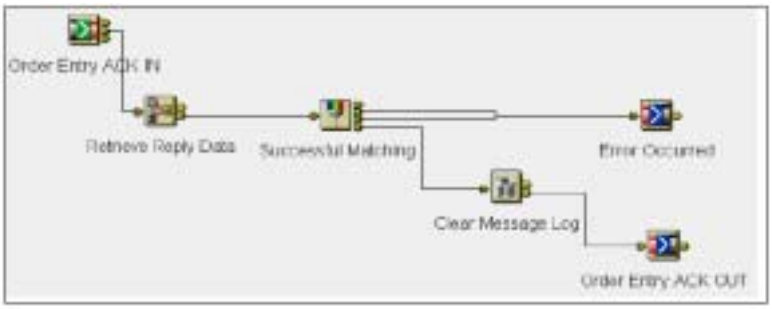

图5-50 消息流概览

消息流步骤如下：

- 从队列 MQSI.INPUT.OEACK 中读取 CICS 应用程序回复消息。
- 从数据库 CUSTOMER 的 MESSAGELOG 表中检索先前存储的数据。
- 如数据检索成功，登记过滤节点。否则，将消息传递到 MQOutput 节点，该节点将其加入群集队列 MQSI.BACKOUT.REQUEUE 中。
- 为 MQSeries Workflow 创建一条 XML 格式的消息。
- 将该消息加入 ReplyToQ 队列以便使执行服务器可以读取它。

消息流以历史格式接收来自 CICS 应用程序的消息，然后将其转换为 XML 格式。为了完成此转换并用正确的数值填写该 XML 字段，消息流将从数据库 CUSTOMER 的 MESSAGELOG 表中检索数据。

如果数据检索失败，该消息将传递至 MQOutput 节点，该节点将其加入群集队列 MQSI.BACKOUT.REQUEUE 中。

如果数据检索成功，该消息将加入 ReplyToQ 队列，然后从数据库中将回复流数据删除。

输入数据结构 Order_Entry

从 CICS 应用程序到 MQSI BuyXYZ_Order_Entry_CICSACK 消息流的输入数据结构与从 MQSI BuyXYZ_Order_Entry_CICS 消息流到 CICS 应用程序的输出数据结构是相同的。请参考第 227 页的“ 输出数据结构 Order_Entry ”。

输出数据结构 OrderInfo

在 MQSeries Workflow 中，无论输入还是输出，我们都使用同一 OrderInfo 数据结构。在回复 MQSeries Workflow 的消息中，我们将添加完成代码（ProgramRC）。该代码将告诉我们事务处理是否成功完成。

从 MQSeries Integrator 到 MQSeries Workflow 的 XML 输出消息实例如实例 5-30 所示：

实例 5-30OrderInfo 数据结构

```
<WfMessage>
  <WfMessageHeader>
    <ResponseRequired>No</ResponseRequired>
  </WfMessageHeader>
  <ActivityImplInvokeResponse>
    <ActImplCorrelID>RUEAAAABABtAEAAAAAAAAAAAAAAABg</ActImplCorrelID>
    <ProgramOutputData>
      <OrderInfo>
        <CustID>1</CustID>
        <ProdID>1</ProdID>
        <ProdQty>1</ProdQty>
        <ProdPrice>1</ProdPrice>
        <ProdDesc>CD1</ProdDesc>
      </OrderInfo>
    </ProgramOutputData>
    <ProgramRC>####################</ProgramRC>
  </ActivityImplInvokeResponse>
</WfMessage>
```

5.8.1 消息流细节

我们现在来看一下消息流 BuyXYZ_Order_Entry_CICSACK 中每个节点的细节。

MQInput 节点“ 订单条目 ACK IN ”（Order Entry ACK IN）

图 5-51 和图 5-52 显示了该 MQInput 节点的配置窗口。

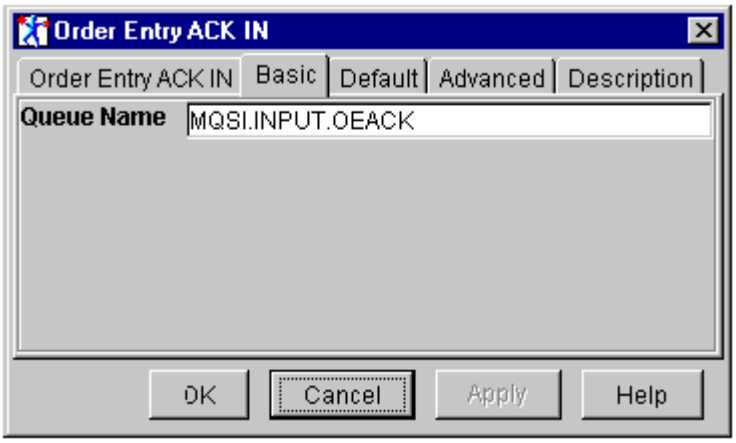

图 5-51MQInput 节点细节

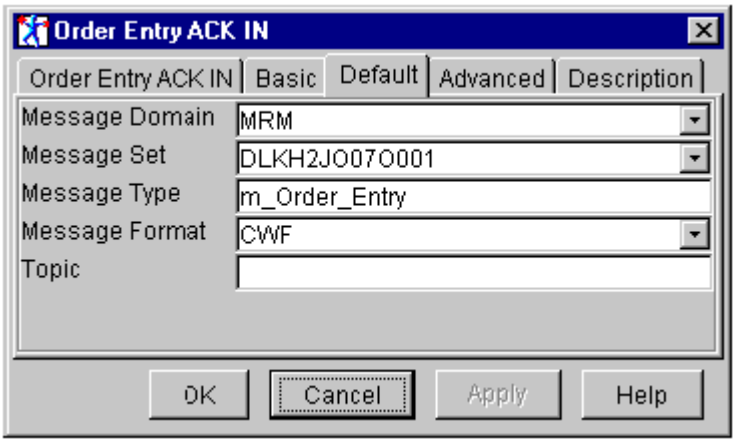

图 5-52MQInput 节点细节

在该节点中，我们将设置以下参数：

- 将队列名设置为 MQSI.INPUT.OEACK。
- 将默认消息域设置为 MRM。
- 指定默认消息集，本案例中为 DLKH2JO07O001。
- 指定默认消息类型（标识符）。本案例中为 m_Order_Entry。
- 将默认消息格式设置为 CWF。

计算节点“检索回复数据”（Retrieve Reply Data）

图 5-53 和图 5-54 显示了该计算节点的配置窗口。

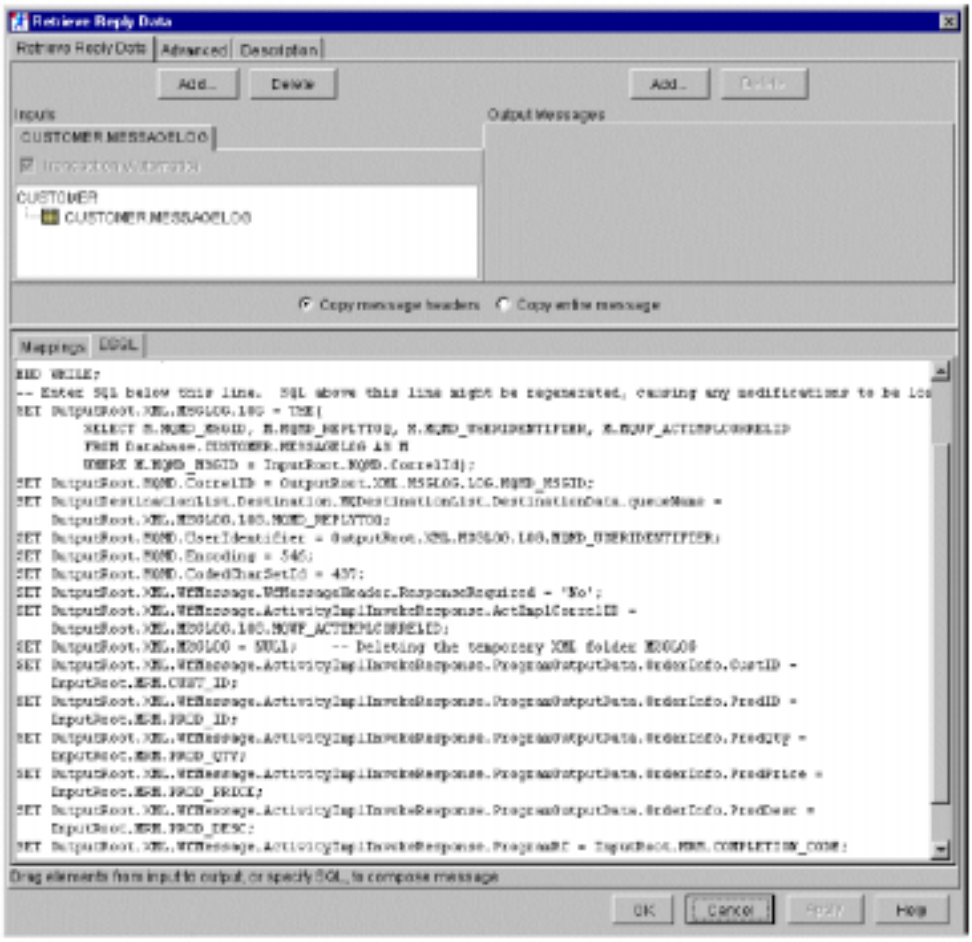

图 5-53 计算节点细节

在数据库节点中，我们必须从 MESSAGELOG 表中检索消息数据并将其分配到临时 XML 输出消息日志中：

实例 5-31 计算节点“检索回复数据”（Retrieve Reply Data）

```
SET OutputRoot.XML.MSGLOG.LOG =THE(
  SELECT M.MQMD_MSGID,
    M.MQMD_REPLYTOQ,
```

```
    M.MQMD_USERIDENTIFIER,
在MQSeries Integrator的目的地列表中设置队列名，
从而把消息加入MQSeries Workflow的reply-to-queue中。
```

实例5-32 计算节点“检索回复数据”（Retrieve Reply Data）

```
SET OutputDestinationList.Destination.MQDestinationList.
  DestinationData.queueName =OutputRoot.XML.MSGLOG.LOG.MQMD_REPLYTOQ;
```

将 MQMD 字段设置为消息从 MQSeries Workflow 发来时所含有的数值

实例5-33 计算节点“检索回复数据”（Retrieve Reply Data）

```
SET OutputRoot.MQMD.UserIdentifier =
  OutputRoot.XML.MSGLOG.LOG.MQMD_USERIDENTIFIER;
SET OutputRoot.MQMD.Encoding =546;
SET OutputRoot.MQMD.CodedCharSetId =437;
```

创建 MQSeries Workflow 消息标题。尤其重要的是设置 ActImplCorrelID，以便使执行服务器能够确定该应答是对哪一活动的回复。

实例5-34 计算节点“检索回复数据”（Retrieve Reply Data）

```
SET OutputRoot.XML.WfMessage.WfMessageHeader.ResponseRequired='No';
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.
  ActImplCorrelID =OutputRoot.XML.MSGLOG.LOG.MQWF_ACTIMPLCORRELID;
```

重置 XML 消息日志文件夹，以便使其在流出消息中不被传递。

实例5-35 计算节点“Retrieve Reply Data”

```
SET OutputRoot.XML.MSGLOG =NULL;
```

在 ESQL 的第二部分中，我们将指定转换字段：

实例5-36 计算节点“检索回复数据”（Retrieve Reply Data）

```
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.
  OrderInfo.CustID =InputRoot.MRM.CUST_ID;
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.
  OrderInfo.ProdID =InputRoot.MRM.PROD_ID;
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.
  OrderInfo.ProdQty =InputRoot.MRM.PROD_QTY;
```

```
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.
  OrderInfo.ProdPrice =InputRoot.MRM.PROD_PRICE;
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramOutputData.
  OrderInfo.ProdDesc =InputRoot.MRM.PROD_DESC;
SET OutputRoot.XML.WfMessage.ActivityImplInvokeResponse.ProgramRC =
  InputRoot.MRM.COMPLETION_CODE;
```

请注意，我们添加了数据库 CUSTOMER 的 MESSAGELOG 表作为输入 ODBC 数据源。

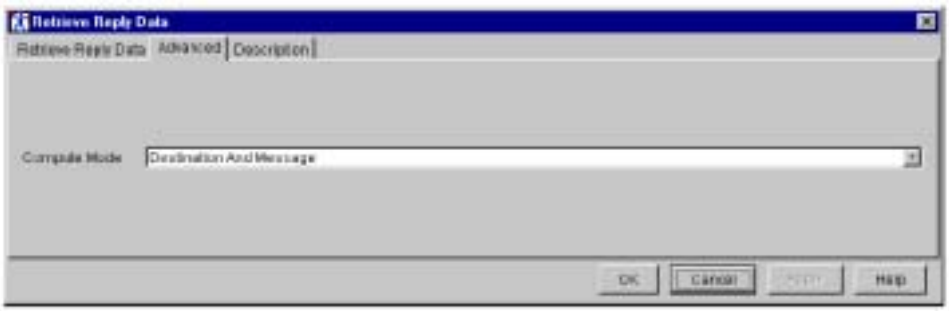

图5-54 计算节点细节

如图 5-54 所示，在计算节点的 Advanced 标签中，我们选择了目的地和消息（Destination and Message）以确保节点 MQOuput 将消息发送到正确的队列。

过滤节点“ 成功匹配 ”（Successful Matching）

图 5-55 显示了该过滤节点的配置窗口。

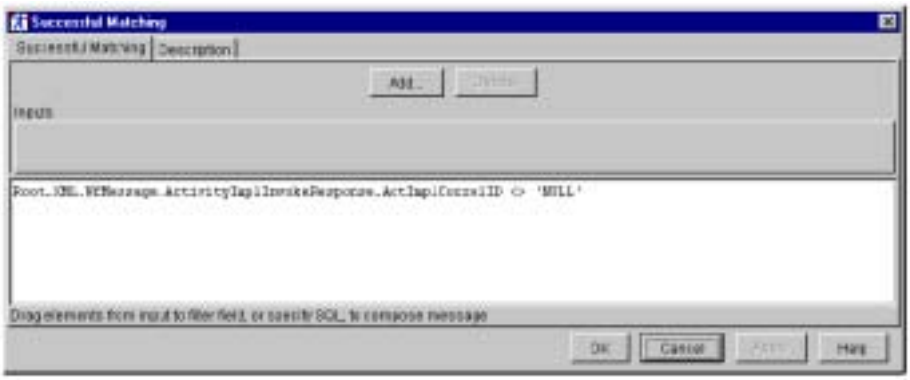

图5-55 过滤节点细节

在该过滤节点中，我们将检查数据库检索是否完成：

- 如数据库检索失败，该消息将在 MQSI.BACKOUT.REQUEUE 队列中终止。
- 如成功，该消息将被传给数据库节点清除消息日志。

数据库节点“ 清除消息日志 ”（Clear Message Log）

图 5-56 显示了该数据库节点的配置窗口。

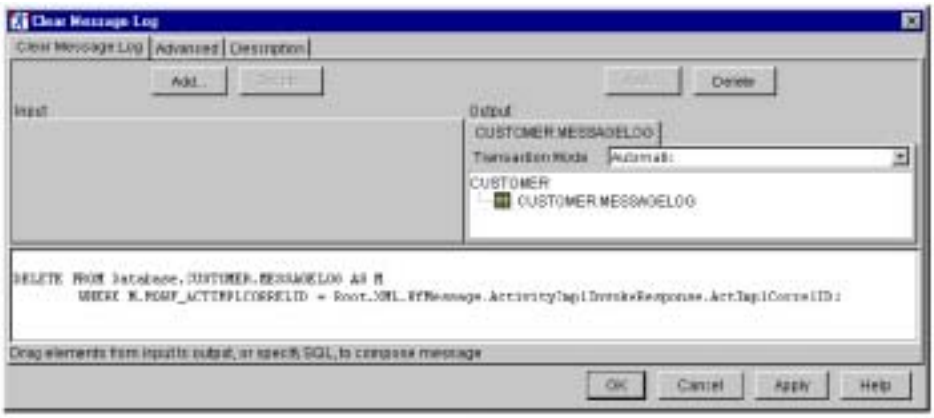

图 5-56 数据库节点细节

消息数据已检索成功，我们将不再需要它。因此，我们可以用以下语句将其删除：

实例 5-37 数据库节点“清除消息日志”(Clear Message Log)

```
DELETE FROM Database.CUSTOMER.MESSAGELOG AS M
WHERE M.MQWF_ACTIMPLCORRELID =
  Root.XML.WfMessage.ActivityImplInvokeResponse.ActImplCorrelID;
```

MQOutput 节点“订单条目 ACK OUT”（Order Entry ACK OUT）

图 5-57 和图 5-58 显示了该 MQOutput 节点的配置窗口。

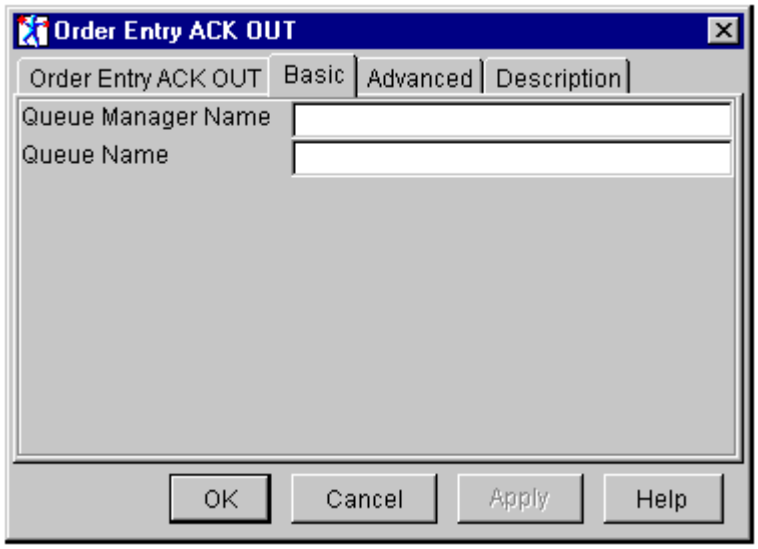

图 5-57MQOutput 节点细节

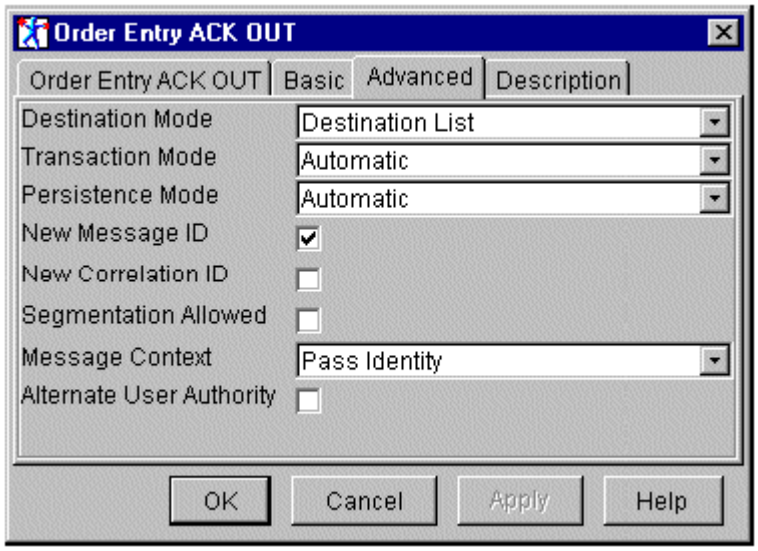

图 5-58MQOutput 节点细节

在该 MQOutput 节点中，我们不需设置队列管理名或队列名字段，因为它们将由目的地列表设置。在高级文件夹中，我们将目的地模式设置为目的地列表，将消息上下文设置为 Pass Identity。

MQOutput 节点“ Error Occurred ”

图 5-59 显示了该 MQOutput 节点的配置窗口。

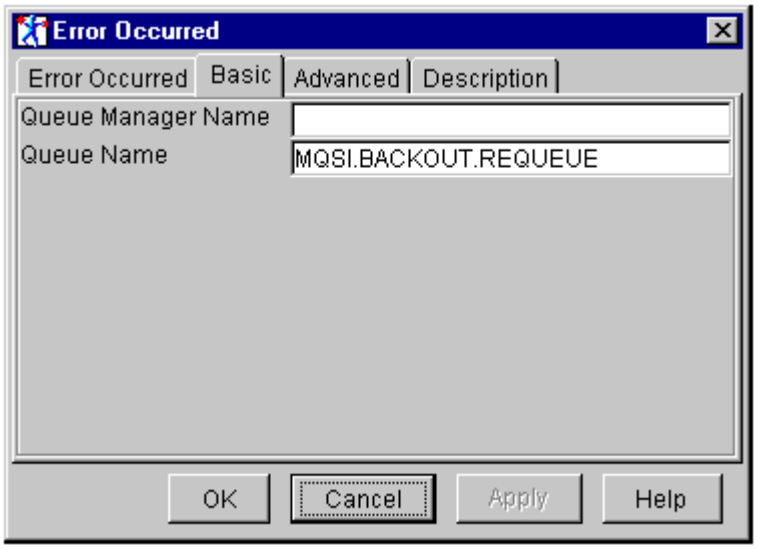

图 5-59 MQOutput 节点细节

在该 MQOutput 节点中，我们只设置群集队列名 MQSI.BACKOUT.REQUEUE。假设从数据库读取数据时出现错误，该队列将是错误消息所放的位置。

5.9 安装实例应用程序

让我们对迄今为止所做的工作做一次快速概述：

- 在第三章“技术组件的配置”（第 17 页）中，我们描述了 MQSeries、MQSeries Workflow 和 MQSeries Integrator 的安装和配置。我们也使用“WebCredit”实例检验了 MQSeries Workflow 的配置，并且使用一个简单的消息流检验了 MQSeries Integrator 的配置。
- 在第四章“在 MQSeries Workflow 中实现模型”（第 109 页）中，我们创建了只含有一个供应商的简单案例 BuyXYZ。
- 本章中，我们在 MQSeries Integrator 中创建了应用数据库和必要的消息流，以便在工作流中执行 UPES 活动。

现在是安装测试简单案例 BuyXYZ 所需的所有其它资源的时候了。

5.9.1 定义MQSeries对象

为了测试此简单案例 BuyXYZ，我们必须定义另外一些 MQSeries 对象。

这些对象是：

1. 群集队列。我们必须在 MQSI01QM 和 MQSI02QM 队列管理器上定义群集队列。以下参数对于所有群集队列都是相同的：
   - 所有队列都是持久的。
   - 默认绑定设置为“ NOTFIXED ”以实现工作负荷均衡。
   - 逆序恢复极限设置为 3。
   - 逆序恢复队列名设置为“ MQSI.BACKOUT.REQUEUE ”。

该队列的定义实例如下：

```
DEFINE QLOCAL (MQSI.INPUT.VC)+
  DESCR ( ‘ Input for Validate_Customer msg flow -from MQWF ’ )+|
  CLUSTER (BUYXYZ)+
  DEFPSIST (YES)+
  DEFBIND (NOTFIXED)+
  BOTHRESH (3)+
  BOQNAME (MQSI.BACKOUT.REQUEUE)+
  REPLACE
```

我们必须在两个代理队列管理器上定义如表 5-1 所示的队列。注意，该列表包含了我们用于 UPES 执行的一些队列。这将在第六章“ 使用 MQSeries Adapter Offering 创建 BOD 消息 ”（257 页）中以及第七章“ 在工作流中调用 Enterprise JavaBean ”（301 页）中讨论。

表 5-1 代理队列管理器上定义的队列列表

队列名	队列描述
MQSI.DUMMY	从 MQWF 为 BuyXYZ_Dummy 消息流输入队列。
MQSI.INPUT.VC	从 MQWF 为 BuyXYZ_Validate_Customer 消息流输入队列。
MQSI.INPUT.VS	从 MQWF 为 BuyXYZ_Validate_Stock 消息流输入队列。
MQSI.INPUT.SO	从 MQWF 为 BuyXY_Supply_Order_PO 消息流输入队列。
MQSI.INPUT.POACK	从 MQAO 为 BuyXY_Supply_Order_OE 消息流输入队列。
MQSI.INPUT.OE	从 MQWF 为消息流输入队列。
MQSI.INPUT.OEACK	从 CICS 为 BuyXY_Supply_Order_OEACK 消息流输入队列。

队列名	队列描述
MQSI.INPUT.SH	从 MQWF 输入 Shipping EJB 的队列。
MQAO.INPUT.BS	从 MQWF 输入 Buyxyz_Backup_Supplier msg 流的队列
PAM.INPUT.PO	从 MQAO.输入 BOD Purchase Order 的队列
MQAO.INPUT.POACK	从 DOS 程序（未来的）输入 BOD Purchase Order ACK 的队列
PAM.INPUT.PO	从 MQAO 输入 BOD Purchase Order 的队列

2. 通道。我们必须创建从两个代理设备到 CICS 后台的发送—接收对，反之亦然。

下面是一个在设备 MQSI01QM 上创建通道的实例脚本：

```
DEFINE CHANNEL (MQSI01QM.TO.MQGV)+
  CHLTYPE (SDR)+
  TRPTYPE (TCP)+
  CONNAME (9.12.14.204)+
  XMITQ (MQGV)+
  DESCR ( ‘ Sender channel from MQSI01QM to MQGV ’ )+
  REPLACE
DEFINE CHANNEL (MQGV.TO.MQSI01QM)+
  CHLTYPE (RCVR)+
  TRPTYPE (TCP)+
  DESCR ( ‘ Receiver channel from MQGV to MQSI01QM ’ )+
  REPLACE
```

在队列管理器 MQSI01QM 上，我们必须指定如表 5-2 所示的数值。

表 5-2 在队列管理器 MQSI01QM 上定义的通道

通道名	Chl.类型	通道描述
MQSI01QM.TO.MQGV	发送器	发送器通道从 MQGV 到 MQSI01QM
MQGV.TO.MQSI01QM	接收器	接收器通道从 MQGV 到 MQSI01QM

在队列管理器 MQSI02QM 上，我们必须指定如表 5-3 所示的数值。

表 5-3 在队列管理器 MQSI02QM 上定义的通道

通道名	Chl.类型	通道描述
MQSI02QM.TO.MQGV	发送器	发送器通道从 MQSI02QM 到 MQGV

通道名	Chl.类型	通道描述
MQGV.TO.MQSI02QM	接收器	从 MQGV 到 MQSI02QM 接收器通道

在队列管理器 MQGV 上，我们必须定义：

表 5-4 在队列管理器 MQGV 上定义的通道

通道名	Chl.类型	通道描述
MQGV.TO.MQSI01QM	发送器	从 MQGV 到 MQSI01QM 的发送器通道
MQSI01QM.TO.MQGV	接收器	从 MQSI01QM 到 MQGV 的接收器通道
MQGV.TO.MQSI02QM	发送器	从 MQGV 到 MQSI02QM 的发送器通道
MQSI02QM.TO.MQGV	接收器	从 MQSI02QM 到 MQGV 的接收器通道

3. 远程队列。我们必须定义从两个代理队列管理器到队列 MQGV 的远程队列，反之亦然。

   下面是在队列管理器 MQSI01QM 或 MQSI02QM 上定义远程队列的实例定义：

```
DEFINE QREMOTE (SYSTEM.CICS.BRIDGE.QUEUE)+
  DESCR ( ‘ Remote queue to queue manager MQGV on OS/390 ’ )+
  DEFPSIST (YES)+
  DEFBIND (NOTFIXED)+
  RNAME (SYSTEM.CICS.BRIDGE.QUEUE)+
  RQMNAME (MQGV)+
  XMITQ (MQGV)+
  REPLACE
```

   在队列管理器 MQGV 上，我们不需要指定远程队列，因为消息中含有 MQMD 中的 reply-to-queue 信息和正确设置的传送队列，也就是说远程队列管理器名即传送队列名。

4. 传送队列。我们必须在两个代理和队列管理器 MQGV 上定义传送队列。

   下面是在队列管理器 MQSI01QM 或 MQSI02QM 上定义传送队列的实例定义：

```
DEFINE QLOCAL (MQGV)+
  DESCR ( ‘ Transmission queue to queue manager MQGV on OS/390 ’ )+
  DEFPSIST (YES)+
  DEFBIND (NOTFIXED)+
  USAGE (XMITQ)+
```

```
TRIGGER +
TRIGTYPE (FIRST)+
INITQ (SYSTEM.CHANNEL.INITQ)+
REPLACE
```

为了使用通道启动程序来启动从 MQGV 到代理队列管理器的通道，我们需要定义过程并将其与传送队列定义相关联。

5. 改变 MQSeries Workflow 队列管理器队列定义，使其包含默认绑定选项 NOTFIXED。这些队列管理器是 WF01QM 、WF01QM 和 WASQM。每一队列管理器都包含一组必须修改的队列设置。关于这些队列的详细说明，请参看附录 B “ 实例程序安装 ”（第 397 页）。

提示：关于所有 MQSeries 对象定义脚本，请参看附录 B “ 实例程序安装 ”（397 页）。脚本名可在表 5-5 中找到。

创建 MQSeries 对象的脚本列表如表 5-5 所示。

表 5-5 MQSeries 对象创建脚本

脚本名	描述
buyxyz_mqsi01qm.cfg	MQSI01QM 队列管理器的队列定义
buyxyz_mqsi02qm.cfg	MQSI02QM 队列管理器的队列定义
buyxyz_wasqm.cfg	WASQM 队列管理器的队列定义
buyxyz_wf01qm.cfg	WF01QM 队列管理器的队列定义
buyxyz_wf02qm.cfg	WF02QM 队列管理器的队列定义

请注意，我们可以在相应的队列管理器上使用工具 “ runmqsc ” 来运行这些脚本。

5.9.2 创建应用数据库

我们将提供不同的脚本，以便在设备 M23BZZYP 的现有数据库的 CUSTOMER 中创建新表并在这些表中插入实例数据。

表 5-6 MQSeries 对象创建脚本

脚本名	描述
buyxyz_ddl.txt	此脚本包含了 SQL 命令以便在现存 CUSTOMER 数据库中创建表格。

脚本名	描述
buyxyz_data.txt	针对客户、供应商和产品的脚本实例数据。

表定义如实例 5-36 所示。

表 5-38CUSTOMER 数据库表

```
CREATE TABLE CUSTOMER.CUSTOMER_ORDER(
  ORDER_ID CHAR(80)NOT NULL,
  CUSTOMER_ID INTEGER,
  PRODUCT_ID INTEGER,
  ORDER_QTY INTEGER,
  ORDER_PRICE INTEGER,
  SHIPPING_NAME VARCHAR(50),
  SHIPPING_ADDRESS VARCHAR(100),
  PRIMARY KEY (ORDER_ID))
IN USERSPACE1

CREATE TABLE CUSTOMER.PRODUCT(
  PRODUCT_ID INTEGER GENERATED ALWAYS AS IDENTITY
    (START WITH 1,INCREMENT BY 1,CACHE 20),
  DESCRIPTION VARCHAR(50),
  STOCK INTEGER,
  RETAILPRICE INTEGER,
  PRIMARY KEY (PRODUCT_ID))
IN USERSPACE1

CREATE TABLE CUSTOMER.SUPPLIER(
  SUPPLIER_ID INTEGER GENERATED ALWAYS AS IDENTITY
    (START WITH 1,INCREMENT BY 1,CACHE 20),
  SUPPLIER_NAME VARCHAR(255),
  DESCRIPTION VARCHAR(255),
  PRIMARY KEY (SUPPLIER_ID))
IN USERSPACE1

CREATE TABLE CUSTOMER.SUPPLIER_PRODUCT(
  PRODUCT_ID INTEGER NOT NULL,
  SUPPLIER_ID INTEGER NOT NULL,
  PRIMARY_SUPPLIER CHAR(1),
  PRIMARY KEY (PRODUCT_ID,SUPPLIER_ID))
IN USERSPACE1

CREATE TABLE CUSTOMER.SUPPLIER_ORDER(
  SUPPLIER_ORDER_ID INTEGER GENERATED ALWAYS AS IDENTITY
    (START WITH 1,INCREMENT BY 1,CACHE 20),
  SUPPLIER_ID INTEGER,
  PRODUCT_ID INTEGER,
```

```
  QTY INTEGER,
  WHOLESALEPRICE INTEGER,
  MQWF_ACTIMPLCORRELID VARCHAR(80),
  PRIMARY KEY (SUPPLIER_ORDER_ID))
IN USERSPACE1

CREATE TABLE CUSTOMER.MESSAGELOG(
  MQWF_ACTIMPLCORRELID VARCHAR(80)NOT NULL,
  MQMD_MSGID CHAR(24),
  MQMD_REPLYTOQ VARCHAR(48),
  MQMD_USERIDENTIFIER VARCHAR(12),
  SUPPLIER_ORDER_ID INTEGER,
  PRIMARY KEY (MQWF_ACTIMPLCORRELID))
IN USERSPACE1
```

5.9.3 创建远程ODBC连接

为了能够访问数据库，我们必须创建从两个代理设备到数据库 CUSTOMER 的一个 ODBC 连接。

这样做最容易的方法就是通过“ DB2 客户机配置助理实用程序 ”来创建该连接。

启动该实用程序并单击 Add，将弹出如图 5-60 所示的窗口。

图 5-60 添加数据库向导——步骤 1

选择手动配置到数据库的连接（Manually configure a connection to a database）并单击下一步（Next），将出现如图 5-61 所示的窗口。

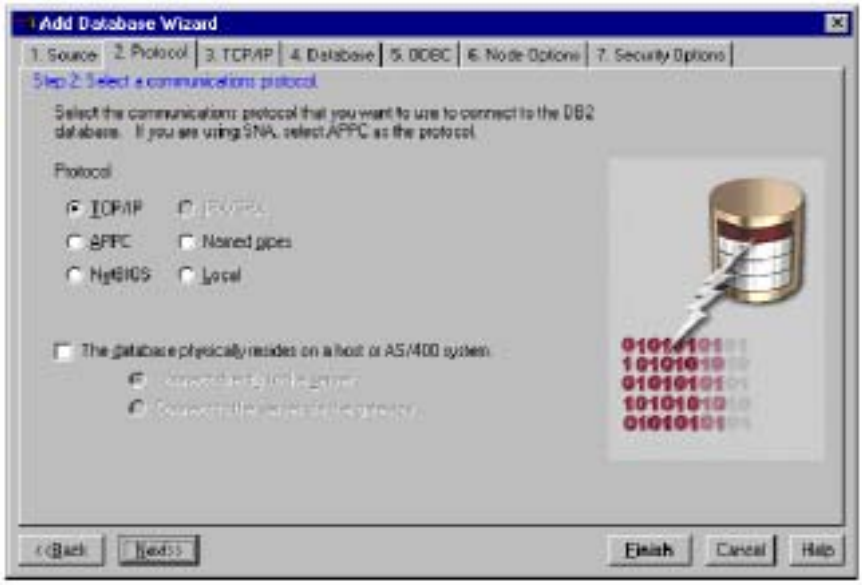

图5-61 添加数据库向导——步骤2

选择 TCP/IP 协议并单击下一步（Next），将出现如图 5-62 所示的窗口。

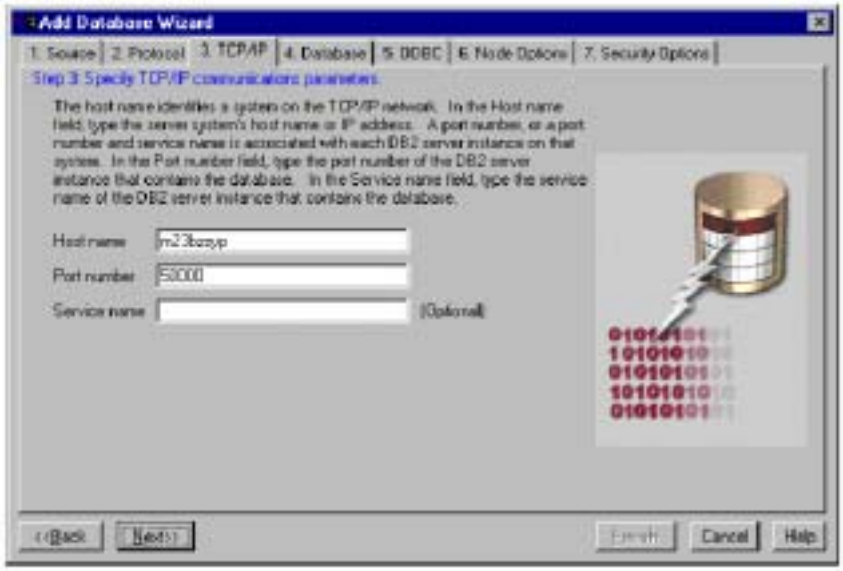

图5-62 添加数据库向导——步骤3

对于主机名，键入驻留数据库“ CUSTOMER ”的设备名。DB2 端口号默认为 50000。单击下一步（Next），将弹出如图 5-63 所示的窗口。

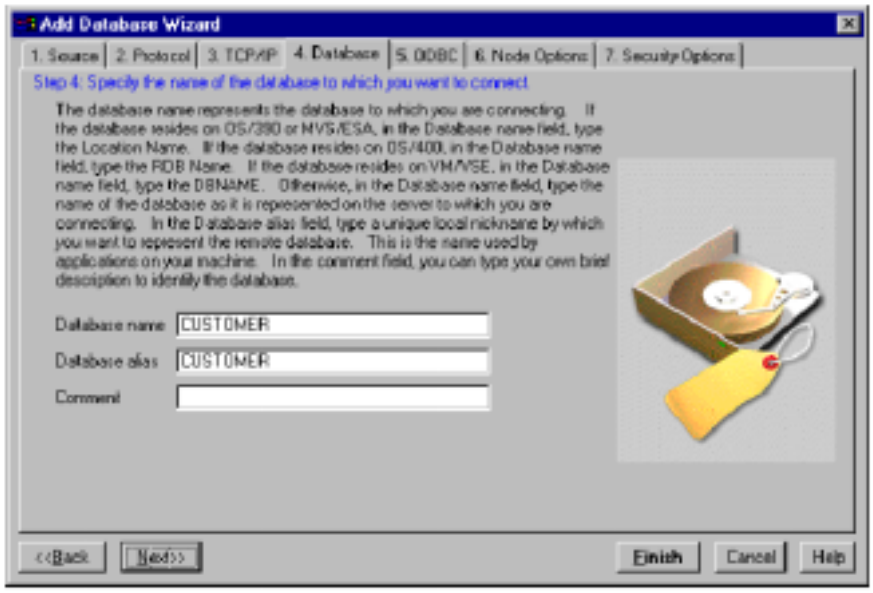

图 5-63 添加数据库向导——步骤 4

数据库名键入 CUSTOMER；数据库别名应键入下面将注册的 ODBC 驱动器名。在本案例中，我们可以使用默认名 CUSTOMER。单击下一步（Next），将出现如图 5-64 所示的窗口。

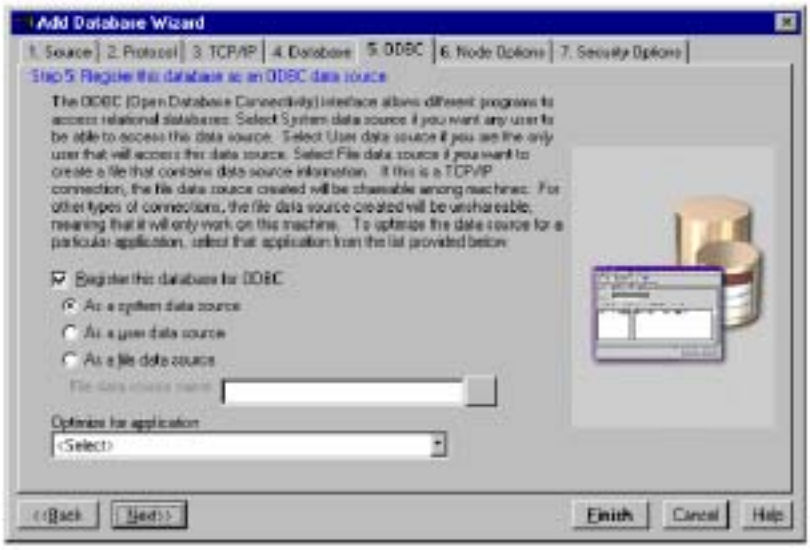

图5-64 添加数据库向导——步骤5

单击完成（Finish）。我们可以测试该连接。如测试成功，则说明 ODBC 连接已成功添加。

5.9.4 测试实例应用程序

该应用程序的体系结构非常复杂，很可能会在测试过程中出现问题，特别是集成测试，因为其中包含许多组件，包括在 MQSI、MQAO 和 EJBs 中的 UPES 活动执行。在此，我们将给出一些解决可能出现问题的实际建议。

为了测试组件之间的一致性，在每次一个组件终止时停止过程并且把其它组件当作黑盒子是个好主意。

您可以使用类似下面的方案来进行测试：

- 在 MQSeries Workflow 构建时环境中，把要测试的活动设置为手动启动。推荐此步骤是因为这样可以使整个测试过程更可追踪。照例将已修改的 FDL 导出并将其导入正常运行时环境。关于更多细节，请参考第 4.1.5 节“ 从构建时环境导出模型到正常运行时环境 ”（第 144 页）。

- 停止 UPES 执行。例如：停止相应的 MQSeries Integrator 消息流。
- 使用 MQSeries Workflow 客户机启动工作流。关于更多细节，请参考第 4.1.2 节“ 使用 MQSeries Workflow 客户机确认消息流 ”（第 144 页）。
- 在相应的 UPES 输入队列中检查入站消息的内容。别忘了记下 MQMD 标题中 ReplyToQ 字段的数值，因为以后将用到该信息。您可以用以下几种方法来分析该消息：

  a. 您可以简单地在命令提示符下使用命令 amqsbcg：

  ```
  C:\amqsbcg MQSI.INPUT.VC >testmsg_vc.txt
  ```

  b. 您也可以使用程序 rfhutil。该程序是支持包 IH03 MQSeries Integrator V2 - Message display and test utility 的一部分。您可以从 IBM 网站中下载该支持包：

  ```
  http://www-4.ibm.com/software/ts/mqseries/txppacs
  ```

  在本案例中，您应该特别小心，因为工具 rfhutil 将改变该消息的 MQMD 上下文。我们建议重复测试程序而不是使用工具 rfhutil 直接将消息写回队列。
- 在重新启用 UPES 之前，使用 Windows NT Services 窗口停止 MQSeries Workflow 服务器。该步骤是检查流出消息的关键；否则，MQSeries Workflow 服务器将直接处理该消息。
- 启动 UPES 处理该消息。
- 一旦停止 MQSeries Workflow 服务器，您将马上在 MQSeries Workflow 服务器相应的队列中发现回复消息。最初的消息包含回复到队列参数。您可以检查结果消息看其是否正常。
- 如果消息格式和内容与您所期望的相匹配，启动服务器来完成该活动。

调试

我们可以使用由 MQSeries Workflow 写的系统和错误日志进行调试。系统日志可以使用 MQSeries Workflow 管理工具访问。

1. 通过选择启动（Start） -> 程序（Programs） -> IBM MQSeries Workflow -> MQSeries Workflow 管理实用程序。

2. 以管理员身份登录系统（默认用户 ID 为 ADMIN）。
3. 通过选择选项从主菜单中选择系统日志命令菜单。
4. 再次选择选项从系统日志命令菜单中选择列表（List）。

如果不能发现有用的信息，您可以查看管理工具中的错误日志命令菜单。您可以通过选择选项 e 从主菜单中选择该菜单。

关于管理工具的输出实例，请参考第 194 页实例 5-5。

更多关于系统日志的信息，请参看《MQSeries Workflow 3.3 管理员手册》编号：SH12-6289-04。

您可以通过系统跟踪文件获得关于系统活动的更多详细信息。我们不再回顾本书中的跟踪工具，有关其更多信息，请参看《管理员手册》。

6

使用MQSeries Adapter Offering创建BOD消息

在本章中，我们将描述如何把包含供应商定单信息的消息转换成 BOD 格式。商业对象文档（BOD）是 XML 文档，其中的词汇由开放应用程序组（Open Application Group）进行定义。

因为 MQSeries Adapter Offering 是最近才加入 MQSeries 产品家族的新产品，所以我们将从概述该产品来开始本章。然后，我们使用 MQSeries 适配器生成器创建适配器并将 MQ 消息转换成 BOD 消息，因为这是我们将用来与供应商通信的消息格式。

6.1 MQSeries Adapter Offering简介

6.1.1 概述

MQSeries Adapter Offering (MQAO)是 MQSeries 产品家族的最新成员之一。MQAO 提供了为公司和特定行业应用而构建和定制 MQSeries 适配器的框架和工具。

MQSeries Adapter Offering 由两个核心组件组成：

MQSeries Adapter Offering 的核心组件是：

- MQSeries 适配器生成器 V1.0.1
  这是创建和生成多个适配器的图形工具包。
- MQSeries Adapter Kernel V1.1.1
  这是已生成的 C 和 java 适配器的 MQAO 正常运行时环境。MQSeries Adapter Kernel 为 MQSeries 或 JMS 提供了一套通用 API。

MQSeries Adapter Offering 是为补充 MQSeries 和 MQSeries Integrator 解决方案（从相对简单有明确定义的“ 应用与应用间的集成 ”到更加复杂的包含工作流和消息代理的“ 商业解决方案 ”）而设计的。例如：一个简单案例可能只包含需要在两种不同的格式之间发送和接收消息的两个 C++应用程序。对于这种类型的解决方案，MQAO 可以通过 Adapter Kernel 使用适配器生成器（适配器生成器）和正常运行时来提供必需的适配器。

更复杂的包含 MQSeries Workflow 和 MQSeries Integrator 的解决方案也可以利用 MQSeries Adapter Offering 来实现。例如：MQSeries Workflow 可以启动依次传递消息到 MQSeries Integrator 的业务流程。MQSeries Integrator 应用必要的规则并将该消息路由到 MQAO 正在监听的队列。反过来，MQAO 也可以执行快速的 BOD 转换，然后将该消息传递给组织内的一个业务伙伴/供应商或另一个接收应用程序。

以下部分给出了 MQAO 体系结构的图形演示，并且将帮助您更好地理解 Adapter Offering 是适合于专用业务还是企业应用集成（EAI）解决方案的位置。

6.1.2 体系结构概述

MQSeries Adapter Offering 是点到点解决方案。图 6-1 演示了当组织中的 EAI 环境的复杂性增长和范围扩大时，如“ proprietary ”模型所示，改变基础架构所花费的时间和成本也会同时增加。原因很简单，当越来越多的应用程序绑在一起时，理解消息格式将更加困难，对每一变化编码将花费更多的时间，并且冗余数据的可通用性也会增加。而且，当一个应用程序改变时，所有与该应用程序连接的其它应用程序也需要改变。在经常改变的电子环境中，时间和成本的限制将成为更大的挑战，因此，对一个简单的点到点模型的需求就是不言自明的了。

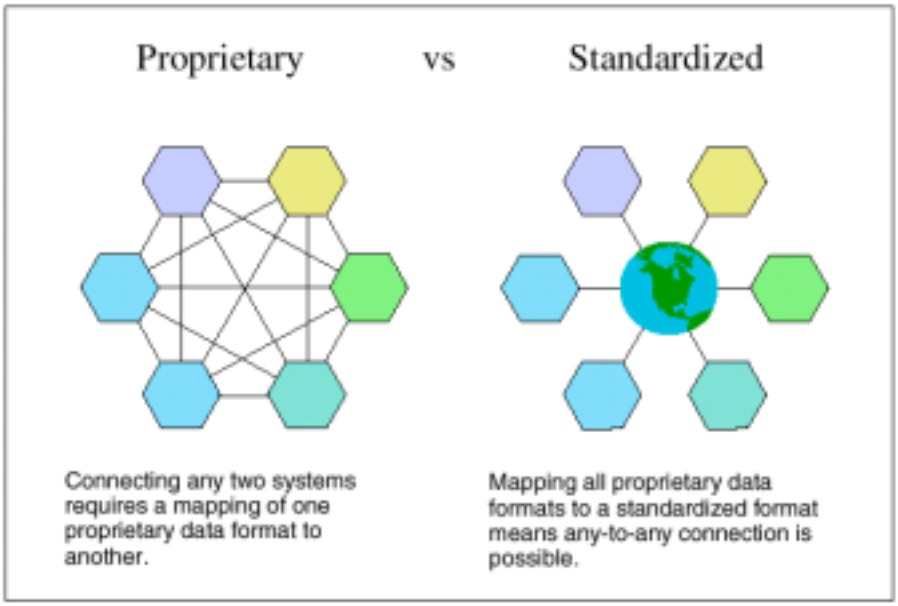

图 6-1 企业应用集成方法

MQSeries Adapter Offering 将通过为构建和应用点到点解决方案提供一个标准化框架来战胜这一（proprietary）挑战。现在，消息可以转换成标准格式，比如 XML、OAG 和 BOD，所有目标应用程序只需解释一种格式。请参考 6.1.6 节“ MQSeries Adapter Offering 业务便利 ”（269 页）以便了解该方法的商业便利。OAG 和 BOD 将在本书第 266 页的 6.1.5 节 “ 开放应用程序组 ”中解释。

图 6-2 演示了一个典型的 MQAO 环境。在模型的左边，我们可以看到源应用程序，该程序使用源适配器将独特的消息格式转换成通用 XML 结构。反过来，该 XML 结构由目标适配器接收，该目标适配器是与目标应用程序连接的。适配器生成器用来开发和应用适配器，而做为正常运行时环境的 Adapter Kernel 使用 MQSeries 或 JMS 将消息路由到相应的适配器。

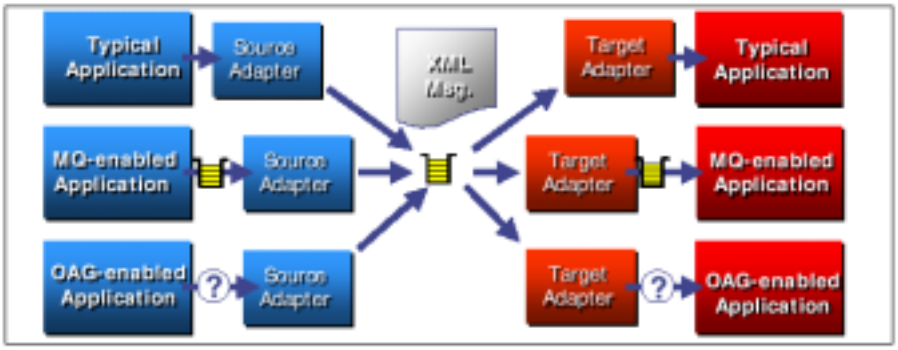

图 6-2MQSeries Adapter Offering 使用实例

如图 6-3 所示，我们可以从该模型中看到源应用程序使用源适配器将消息转换成本地格式（该实例中为 BOD 格式）。然后，Adapter Kernel 将该消息加入 MQSeries 队列，MQSeries Integrator 将把该消息路由到指定队列。在该实例中 MQSeries Integrator 也用来执行记录和跟踪任务。一旦该消息到达指定队列，目标适配器将立即从该队列中获取消息，并将本地格式转换成统一格式，然后将该消息发送给相应的应用程序。这在模型的底部右手边演示。目标适配器将使用 Adapter Kernel 后台程序来侦听指定的 MQseries 队列。

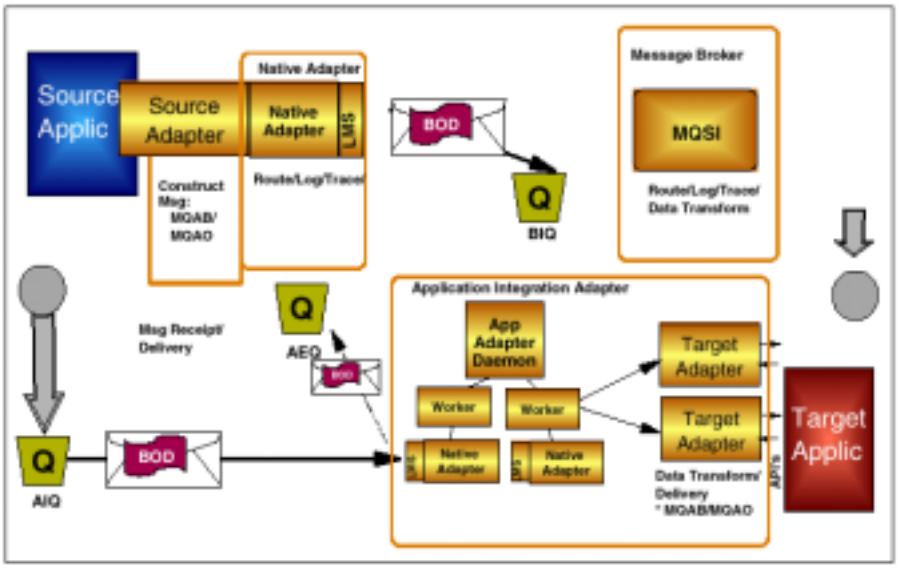

图 6-3MQSeries Adapter Offering 典型应用的详细视图

现在，我们已经知道了 MQSeries Adapter Offering 可以既用于仅包含两个应用程序的简单的点到点解决方案，也可用于拥有更大的 MQSeries Workflow/MQSeries Integrator 环境的合伙企业中。

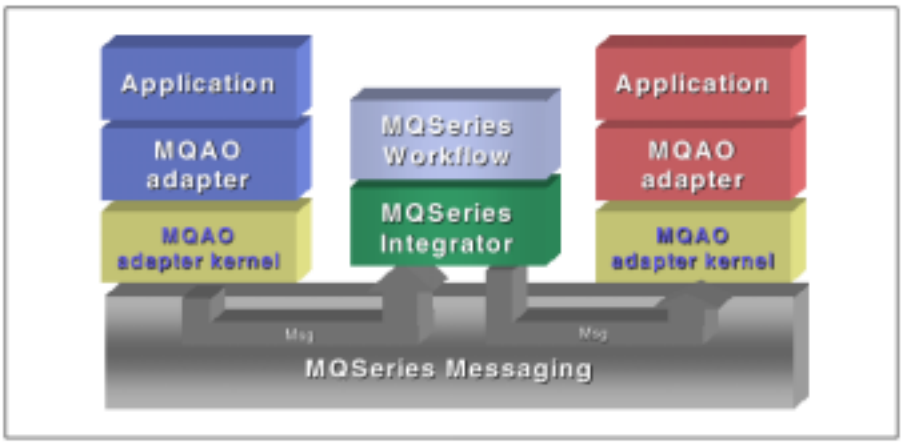

图 *6-4MQSeries Adapter Offering 与 MQSeries Integrator 和 MQSeries Workflow* 的结合

冒着过于简单化的危险，您可能会说在应用程序和队列之间有了 MQSeries Adapter Offering 之后，应用程序就不再需要使用 MQSeries 的 API 了。但是有时对于已封装的应用程序来说，可能没有介绍这些 API 使用方法的选项。至于 MQSeries Integrator，起始点实际上就是队列中的一个消息。在 MQSeries Integrator 中，大多数消息流将从 MQInput 节点开始，然后做多次转换并将消息传送到需要它的地方。

表 6-1 提供了一个简单图表以帮助比较 MQSeries 家族产品之间的不同，并且帮助您更好定位和设计 EAI 解决方案。

表 6-1MQSeries 产品家族比较

	MQ Series	MQ Series Pub/Sub	MQ Series Adapters	MQ Series Integrator	MQ Series Workflow
Point-to-Point Communication	✔		✔		
Intelligent message routing		✔		✔	✔
Syntactic translation			✔	✔	✔
Semantic translation				✔	✔
Extended application interactions					✔

MQSeries Adapter Offering 当前支持五种平台：

- Microsoft Windows NT/ 2000
- IBM AIX V4.3.x
- IBM OS/400 V4R4/5
- HP-UX V11
- Sun Solaris 7 and 8

6.1.3 MQSeries 适配器生成器（适配器生成器）

适配器生成器是图形工具。该工具允许开发者通过处理 C 头文件（包含函数原型和结构定义）或 Java 类的方法来把应用程序接口导入储存库。适配器生成器也允许通过处理文档类型定义文档（DTD）（例如 OAG、BOD）将消息格式定义导入储存库。

这将在图 6-5 中演示。

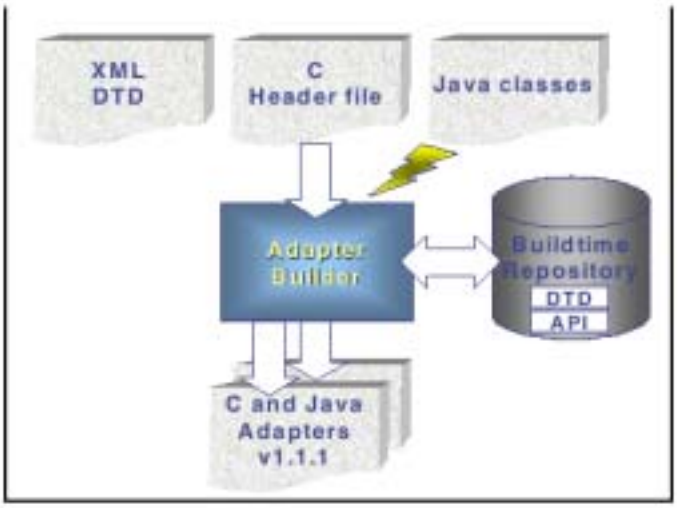

图 6-5 在 MQSeries Adapter Offering 中导入应用程序接口

在导入或定义消息类型之后，适配器生成器将模拟并映射到通用 XML 格式的消息数据转换，反之亦然。MQSeries Adapter Offering 通过使用*微流*（如 277 页图 6-16 所示）来实现该功能。请注意类似 MQSeries Integrator 的图形接口。

提示：MQAO 的微流可能在功能上类似于 MQSI 消息流，但它们之间有一些非常关键的差别。例如：MQAO 的输入和输出终端并不象在 MQSI 中一样代表输入和输出 MQSeries 队列，而是代表单一应用程序的输入和输出终端。您应该始终谨记，您正在创建的每一微流都代表一个单一应用程序的 API。

适配器生成器使用微流及与其相关的数据映射来创建和生成 C 或 Java 适配器（映射成通用 XML 格式）。

6.1.4 MQSeries Adapter Kernel

MQSeries Adapter Kernel 将提供 MQSeries 适配器（使用适配器生成器工具创建）所需的标准正常运行时功能性。这包括与 MQSeries 排队、配置以及处理日志和调试信息的交互。所有适配器的 Adapter Kernel 都是相同的，所以在每种支持平台上所构建的代码也是相同的。

我们可以从图 6-6 中看到，源适配器绑在由适配器生成器创建的特定应用程序中。源适配器将源数据转变为与应用程序无关（application-neutral）的格式，比如 XML BOD。然后，该消息被传递给本地适配器，本地适配器将直接与 MQSeries 通信。

适配器后台程序（daemon）将为每个应用程序输入队列实例一个或更多“ 工人 ” 。每个“ 工人 ”管理一个本地适配器并将消息传送给相应的目标适配器。

Adapter Kernel 正常运行时的主要组件

以下部分将给出关于 Adapter Kernel 正常运行时主要组件的一些高级介绍。关于每一组件的更多信息，请参考《多平台 MQSeries Adapter Kernel:快速入门》编号：GC34-5855。参看图 6-6，它提供了所有组件的图解。

源适配器

源适配器是由适配器生成器构建的生成码，它为特定应用程序提供 API。

消息储存器（Message holder）

Adapter Kernel 把消息储存器用作元数据容器，并用它来压缩引入的消息和其它相关数据，即应用程序标识符、种类和消息类型。

本地适配器

Adapter Kernel 使用本地适配器发送和接收消息储存器对象。本地适配器提供简单数据路由功能，并能够支持多消息传输层，即 MQSeries 和 JMS。

适配器后台程序

Adapter Kernel 使用适配器后台程序例示目标应用程序的适配器“ 工人 ”。一旦将其激活，适配器后台程序将始终保持活跃状态，即使队列中没有消息。适配器后台程序位于 EAI 体系结构的接收方。

适配器“ 工人 ”

适配器“ 工人 ”是将消息发送到相应目标适配器的过程。每个本地适配器有一个“ 工人 ”。适配器“ 工人 ”由适配器后台程序创建并启动，并且可以激活多个“ 工人 ”，因此允许多线程传送消息到预定义目标适配器。

目标适配器

目标适配器由适配器生成器构建，并为目标应用程序提供 API。

配置组件

关于 Kernel 正常运行时环境的配置组件概述，请参考第 265 页“ 配置文件－aqmsetup 和 aqmconfig.xml ”。

如图 6-6 所示，黄色盒子是 Adapter Kernel 的组件和部分产品，红色盒子则代表由适配器生成器生成的代码。

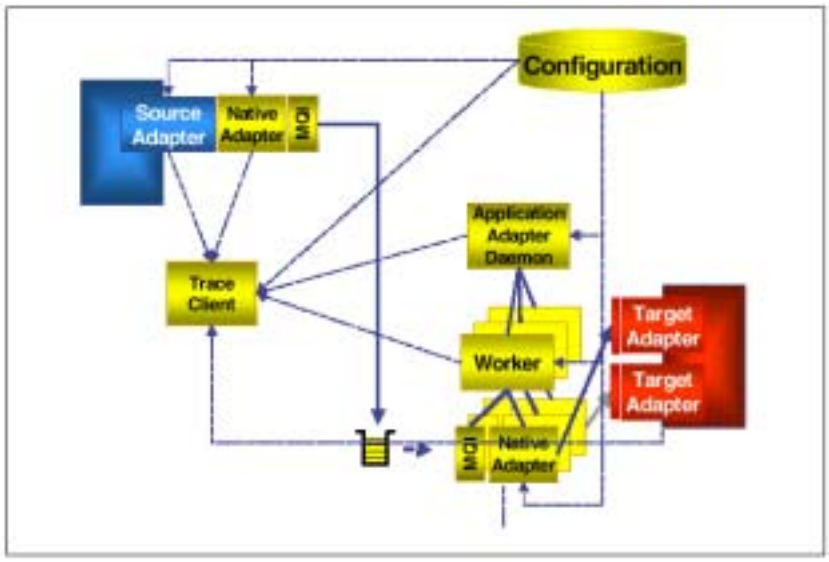

图 6-6　MQSeries Adapter Offering 不同组件详细视图

配置文件－aqmsetup 和 aqmconfig.xml

MQSeries Adapter Kernel 使用以下两个配置文件来存储正常运行时信息。

aqmsetup－定义引导程序值文件。

aqmconfig.xml－用于定义正常运行时参数（对于源适配器和目的适配器有不同的数值）。

提示：“ aqmconfig.xml ”文件中含有多个 Adapter Kernel 在运行期间使用的可配置参数。想要更深入的了解这些参数的功能，请参考《多平台 MQSeries Adapter Kernel 快速入门》。

可以通过查寻系统环境变量“ AQMSETUP ”的数值来找到“ aqmsetup ”文件。然后，您可以在文件“ aqmsetup ”中指定文件“ aqmconfig.xml ”的位置。

虽然 “ aqmconfig.xml ” 文件可能看起来比较复杂，但应注意到，适配器生成器生成了一个模板配置文件，您可以将其嵌入当前配置。

6.1.5 开放应用程序组公司

首先，让我们来澄清一些缩写，它们的某种变化指的是同一实体。OAG（Open Applications Group）代表开放应用程序组。有时提到组织 OAGI （Open Application Group Incorporated），它代表开放应用程序组公司。OAGIS（Open Application Group Integration Specification）称为规范。开放应用程序组（OAG）是一个非赢利团体，主要关注于最好的惯例、基于过程的 XML 电子商务目录和应用程序集成。它的任务是开发高质量的商业目录及其 XML 表示法。

开放应用程序组已经创建了目录级规范。这些规范称为商业对象文档（BOD）。BOD 将指定在商业伙伴之间交换的 XML 文档中包含哪些字段。最高等级 BOD 含有两个字段：控制区和业务数据区，如图 6-7 所示。

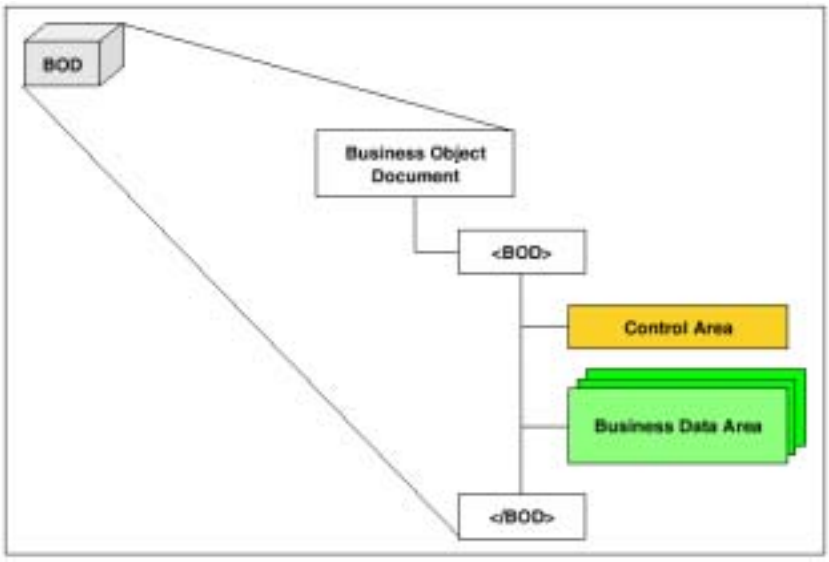

图 6-7 BOD 体系结构

每一 BOD 只包含唯一的一个控制区。如图 6-8（在第 268 页）所示，控制区包含三个主要组件。商业服务请求（Business Service Request<BSR>）、发送者（Sender）和日期（DateTime）三个组件组合成为全球唯一标识符（Global Unique Identifier （GUID））。GUID 将用于安全、事务处理、报告、异常处理、确认和重发。

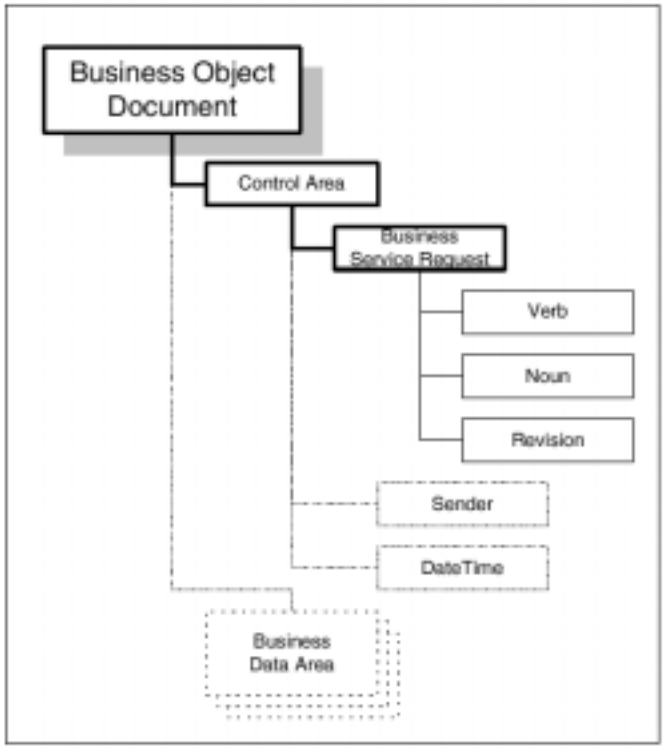

图6-8 BOD：控制区组件

BSR 是发送器应用程序需要接收器应用程序执行的行为。BSR 由动词（verb）、名词（noun）和修订本（revision）构成。动词是 BSR 的行为关键字。一些可能的动词为：

- Get
- Create
- Sync
- Add
- process

名词指出必须执行动词的对象。一些可能的名词为：

- RFQ (Request for Quote)
- Prodorder
- BOM (Bill of Material)
- PO (Purchase Order)

修订本用来标识 BSR 版本。每一 BOD 都有自已的版本号，它是以 001 开始的三位数字。

结合这三个元素，您可以得到下面 BOD 的 BSR：

```
add PO 006
```

商业对象文档（BOD）的商业数据区域（BDA）包含支持商业服务请求（BSR）所需的所有代码、参数和数值。例如：将购买定单（Purchase Order）或定单（Orders）发送给业务伙伴，商业数据区域就会包含所有购买项目（以行表示）的标题和行信息。

276 页的图 6-15 在适配器生成器中导入 DTD 之后显示了 BOD 的某些字段。

6.1.6 MQSeries Adapter Offering商业优势

MQSeries Adapter Offering 作为消息层主要基于 MQSeries 这一事实意味着它将提供 MQSeries 所提供的所有优势。例如：有保证的传输、控制和发送消息到超过 35 个不同平台应用程序的能力。

其它主要优势包括：

- MQAO 解决方案体系结构中的所有应用程序可以使用单一通用接口，比如 XML OAG BODs。
- MQSeries Adapter Offering 可以将消息从单一源应用程序路由到多个目标应用程序，虽然这种路由是静态的。
- 典型地，一个应用程序的改变只影响一个接口，从而保持了通用接口的完整性。
- 缩短了开发和应用时间。

6.2 创建MQAO源适配器

回忆我们在 BuyXYZ.com 解决方案中遇到的问题，MQSeries Integrator 正在生成一条包含定单信息的消息。然而，供应商希望我们使用 XML BOD 消息。我们将使用 MQSeries Adapter 来转换消息，因为 MQSeries Adapter Offering 中含有可以导入 OAG DTD 文件（用以描述 BOD）的工具。

图 6-9 显示了 MQSeries Adapter Offering 在整个解决方案中的位置。

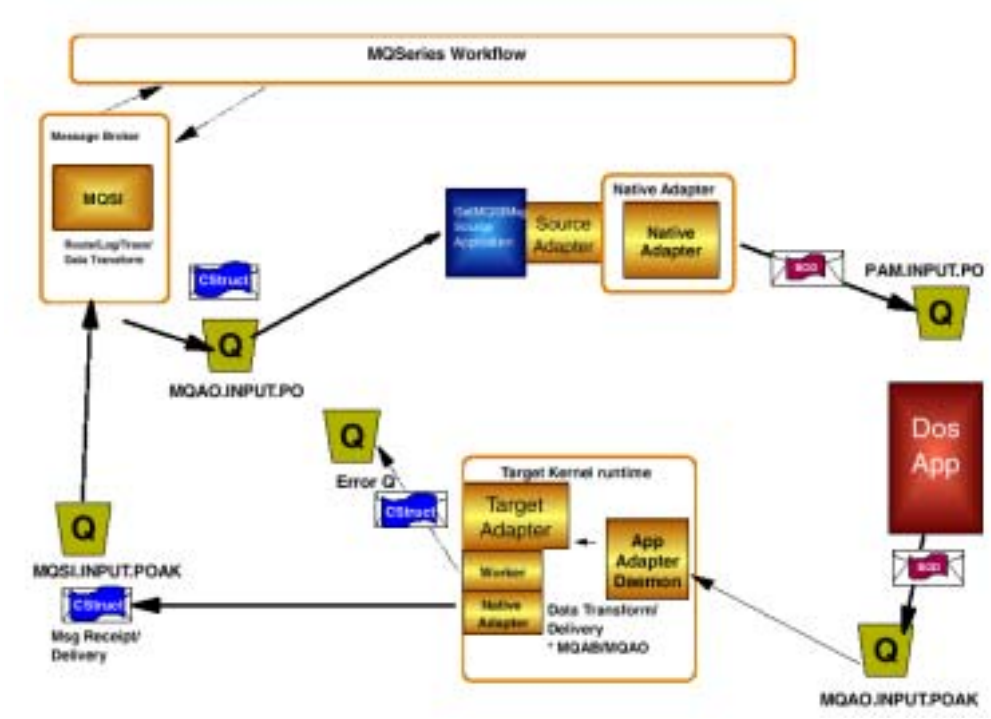

图6-9 在BuyXYZ.com环境中使用MQSeries Adapter Offering

我们所采用的方法并不是对此问题唯一的解决方案。它是人们理解 MQSeries Adapter Offering 概念的基本方法。我们开发了一个 Java 小程序来从消息流“BuyXYZ_Supply_Order_PO”使用的队列中读取订单消息。然后，该 Java 程序将调用我们将在本章中创建并生成的适配器。该适配器负责将数据格式重新定义为 XML BOD 并将其写入目标队列。在该方法背后隐含着这样一种理念——它将引出更多我们所期望的使用 MQSeries Adapter Offering 的方法。应用程序 A 正在使用某些数据（在本案例中，已从队列中检索出数据，但这些数据不相关）工作并希望将其发送给应用程序 B。因此，应用程序 A 将调用相应适配器以完成该工作。也就是说，它将把该消息转换成所期望的格式并将其路由到应用程序 B。

6.2.1 在MQSeries Integrator中生成语言绑定

在 MQSeries Adapter Offering 中创建消息集有多个选项。第一个选项由在 MQSeries Integrator 中生成的语言绑定以及在 MQSeries 适配器生成器中导入已生成的标题文件组成。打开 MQSeries Integrator 控制中心并选择标签消息设置（Message Sets）（见图 6-10），在 MQSeries Integrator 中生成语言绑定。

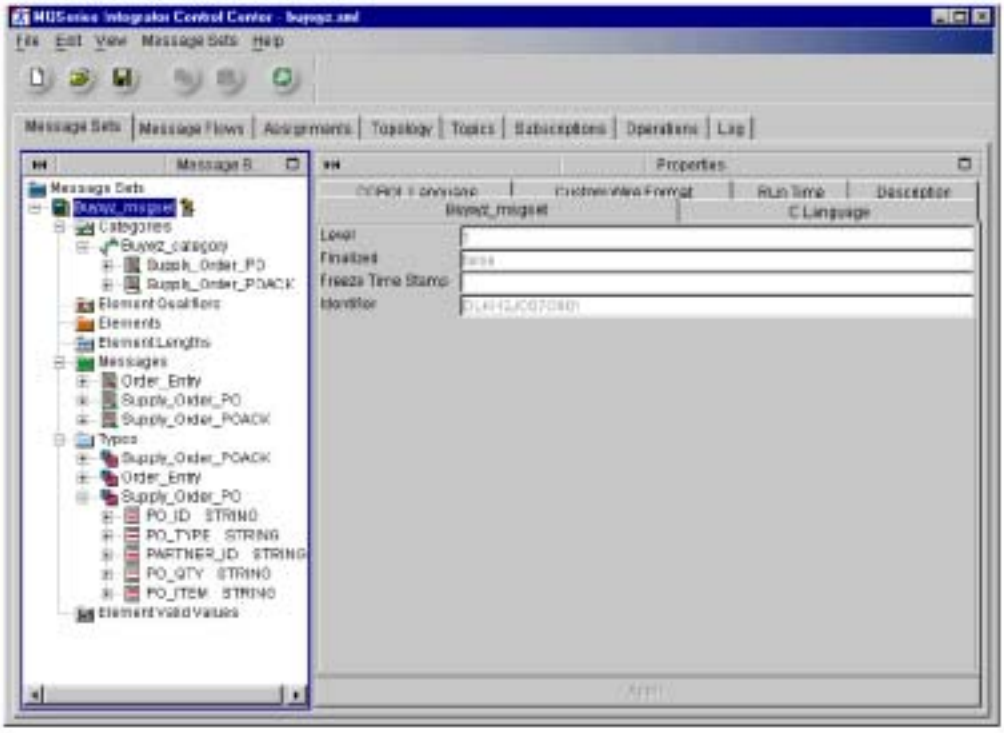

图 6-10MQSeries Integrator 控制中心

因为语言绑定是基于消息类型类生成的，所以您可能需要定义一个类。如果是这种情况，我们还需要在类中添加“ Supply_Order_PO ”和“ Supply_Order_POACK ”消息类型。右键单击该类并选择生成（Generate） -> 语言绑定（Language Bindings） -> C，将出现如图 6-11 所示的窗口。

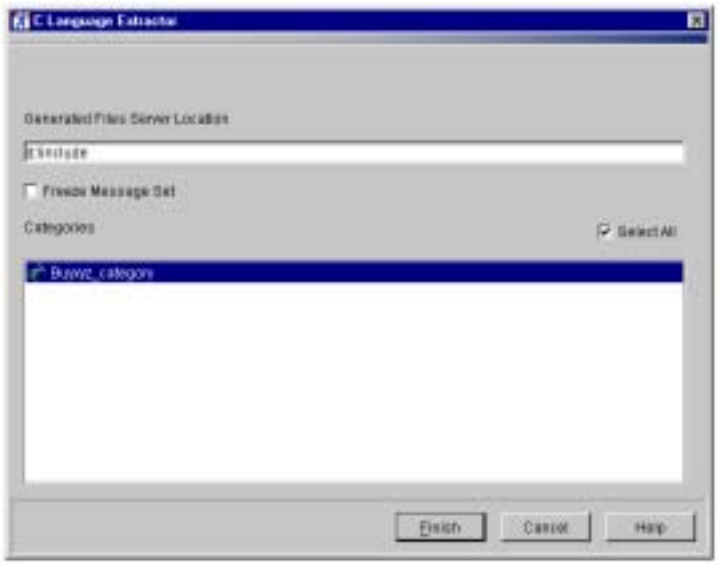

图 6-11 语言绑定生成器

为 C 头文件提供一个目的文件目录并选择完成（Finish）。结果 C 头文件可以直接从 MQSeries 适配器生成器中导出。这样可以保证 MQSeries Adapter Offering 中的消息定义与 MQSeries Integrator 中的定义完全相同。

6.2.2 6.2.2在MQSeries Adapter Offering中创建消息类型

代替使用之前创建的头文件，我们将在 MQSeries Adapter Offering 中使用另一种机制定义消息集。如实例 6-1 所示，一个将从队列中读取消息的 Java 程序摘录。

实例 6-1Java 源文件 mqjbget.java 摘录

```
package po;

import com.ibm.mq.*;
import java.util.Hashtable;
public class mqjbget

{
  private String qManager ="";
```

```
  private MQQueueManager qMgr;

  public java.lang.String PO_ID;
  public java.lang.String PO_TYPE;
  public java.lang.String PO_PARTNER_ID;
  public java.lang.String PO_QTY;
  public java.lang.String PO_ITEM;
  public mqjbget(){
....
```

组成消息（由 MQSeries Integrator 生成）的五个元素构成了该类的五个字段。可以直接将该 Java 源文件导入 MQSeries 适配器生成器。对于有使用 MQSeries Integrator 经验的人来说，可以导入完整的类文件并且该操作将生成所需的消息集，虽然这可能听起来有点不习惯。

要明白这是如何起作用的，启动 MQSeries 适配器生成器并选择消息集标签。右键单击消息设定文件夹并单击创建（Create） ->消息集（Message Set），您将看到如图 6-12 所示的窗口。

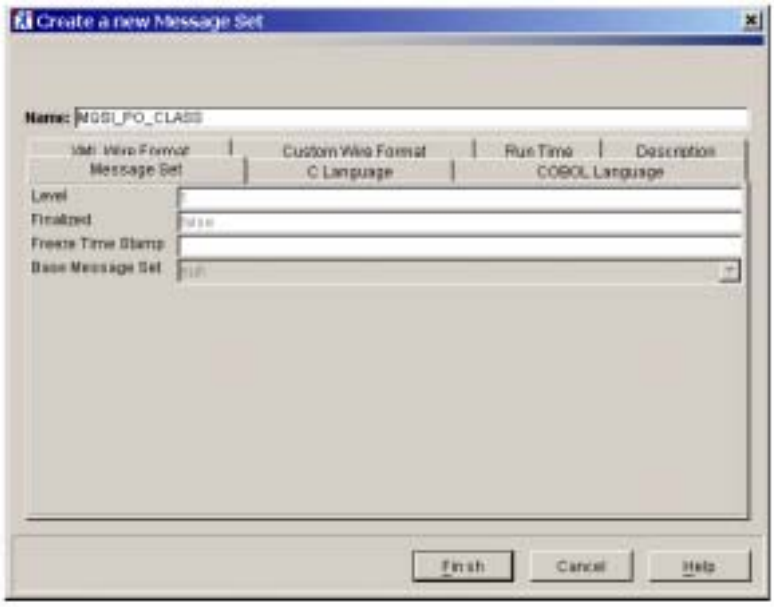

图 6-12 创建消息集

给出一个适当的名称并单击完成（Finish）。

右键单击最新创建的消息集，并单击输入消息集（Import to Message Set） ->Java Bean，将出现一个文件选择窗口。给出类文件存储的目录路径及其名称并单击 Next，将出现如图 6-13 所示的窗口。您可以添加另外的目录，MQSeries 适配器生成器将在该目录中寻找导入类文件所引用的 Java 类。因为导入类文件使用标准 MQSeries Java 类，所以我们将在此添加 com.ibm.mq.jar 文件。

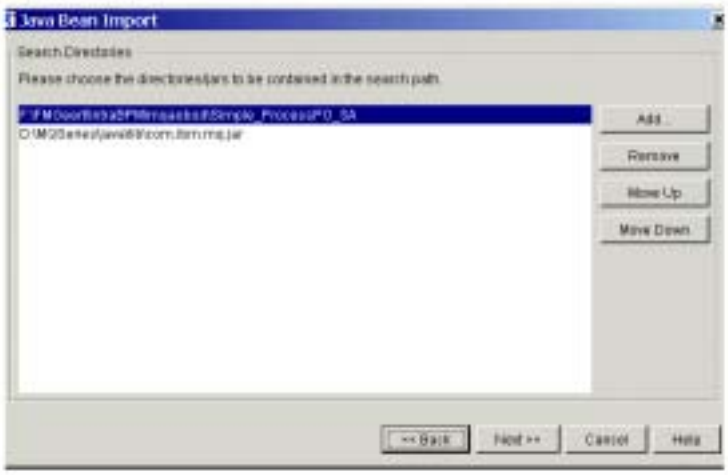

图 6-13Java Bean 导入，给出引用类目录

下一个窗口将允许您从该类文件中导入操作。从 MQSeries Integrator 角度来看，这又是难以理解的。基本上，一种操作仅仅是一个已导入的类方法。这样做的思想是您可以在微流的某点调用这些方法。这次，我们将不使用此概念。但它值得在此提及，因为这是一个非常有用的技巧。单击完成（Finish）完成类文件导入和消息类型创建。

注意到 MQSeries 适配器生成器导入类文件而不只是 Java 源文件也很有趣。这就意味着您可以使用并导入没有源文件的类文件。当创建复杂的微流时，这又是一个非常有用的特征。

与 MQSeries Integrator 相似，一旦导入完成，您就可以在工作间中添加已生成的消息类型了。

您可以导入一个DTD将已导入的消息类型转换成XML BOD。MQSeries Adapter Offering 将提供由 OAG 创建的 DTD。基于需求，您可能需要下载最新版本的 BOD。我们在该解决方案中使用由 MQSeries Adapter Offering 提供的 DTD。

再次右键单击消息集并选择输入消息集（Import to Message Set） -> DTD....，将出现文件选择窗口。在 MQSeries Adapter Kernel 的 runtimefiles\oag 目录中选择文件 003_process_po_005.dtd 并选择下一步（Next），将出现如图 6-14 所示的窗口。您可以在该窗口中指定需要使用 XML 文档中的哪些元素作为一个消息类型。选择元素 PROCESS_PO_005，它是“ process_po ” BOD 中的顶级元素。

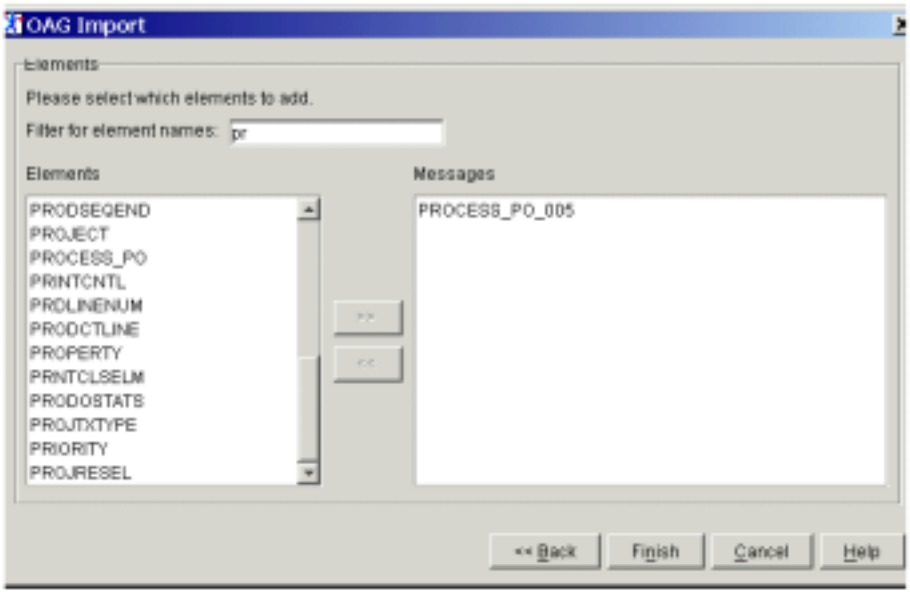

图6-14XML 导入：选择顶级元素

选择 Finish 开始导入过程。一旦导入完成，我们就值得研究一下导入器如何创建该消息类型。有趣的是，我们将看到导入器创建了一些子类型，这些子类型作为元素在 DTD 中并不存在。此技术的一个实例“ 数量元素 ”如图 6-15 所示。

图 6-15DTD 生成消息类型详细视图

现在，我们已经在 MQSeries 适配器生成器中定义了两个消息类型。让我们马上开始创建适配器过程。

6.2.3 创建微流和适配器

创建适配器需要两个步骤。首先，您需要创建微流，并在微流中指定必须执行的操作和转换。这可能会用到其它微流，因为一个微流将被当作创建更加复杂的微流和适配器的构件。然后，使用顶级微流来创建适配器。

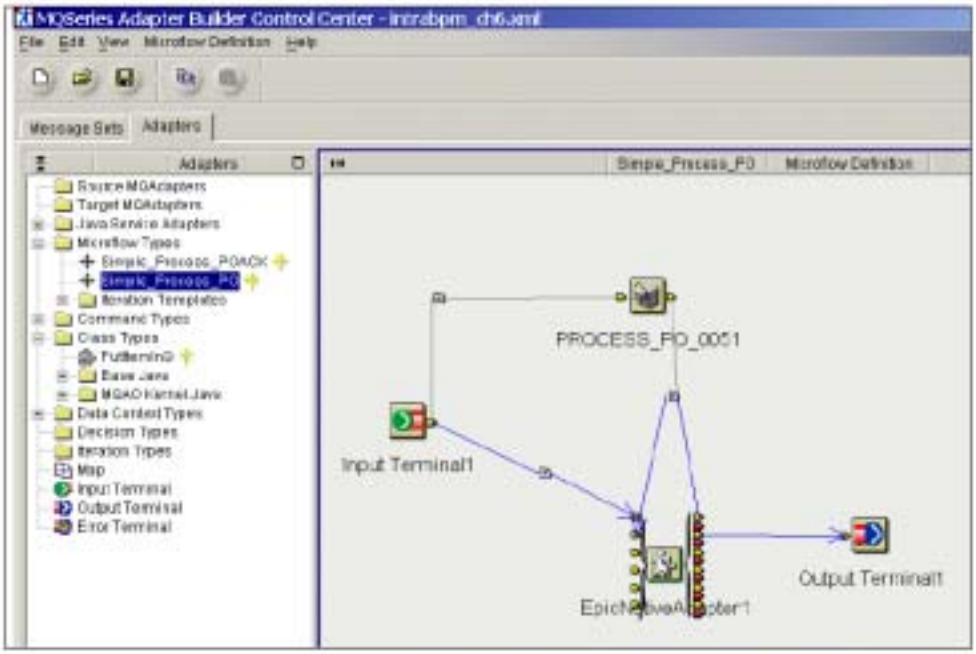

图6-16 微流概览

图 6-16 显示了我们将在此构建的微流。在解释所有细节之前，让我们先看一下它的体系结构。从概念来说，您可以将适配器以及顶级微流当作程序设计语言中的一个函数或面向对象设计语言中的一种方法。它含有多个参数用于输入和输出，并且它可以有返回值。函数或方法的输入参数在 MQSeries Adapter Offering 中映射输入终端，返回值概念则映射输出终端。进一步扩展此映射，数据上下文类型精确映射了函数或方法定义中的本地声明变量。稍后，我们将在本节中讨论数据上下文类型。

我们认为无论您在什么时候创建微流，记住这些相似性是非常重要的。输入终端和输出终端看起来很象 MQSeries Integrator 中的 MQInput 和 MQOutput 节点，但是它们在功能上却完全不同。MQInput 节点通过从队列中读取消息获取其输入；而输入终端则从调用适配器返回的参数中获得其输入。MQOutput 节点将处理后的消息写入 MQSeries Integrator 的队列中，而输出终端只将一个返回值传回适配器的调用者，后者必须将已处理的数据写入另一位置的队列中。因此，不要在输出终端属性中寻找队列名字段。

如图 6-16 所示，创建微流的第一步是创建数据上下文类型。右键单击文件夹 Data Context Types 并选择 Create-> Data Context Type。正如先前解释的一样，数据上下文类型很象程序设计语言中的变量声明。并且，正象您为变量提供一个类型一样，也需要指定数据上下文类型的类型。数据上下文类型将用来保存已转换的数据：XML 文档。从而，消息类型将设置为 PROCESS_PO_type。为数据上下文类型提供适当的名称并选择 Finish。

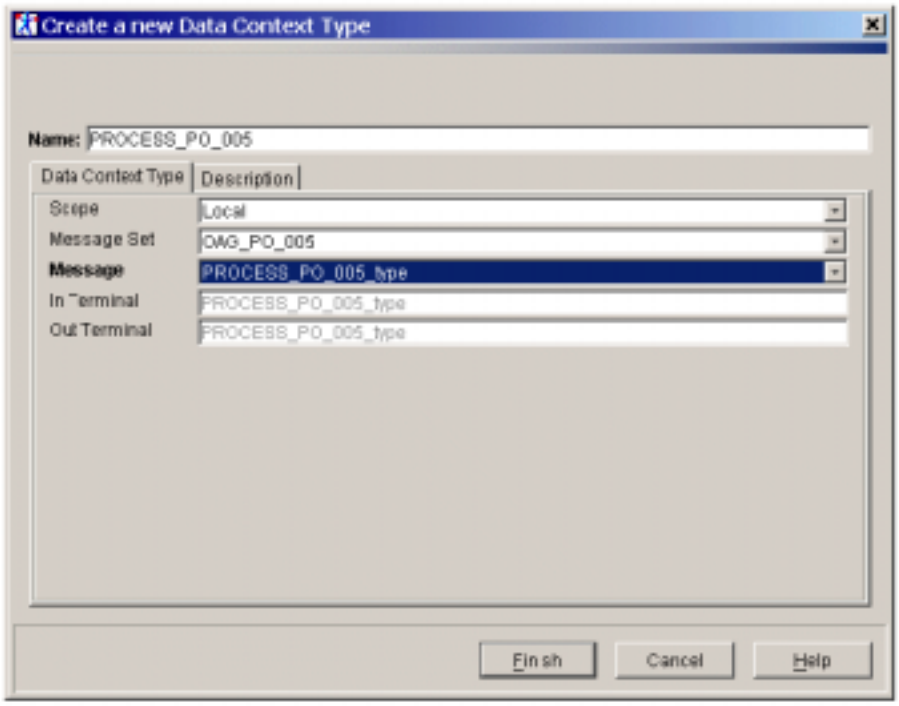

图 6-17 创建新的数据上下文类型

下一步，我们将创建微流。右键单击微流类型（Microflow Types）文件夹并选择创建（Create）。给出适当的名称并选择完成（Finish）。

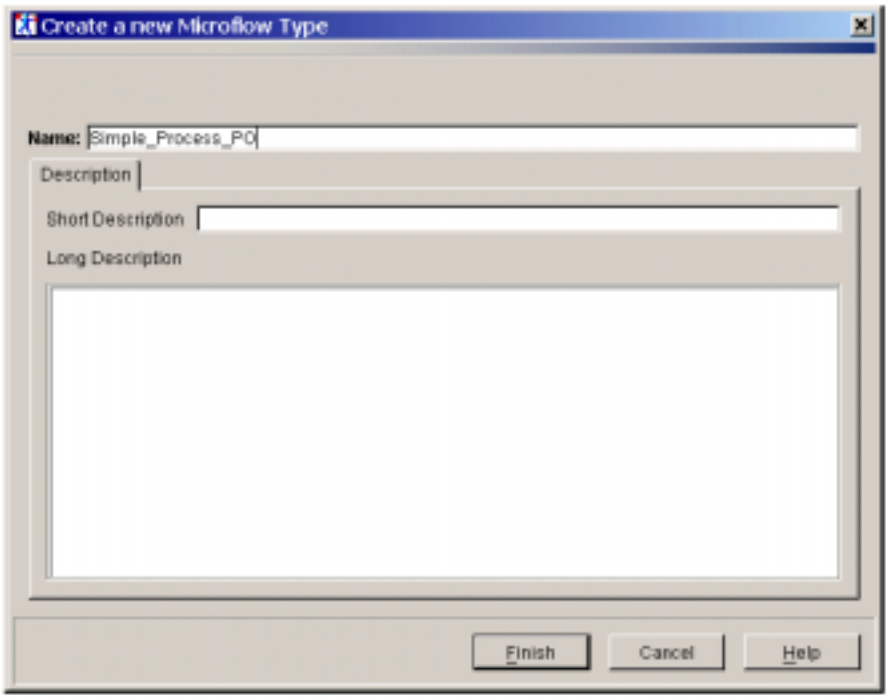

图 6-18 创建新的微流类型

一旦创建了微流类型，我们就可以开始在控制中心右窗格中添加组件并在其中定义微流类型了。将一个输入终端和一个输出终端拖放到右窗格中。在微流定义中添加数据上下文类型，然后添加 EpicNativeAdapter，它是 MQSeries Adapter Kernel 的组件。从程序设计角度来看，它也是在适配器中所使用或调用的函数或方法的接口。

右键单击输入终端（Input Terminal）并选择属性（Properties），然后指定正确的消息集和消息类型。

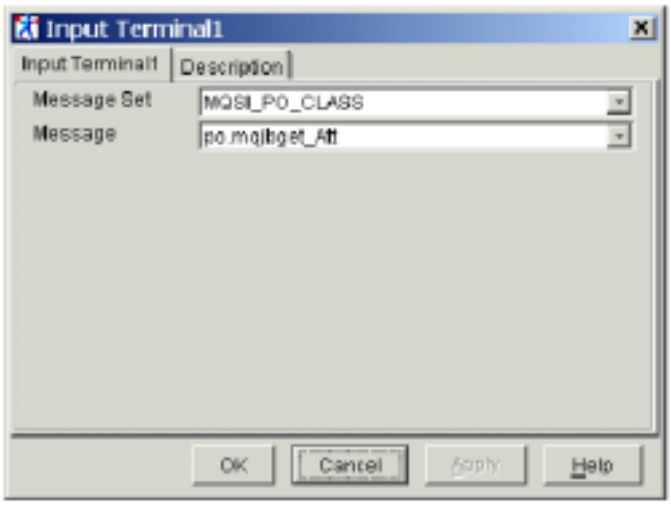

图6-19 输入终端属性

右键单击输出终端（Output Terminal）并选择属性（Properties）。在此您可以再次设置消息集和消息类型。象先前提及的一样，这在概念上是您传回适配器调用者的返回值。对于这一简单的适配器，我们将不使用此特性，而保留消息类型为空。

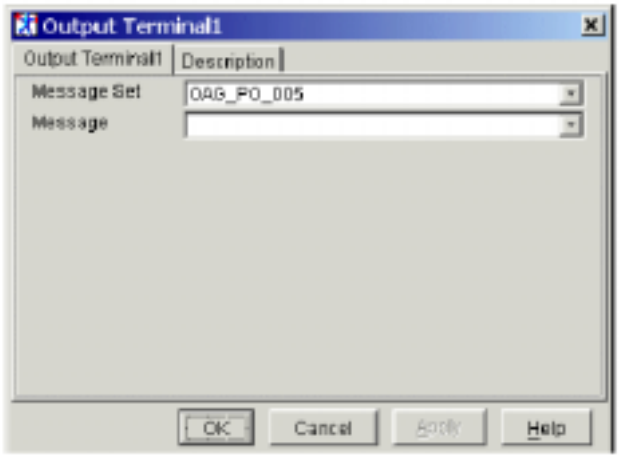

图6-20 输出终端属性

现在，让我们把它们连接起来。右键单击输入终端（Input Terminal）并选择连接（Connect）-> out。移动鼠标将其指向 EpicNativeAdapter 左边的顶级终端并单击它。您可以将这一连接器称为 EpicNativeAdapter 构造器。

再次右键单击输入终端（Input Terminal）并选择连接（Connect） -> out。移动鼠标将其指向数据上下文左边的终端。选择该终端并单击 In Terminal -> DataConnectionType。该数据连接器将指示 MQSeries Adapter Offering 把数据存储在数据上下文中。该连接器上的小图标被称为“ map ”。通过修改“ map ”的属性，我们将详细描述如何将引入消息类型“ po.mqjbget_Att ”转换成数据上下文类型“ PROCESS_PO_type ”。

单击“ EpicNativeAdapter ”右上端的终端并将其与左边的终端相连，如图 6-16 所示。这将调用本地适配器的“ sendMsg ”方法。在该连接器上，我们将用到第三个“ map ”图标。将数据上下文的输出终端与该“ map ”的输入终端相连并将该连接器设置为“ DataConnectionType ”连接器。最后，构造一个从“ EpicNativeAdapter ”到输出终端的控制连接器。

现在，让我们定义转换本身。双击输入终端和数据上下文之间的“ map ”图标并选择数据映射表达式（DataMappingExpression）标签。

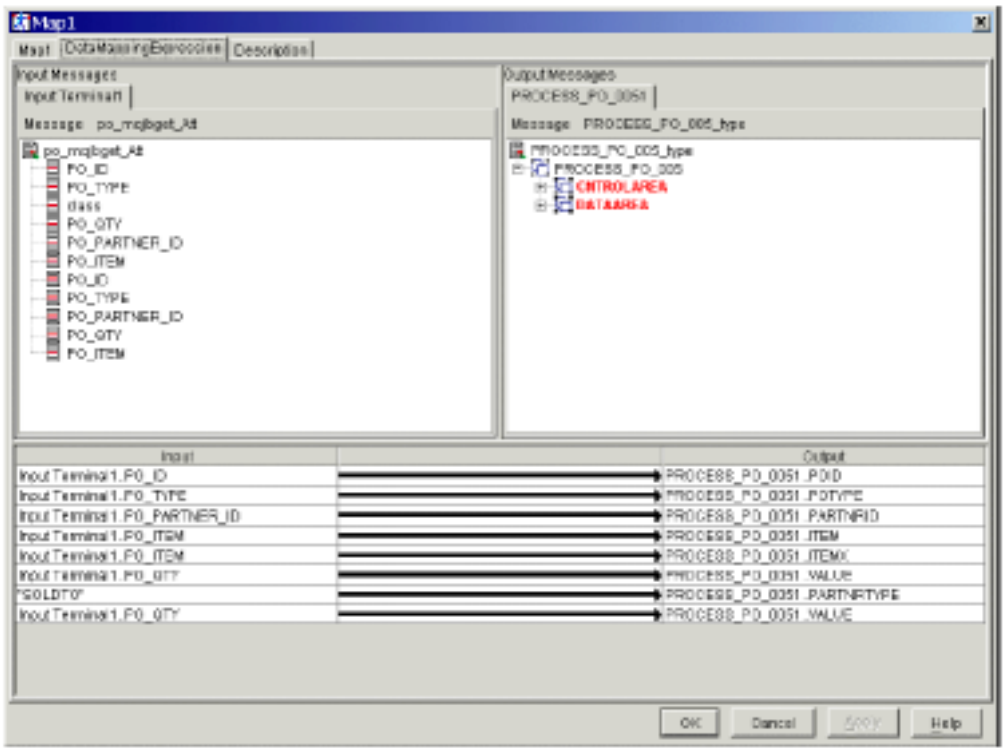

图 6-21 数据映射表达式标签

将该字段从左边窗格拖放到右边窗格以便创建转换。右键单击输入字段和输出字段之间的连接器，将弹出一个菜单。选择 Advanced，将出现一个显示当前状态的小编辑器。您可以使用该编辑器添加自定义码来调整该转换。该自定义码所使用的语言与 MQSeries Integrator 中使用的 ESQL 很相似。

右键单击输入和输出字段本身，也将出现上下文菜单。选择高级的（Advanced）将获得调整该转换的另一种可能性。图 6-22 和图 6-23 显示了相应的窗口。

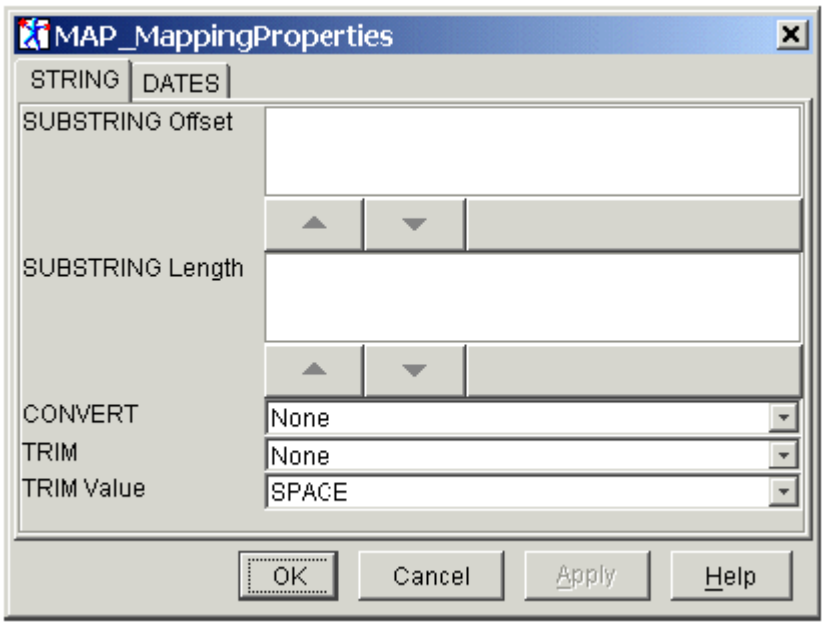

图 6-22 高级转换设置中的输入字段

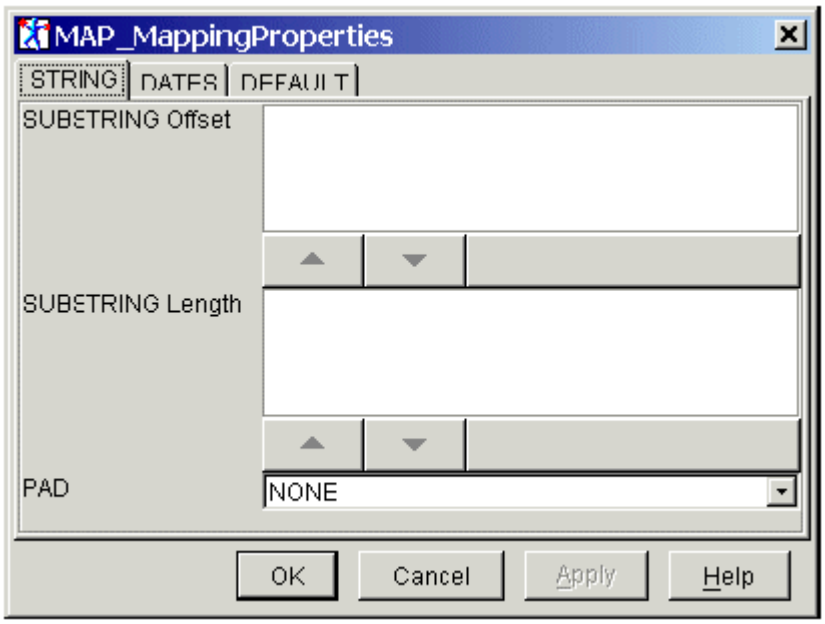

图 6-23 高级转换设置中的输出字段

现在，双击 EpicNativeAdapter 上面的图标并选择数据映射表达式（DataMappingExpression）标签（如图 6-24 所示）。

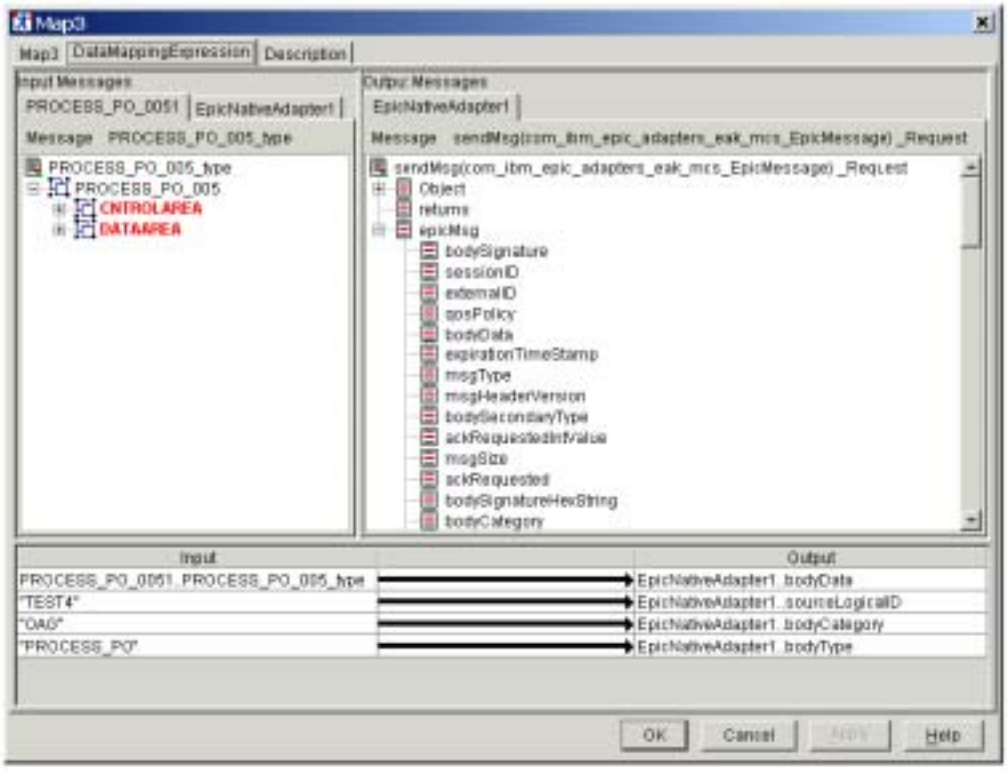

图 6-24 数据映射表达式标签

在该映射中，您将指定如何构成到本地适配器的输入。本地适配器需要知道怎样使用“ sourceLogicalID ”、“ bodyType ”和“ bodyCategory ”。“ bodyData ”将映射为“ process_po ”消息。在此设置的三个文字将是读取 MQSeries Adapter Offering 配置文件的关键。该配置文件将决定消息以及其它事物的目的文件。右键单击输出字段，将输出字段映射为一个文字。然后选择 Add Element 菜单选项，我们将在作为输入字段的“ LITERAL ”和所选的输出字段之间添加一个映射。双击关键字 LITERAL 并将其转变为所需数值。注意，您必须包括字符串的双引号。

下一步将是创建适配器本身。右键单击“ Java Service Adapter ”文件夹并选择创建（Create）-> Java Service Adapter。给出适当的名称并选择正确的微流类型，然后选择完成（Finish）。

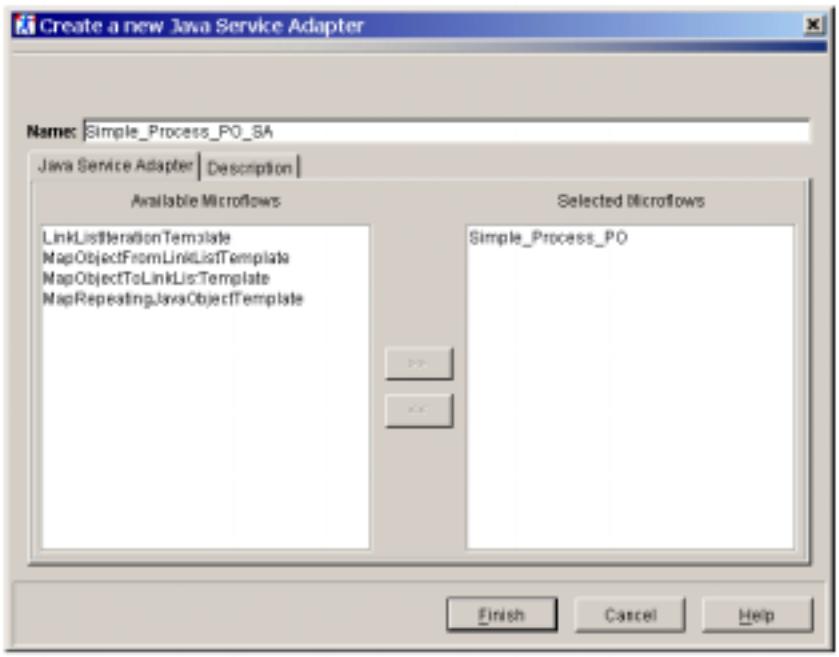

图 6-25 创建 Java 服务适配器

右键单击已创建的适配器并选择生成（Generate） -> Java 中的适配器（Adapter in Java）。将应用程序名设置为“ TEST4 ”并选中生成简单界面（Generate Simple Interface）。

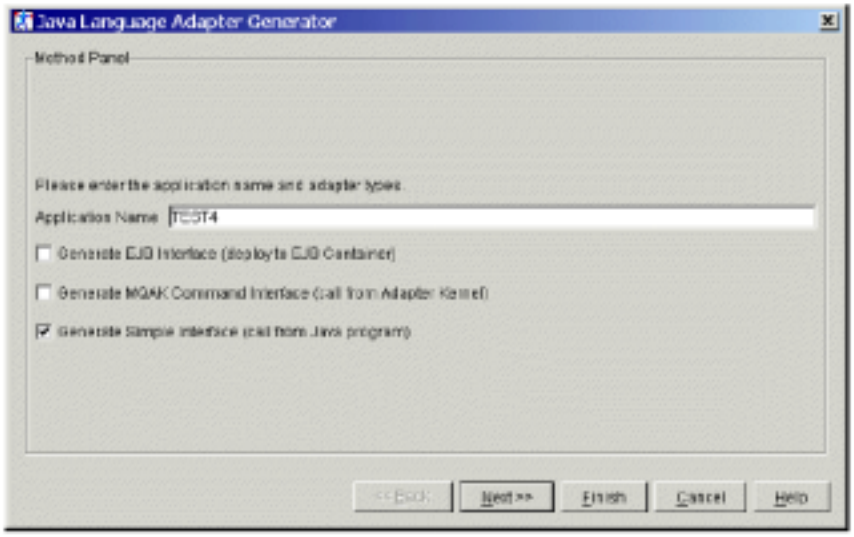

图 6-26 适配器生成器——步骤 1

单击下一步（Next）指定您可能需要附加的导入描述。再次单击下一步（Next）指定生成类将存储的目录以及类名称和包名称。

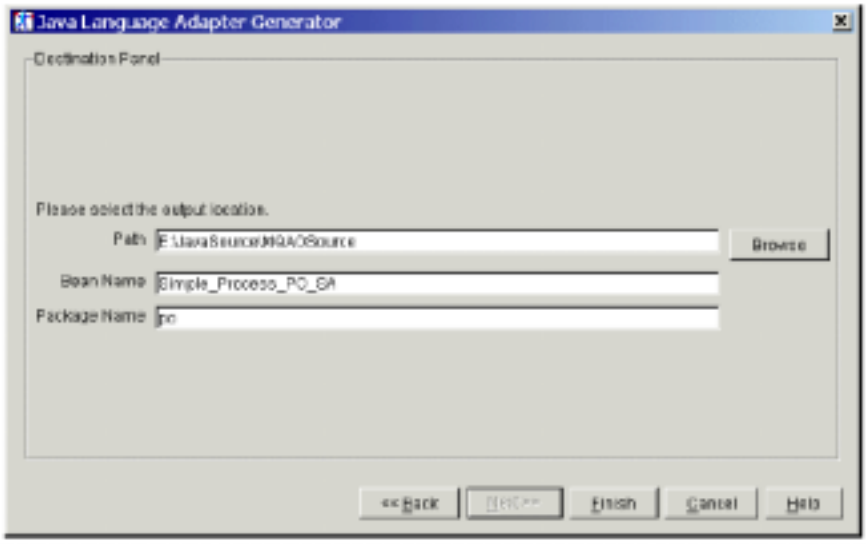

图 6-27 适配器生成器——步骤 2

单击完成（Finish）开始生成过程。

6.2.4 配置适配器

在此，配置适配器需要两个步骤。您将需要更新系统变量“ CLASSPATH ”，使其指向生成类存储的目录。变量“ CLASSPATH ”应包含以下两行：

1. 配置生成阶段源适配器存储目录的名称：

   ```
   E:\JavaSource \MQAOSource \Simple_Process_PO_SA
   ```

2. MQSeries Adapter Kernel 安装的 bin 目录：

   ```
   D:\MQAO \MQAK \bin
   ```

“ CLASSPATH ”变量包括支持 MQSeries Java 所需的条目。其最小值应该是下面行：

```
D:\MQSeries \Java \lib;D:\MQSeries \Java \lib \com.ibm.mq.jar
```

如果您喜欢使用 JMS 接口，还需增加另外一些条目。欲知 JMS 和 MQSeries 所需条目的更多信息，请参考 MQSeries manual 《使用 Java》编号：SC34-5657。

第二个步骤是配置 Adapter Kernel 本身。MQSeries Adapter Kernel 配置可以是基于文件的，也可以存储在 LDAP 中。对于该适配器，我们使用基于文件的配置。顶级配置文件使用环境变量“ AQMSETUP ”来配置：

```
SET AQMSETUP=D:\MQAO \MQAK \SAMPLES \AQMSETUP
```

在该文件中，最重要的设置为：

```
AQMConfig=D:\MQAO \MQAK/samples
```

这意味着 MQSeries Adapter Kernel 将在由参数 AQMConfig 配置的目录中寻找称为“ aqmconfig.xml ”的文件。为了您系统上运行的所有适配器配置文件，您可使用该单一文件。对于该适配器，您将需要如实例 6-2 所示的条目。

实例 6-2 提取 aqmconfig.xml

```
<ePICApplication epicappid="TEST4">
  <!--Tracing on/off.If no entry defaults to false.-->
  <epictrace>true</epictrace>
  <!--Trace levels -512=TYPE_ERROR_EXC (Exceptions),-1=TYPE_ALL (All possible
messages).-->
  <epictracelevel>512</epictracelevel>
  <AdapterRouting cn="epicadapterrouting">
    <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
    <ePICBodyCategory epicbodycategory="DEFAULT">
      <ePICBodyType epicbodytype="DEFAULT">
            <!--Default destinations to send messages to.-->
```

```
          <epicdestids>TEST5</epicdestids>
        </ePICBodyType>
      </ePICBodyCategory>
    </AdapterRouting>
  </ePICApplication>

<ePICApplication epicappid="TEST5">
  <epictrace>true</epictrace>
  <epictracelevel>0</epictracelevel>
  <AdapterRouting cn="epicadapterrouting">
    <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
    <ePICBodyCategory epicbodycategory="DEFAULT">
      <ePICBodyType epicbodytype="DEFAULT">
      <epicreceivemode>MQBD</epicreceivemode>
      <!--Recieve Time out in milliseconds ie.1000 =1 second,-->
      <!---1 means never ending.No entry defaults to 0.-->
      <!--milliseconds.Used when receiving messages.-->
      <epicreceivetimeout>15000</epicreceivetimeout>
      <epicreceivemqppqueue>PAM.INPUT.PO</epicreceivemqppqueue>
      <epicerrormqppqueue>PAM.ERROR</epicerrormqppqueue>
      <epicreplymqppqueue>PAM.INPUT.PO</epicreplymqppqueue>
    </ePICBodyType>
  </ePICBodyCategory>
```

您应该还记得，我们曾在 MQSeries 适配器生成器中使用“ TEST4 ”作为应用程序 ID。kernel 将在配置文件中使用“ TEST4 ”作为关键字。来自应用程序“ TEST4 ”的消息将以应用程序 ID“ TEST5 ”作为默认目的程序。“ TEST5 ”条目表明必须将该消息写入队列“ PAM.INPUT.PO ”，并且消息主体应是 MQBD，这意味着消息中将不含有 MQSeries Adapter 标题信息。由于该消息将以“ 发送－忘记 ”模式（非请求－回复模式）发送，所以回复队列条目与超时在此并不真正相关。

6.2.5 使用适配器

现在，我们有了一个从队列中读取消息的类和多个由 MQSeries 适配器生成器生成的类。将这些类结合在一起，我们将得到一个简单的驱动程序。其中最重要的几行代码如图 6-3 所示。

实例 6-3　Java 主程序代码摘录

```
mqjbget obj =new mqjbget();

//read message from queue to set the fields of the object obj ... (从队列阅读消息以便设置对象obj字段)
```

```
Simple_Process_PO_SASimple adapter =new Simple_Process_PO_SASimple();

adapter.executeSimple_Process_PO(obj);
```

创建类一个“ mqjbget ”对象“ obj ”和一个类适配器对象“ Simple_Process_PO_SASimple ”。一旦将对象“ obj ”的字段填满，您就可以通过读取 MQSeries Integrator 消息来调用适配器上的执行方法并将对象“ obj ”作为一个参数传递。

6.3 创建目标适配器MQAO

第二个适配器将负责转换包含供应商应答的回复消息。此时，一个基于命令行的程序将模拟与供应商的通信。然而，最终目标应是使用例如“ WebSphere 业务伙伴协议管理器 ”这样的程序来处理该通信。或者以另一种方式，在某点上 XML BOD 将到达队列和转换的消息，该消息是 MQSeries Integrator 所期望的 C 结构类型的消息。

该 XML BOD 消息是 “ POACK ”消息类型。

有多个可能方法来解决该问题。我们将选择一种简单的方法。在该方法中，MQSeries Adapter Kernel 后台程序将用于监控包含供应商回复消息的队列。该后台程序使用 MQSeries Adapter Offering 工具将该消息转换成 C 结构格式并调用适配器。然后，适配器将该消息写入用于 MQSeries Integrator 的队列。

该方法背后的思想又与源适配器的思想相同。在典型环境中，后台程序侦听队列并调用一条引入消息所需的适配器。然后，该适配器将做某些特殊应用，例如：将数据插入数据库或调用一个封装应用程序的多个子程序。此案例中所说的特殊应用行为仅仅是将消息写入队列。

6.3.1 创建消息类型

此外，还需创建两个消息类型。对于引入的消息，我们将需要导入描述“ POACK ”消息的 DTD。该 DTD 在 MQSeries Adapter Offering 的安装目录 runtimefiles\oag 中，文件名是“ 004_acknowledge_po_005.dtd ”。以 6.2.2 节“ 在 MQSeries Adapter Offering 中创建消息类型 ”中所描述的相同的方法导入该 DTD。您可以导入现存消息集中的 DTD，或者也可以创建一个单独的 DTD。我们已使用另外一个称为“ OAG_POACK_005 ”的消息集。使用元素“ ACKNOWLEDGE_PO_005 ”作为消息类型的顶级元素。当导入成功

时，您可以看到类似于实例 6-28 所示的消息类型树。

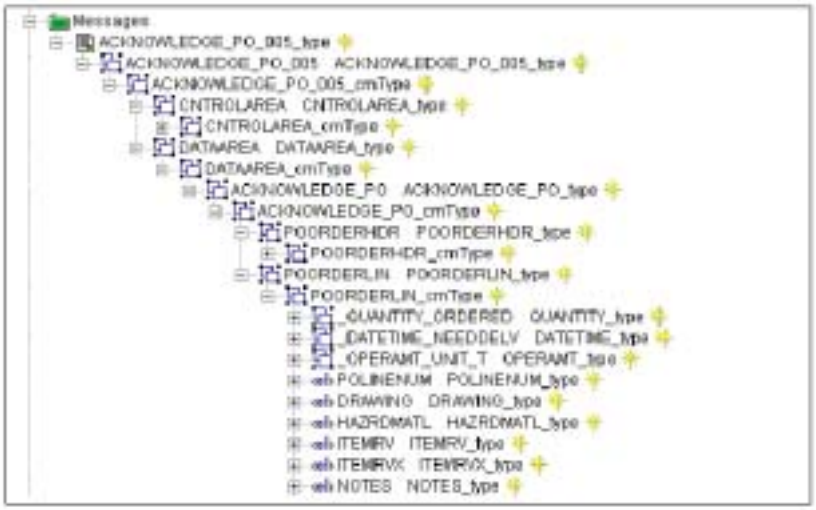

图 6-28 已导入 DTD 的消息类型树

对于目标消息，我们仍然有多个选项来创建消息类型。第一种方法是使用由 MQSeries Integrator 生成的语言绑定来创建消息类型，如 6.2.1 节“在 MQSeries Integrator 中生成语言绑定”（271 页）所述。第二种方法是导入一个 Java 类，将该 Java 类看作一个封装应用程序的子程序。该 Java 类将描述由封装应用程序所提供的方法。实例 6-4 包含了该源文件的一些相关代码行。

实例 6-4 mqbjput.java 的代码摘录

```
package poack;

import com.ibm.mq.*;

public class mqjbput {
  public java.lang.String PO_ID;
  public java.lang.String PO_TYPE;
  public java.lang.String PARTNER_ID;
  public java.lang.String PO_QTY;
  public java.lang.String PO_ITEM;
  public java.lang.String PO_ITEM_PRICE;
```

```
  public java.lang.String PO_ACKCODE;

public void mqjbput_print(){
//.....
```

该类定义含有一些字段和方法 mqjbput_print()。该方法正如其形式所示：它将消息加入队列并在标准输出设备中显示。

如 6.2.2 节“ 在 MQSeries Adapter offering 中创建消息类型 ”（272 页）所述，现在导入“ mqbjput.java ”文件来创建消息类型。象先前我们创建消息集“ MQSI_POACK_CLASS ”一样，您需要将 “ com.ibm.mq.jar ” 文件加入搜索目录窗口。

这时，在“ Operations ”步骤中（见图 6-29），要确保您所选择的方法就是您想在适配器中调用的方法。

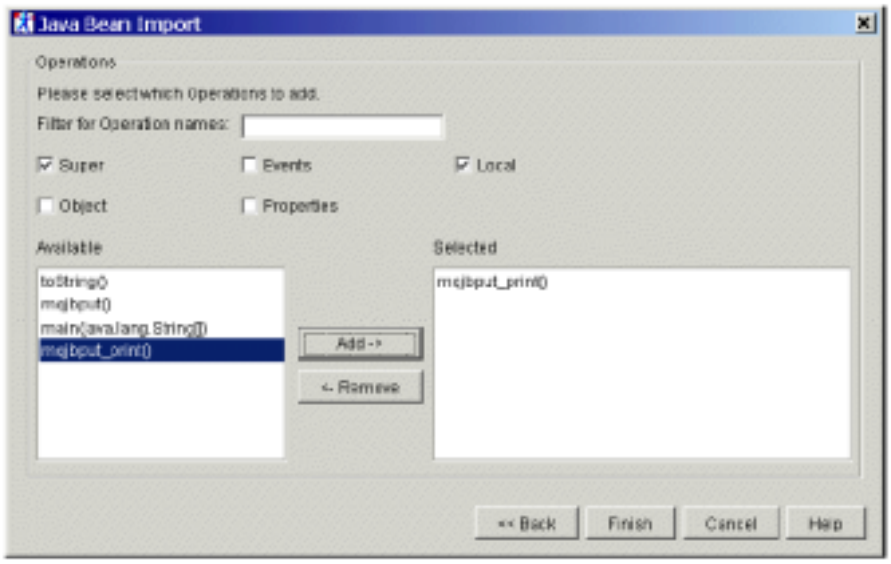

*图*6-29 *选择导入操作*

结果，MQSeries Adapter Offering 将在该消息集中创建一个事务处理类型。为了能够看到此结果，右键单击该消息集并选择添加工作台（Add to Workspace） -> 事务处理（Transaction）选择事务处理程序。图 6-30 显示了在该消息集中所添加的事务处理程序。

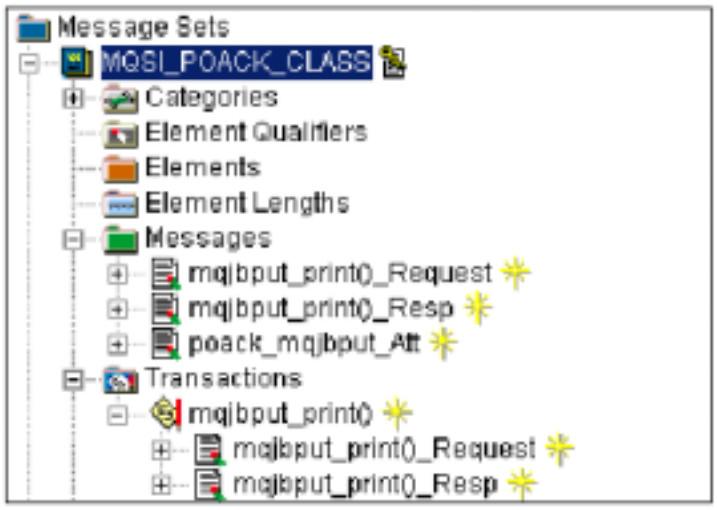

图6-30 导入的事务处理类型

6.3.2 创建微流和适配器

创建目标适配器同样需要两步过程。首先创建微流，然后使用该微流来创建适配器。

右键单击文件夹微流类型（Microflow Types）并选择创建（Create）->微流类型（Microflow Type）。将微流命名为“ Simple_Process_POACK ”并选择完成（Finish）。

将一个输入终端和一个输出终端拖放到工作间。右键单击输入终端（Input Terminal）并选择属性属性（Properties）。以适当数值设置消息集和消息类型。

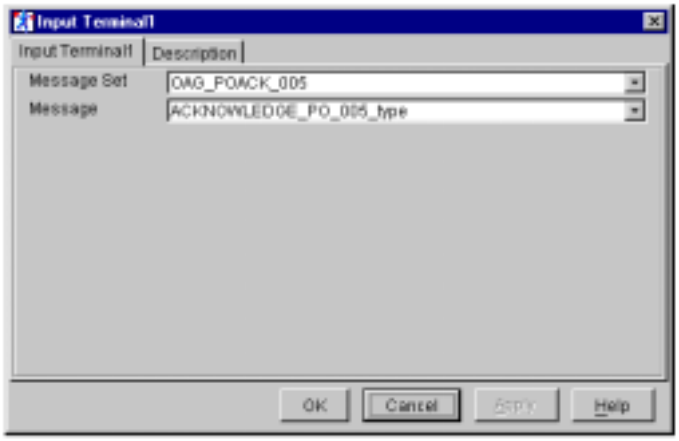

图6-31 输入终端属性

对于源适配器，我们将在工作间中放置一个“ EpicNativeAdapter ”。在此，我们希望使用已导入的方法或事务处理程序。这样做，我们需要创建指向该导入事务处理过程的类字段。右键单击文件夹类字段（Class Types）并选择创建（Create） ->类字段（Class Type）。

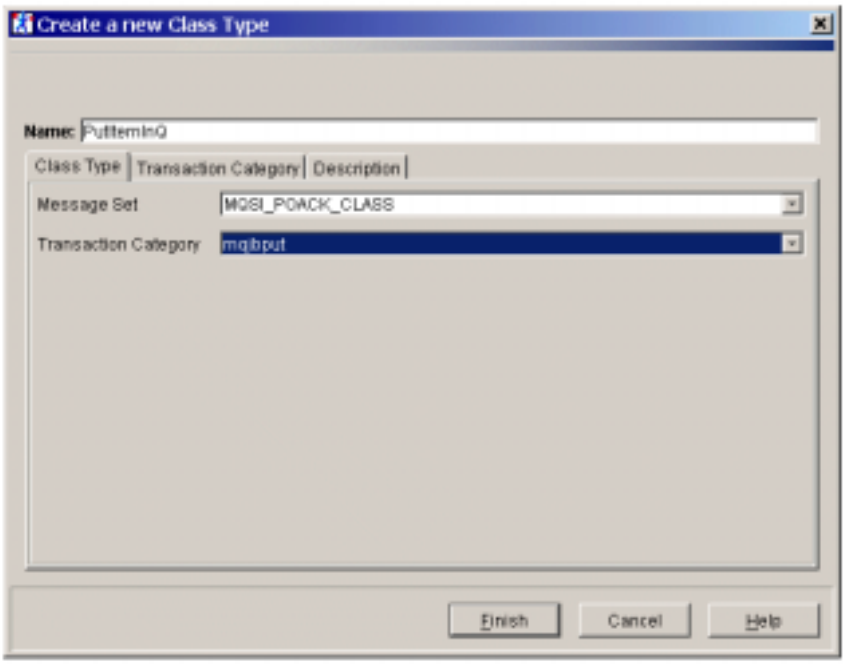

图 6-32 创建新的类字段——步骤 1

给出适当名称并选择“ MQSI_POACK_CLASS ”作为消息集，“ mqbjput ”作为事务处理种类。然后选择事务处理种类（Transaction Category）页。

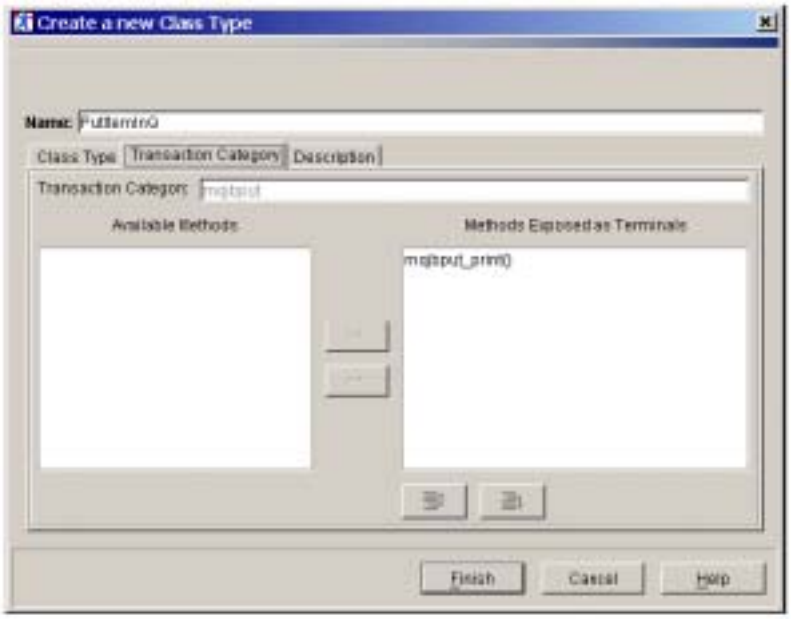

图 6-33 创建新的类字段－步骤 2

选择“ Available Methods ”窗格中的 mqbjput_print()方法并将其移至 “ Methods Exposed as Terminals ”窗格中。选择完成（Finish）完成该类字段的创建。

将该类字段拖放到工作间并在节点之间加入控制连接器。作为结果的微流应如图 6-34 所示。

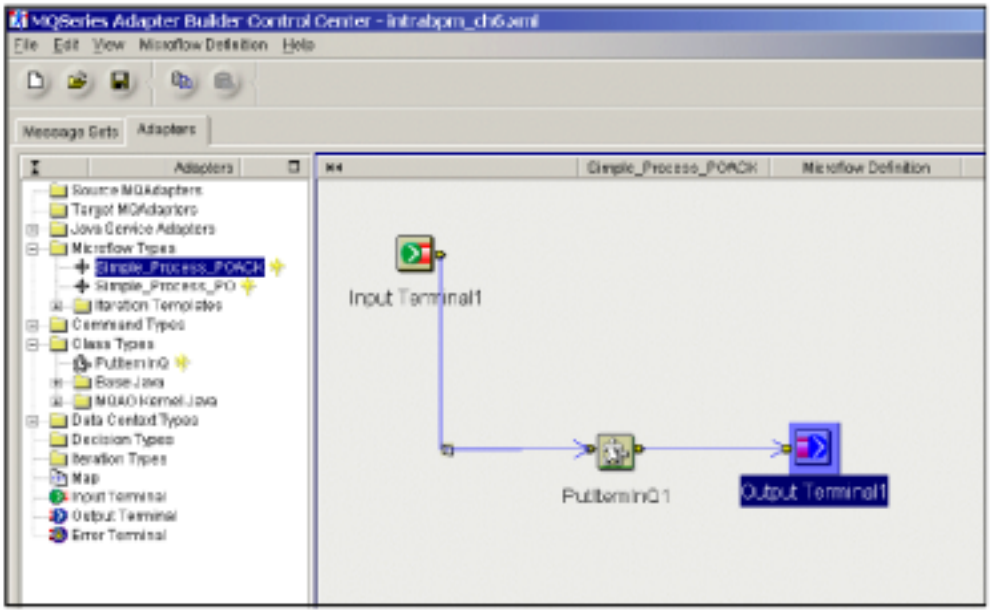

图 6-34 目标适配器的已完成微流

最后，双击“ map ”图标并选择数据映射表达式（DataMappingExpression）标签确认 XML BOD 消息与目标消息之间的转换。

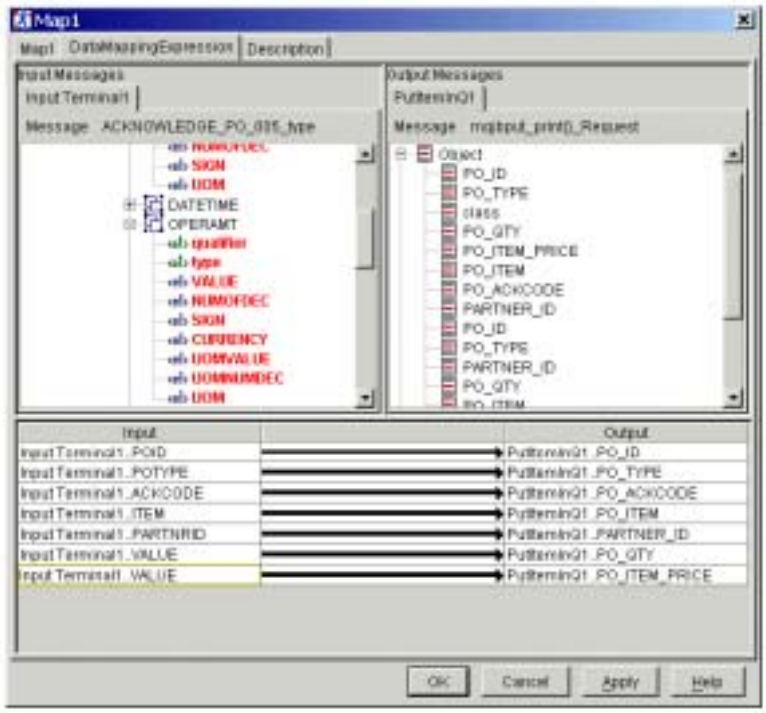

图6-35 数据映射表达式标签

最后，我们需要创建适配器本身。右键单击文件夹 Java 服务适配器（Java Service Adapter）并选择创建（Create） -> Java 服务适配器（Java Service Adapter）。给出适当名称并选择相应微流类型。

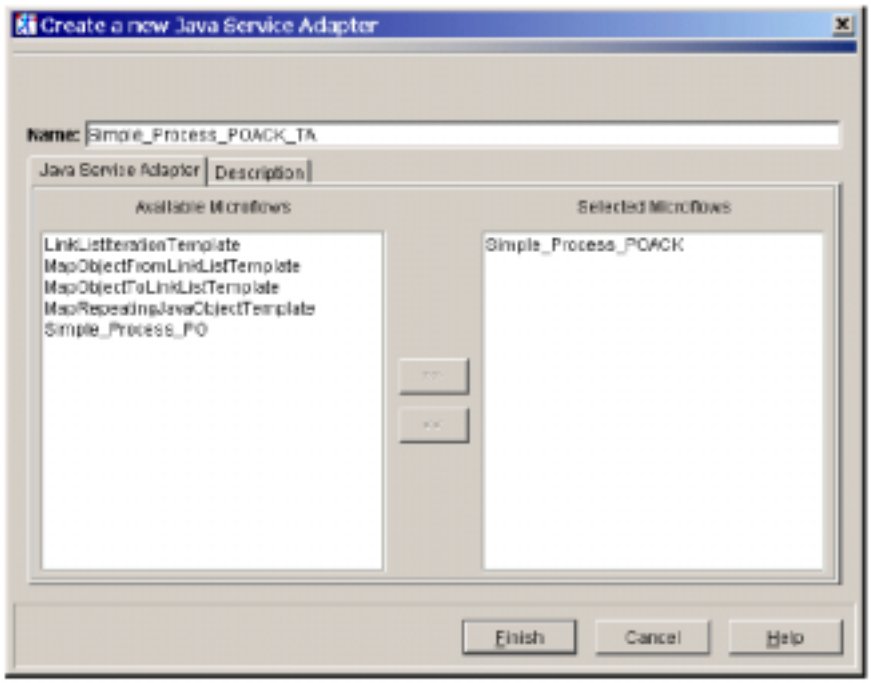

图 6-36 创建新的 Java 服务适配器

右键单击已创建的适配器并选择生成（Generate）-> Adapter in Java。

将应用程序名设置为“ TEST5 ”并选中 Generate MQAK 命令界面，因为我们希望从后台程序“ kernel ”中调用该适配器，单击下一步（Next）。

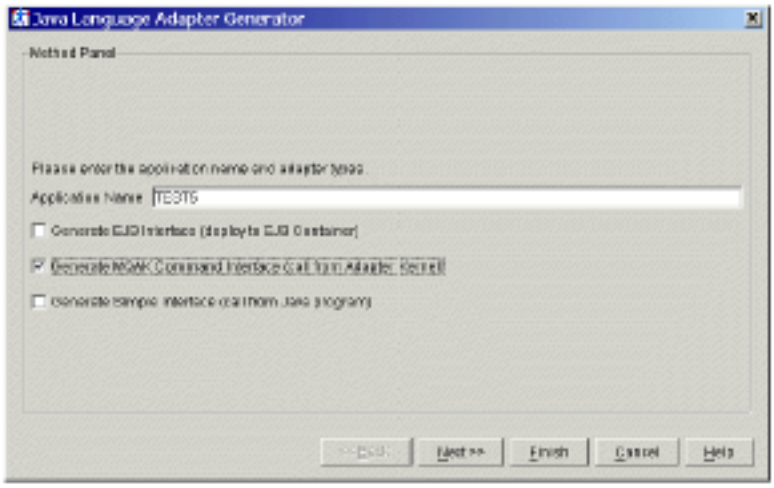

图 6-37 适配器生成器——步骤 1

指定您可能需要的任意附加导入描述并单击下一步（Next）。在此指定该生成类将存储的目录名称以及类名称和包名称。

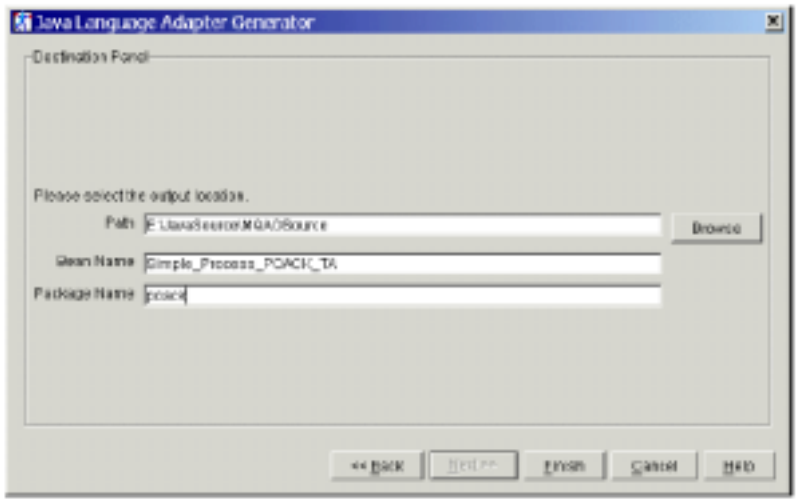

图 6-38　适配器成生器——步骤 2

单击完成（Finish））开始生成过程。

6.3.3 配置适配器

与 6.2.4 节“ 配置适配器 ”（287 页）所述相似，我们需要更新 “ CLASSPATH ” 系统变量以使其包含适配器存储的目录。“ CLASSPATH ”应包含如下命令行：

```
E:\JavaSource \MQAOSource \Simple_Process_POACK_TA
```

也需要包含 MQSeries Adapter Kernel bin 目录和“ com.ibm.mq.jar ”。

适配器后台程序将与源适配器共享配置文件“ aqmconfig ”。实例 6-5 首先显示了“ DEFAULT ”主体类和主体类型条目，这些条目也用于源适配器。接着，您将看到主体类“ OAG ”和主体类型“ ACKNOWLEDGE_PO ”条目，这些条目由适配器后台程序使用。该后台程序将侦听队列“ MQAO.INPUT.POACK ”。为匹配主体类“ OAG ”和主体类型“ ACKNOWLEDGE_PO ”，该后台程序将调用命令类“ poack.simple_process_poack.Simple_Process_POACKInput_Terminal1Command ”。

实例 6-5 aqmconfig.xml 摘录

```
<ePICApplication epicappid="TEST5">
  <epictrace>true</epictrace>
  <epictracelevel>0</epictracelevel>
  <AdapterRouting cn="epicadapterrouting">
    <epicmqppqueuemgr>DEFAULT</epicmqppqueuemgr>
    <ePICBodyCategory epicbodycategory="DEFAULT">
      <ePICBodyType epicbodytype="DEFAULT">
        <epicreceivemode>MQBD</epicreceivemode>
        <!--Recieve Time out in milliseconds ie.1000 =1 second,-->
        <!---1 means never ending.No entry defaults to 0.-->
        <!--milliseconds.Used when receiving messages.-->
        <epicreceivetimeout>15000</epicreceivetimeout>
        <epicreceivemqppqueue>PAM.INPUT.PO</epicreceivemqppqueue>
        <epicerrormqppqueue>PAM.ERROR</epicerrormqppqueue>
        <epicreplymqppqueue>PAM.INPUT.PO</epicreplymqppqueue>
      </ePICBodyType>
    </ePICBodyCategory>
    <ePICBodyCategory epicbodycategory="OAG">
      <ePICBodyType epicbodytype="ACKNOWLEDGE_PO">
        <epicreceivetimeout>15000</epicreceivetimeout>
        <epicreceivemqppqueue>MQAO.INPUT.POACK</epicreceivemqppqueue>
        <!--AdapterDaemon -Command to invoke.-->
        <!--Add the command class name from the Sampleconfig.xml generated by the
builder -->
```

```
<epiccommandclassname>poack.simple_process_poack.Simple_Process_POACKInput_Terminal1Comman
d</ep
iccommandclassname>
        <!--Uncomment this line to Example 5 of the common techniques -->

<!--piccommandclassname>po.ct_array.CT_ArrayInput_Terminal1Command</epiccommandclassname>-
->
        <epiccommandtype>MQAKEAB</epiccommandtype>
      </ePICBodyType>
    </ePICBodyCategory>
  </AdapterRouting>
</ePICApplication>
```

6.3.4 使用适配器

我们现在有了适配器和由该适配器调用对引入消息进行特殊应用处理的 Java 类“ mqbjput ”。方法 mqbjput_print()将在标准输出设备中显示该消息并将其加入相应的队列。

使用以下命令启动后台程序：

```
aqmstrad -a TEST5 -bc OAG -bt ACKNOWLEDGE_PO -noretry
```

参数-a 命令该后台程序使用“ TEST5 ”作为配置文件的关键字。参数-bc 和-bt 用来指定主体类和主体类型。这些参数足以使后台程序找到相应的队列并从中读取和找到引入消息所调用的正确命令。

7

在工作流中调用Enterprise JavaBean

在本章中，我们将看一下在 WebSphere 中调用 Enterprise JavaBean 作为一个活动执行的方法。从 MQSeries Workflow 角度来看，这与在 MQSeries Integrator 中作为消息流执行的活动没有什么不同。

我们将简要讨论在 WebSphere 环境中应用 Enterprise JavaBean。然而，要完全理解应用的每一步骤，请参考本书第 407 页“ 相关出版物 ”以获取更多信息。

7.1 运送EJB

“ ShipOrder ”活动将存储运送细节并通过调用使用 container-managed persistence (CMP) 的 Enterprise Java Bean 来减少产品库存。UPES 调用将是同步的。

我们将从下载称为 *WA03 MQSeries Workflow - API 实例*的 MQSeries Workflow 支持包开始。您可以从下面网站下载该支持包：

http://www-4.ibm.com/software/ts/mqseries/txppacs/wa03.html

我们将基于实例“ IncrementEJBUpes ”来设计解决方案，而该实例是上面支持包的一部分。

运送 UPES 的开发由以下步骤组成：

- 构建和测试 EJB。
- 在 WebSphere 中应用 EJB。
- 在 Java 中写一个 JMS 侦听器类。
- 配置环境并注册 UPES。
- 测试运送应用程序。

7.2 开发和应用EJB

在实例应用程序中，我们将使用非常简单的 EJB 来更新 “ CUSTOMER ”数据库。本部分将概述 EJB 的开发和应用过程。

讨论如何设计和执行 Enterprise JavaBeans 并不是本书的目的。关于该主题的更多信息，请参考《设计和完成服务件, JSPs, and EJBs for IBM WebSphere 应用服务器》编号：SG24-5754。

7.2.1 在VisualAge for Java中应用EJB

我们将在实例应用程序中使用两个 EJB。一个 EJB 用于把运送信息存储到 CUSTOMER.CUSTOMER_ORDER 表中。另一个 EJB 的目的是更新产品表并减少产品库存。

在本部分中，我们将略述一个使用 VisualAge for Java Enterprise 版本 3.5 开发的简单 EJB 实例。我们将使用带有容器管理持久性的 EJB。我们有现存的数据库表，所以将基于现有的信息来创建 EJB。

1. 启动 VisualAge for Java，并确定 EJB 开发环境和 IBM WebSphere 测试环境已正确安装。
2. 确定类路径中已经添加了 DB2 JDBC 驱动器。在工作台中选择 Window -> Options。在“ Options ”窗口中选择 Resources 和 Edit...按钮。如该驱动器不存在，则在类路径中添加 “ db2java.zip ” 文件。
3. 在工作台中选择 EJB 标签。从 EJB 菜单中打开数据库图表浏览器。选择 Open To -> Database Schemas，将出现如图 7-1 所示的窗口。

图 7-1　VisualAge 数据库模式浏览器

4. 导入数据库模式。单击 Schemas - > Import / Export Schema -> Import Schema from Database。
5. 输入模式名，例如：“ SHIPPING ”，将出现数据库连接信息窗口，如图 7-2 所示。输入必要的参数。

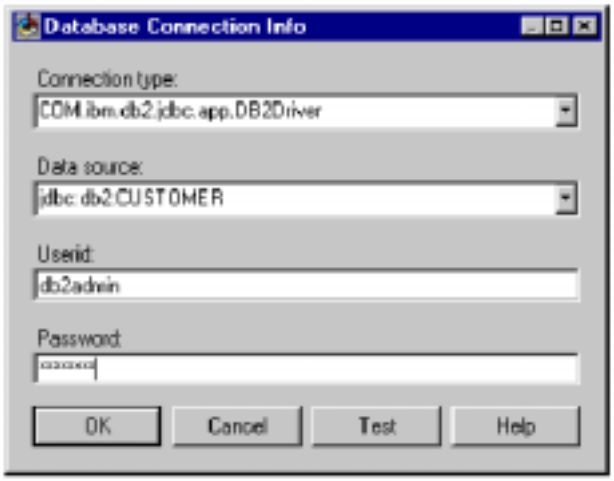

图 7-2　VisualAge 数据库连接信息

6. 在构建数据库连接之后，将出现选择表窗口。选择数据库模式限定词“CUSTOMER”并单击创建表格（Build Table）列表按钮。选择您希望用作 EJB 模式的表。
7. 如图 7-3 所示，所选的表将出现在模式浏览器中。

图 7-3　带有模式的 VisualAge 模式浏览器

8. 保存最新创建的模式。单击 Schemas -> Save Schema...，将出现保存模式智能向导窗口（图 7-4）。确定项目名和包名，例如“BUYXYZ”和“com.buyxyz.ejb”。

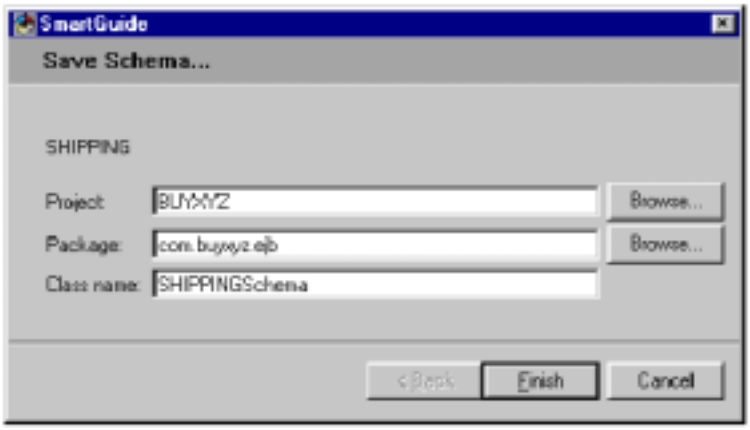

图 7-4 VisualAge 保存模式智能向导

9. 关闭模式浏览器。

现在有了模式，我们便可以基于该模式来创建 EJB 了。

1. 选择 EJB -> Add -> EJB Group from Schema or Model,将出现“ Create EJB Group from Schema or Model ”智能向导窗口（图 7-5）。

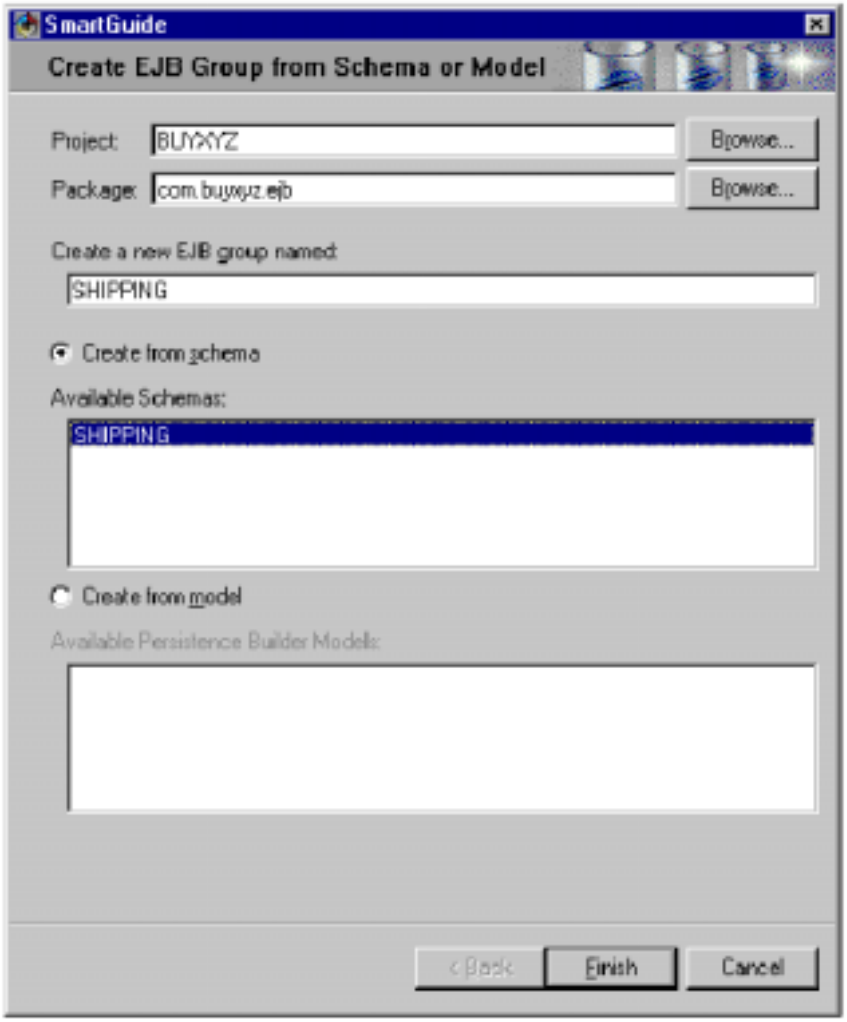

图 7-5　VisualAge 创建 EJB 组智能向导

2. 在“ Project ”和“ Package ”字段中输入先前指定的名称（BUYXYZ，com.buyxyz.ejb ）。您也可以使用 Browse...按钮选择适当的名称。输入新 EJB 组的名称并从可用模式中选择模式 SHIPPING。
3. 单击完成（Finish）完成设置。现在，VisualAge for Java 将分析该模式并生成 Java 类。
4. 最新创建的 EJB 组“ SHIPPING ”和 EJB“ Customer_order ”以及产品将出现在工作区中（见图 7-6）。

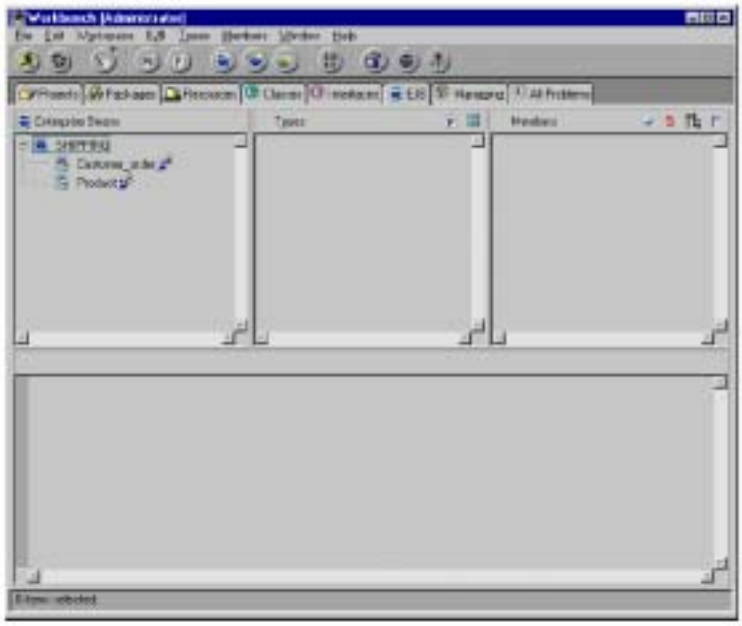

图7-6EJB 组SHIPPING

现在，在使用 WebSphere 测试环境和 EJB 测试客户机生成应用代码并测试 EJB 之前，我们将修改 EJB “ Customer_order ”的方法 ejbCreate()。

1. 在“ Members ”窗格中选择 Customer_order bean、Customer_orderBean 类型和 ejbCreate(string)方法。
2. ejbCreate(string)方法只接收一个输入参数。我们将使用其它“ Customer_order ”实体属性来扩展该方法。

图 7-7 显示了此修改结果。

```
public void ejbCreate(java.lang.String argOrder_id, int argProduct_id,
            int argCustomer_id, int argOrder_qty, int argOrder_price,
            java.lang.String argCust_name, java.lang.String argCust_address)
            throws javax.ejb.CreateException, java.rmi.RemoteException {
    _initLinks();
    // All CMP fields should be initialized here
    order_id = argOrder_id;
    customer_id = argCustomer_id;
    order_price = argOrder_price;
    order_qty = argOrder_qty;
    product_id = argProduct_id;
    shipping_address = argCust_address;
    shipping_name = argCust_name;
}
```

图 7-7　已修改的 ejbCreate()方法

3. 现在，我们将在 EJB 本地接口（EJB Home Interface）中添加新方法。在“ Members ”窗格中选择最新创建的方法并右键单击该方法，然后选择 Add To ->EJB Home Interface。
4. 如图 7-8 所示，现在新的 ejbCreate 方法已成为“ Customer_orderHome ” EJB 本地接口的一部分。

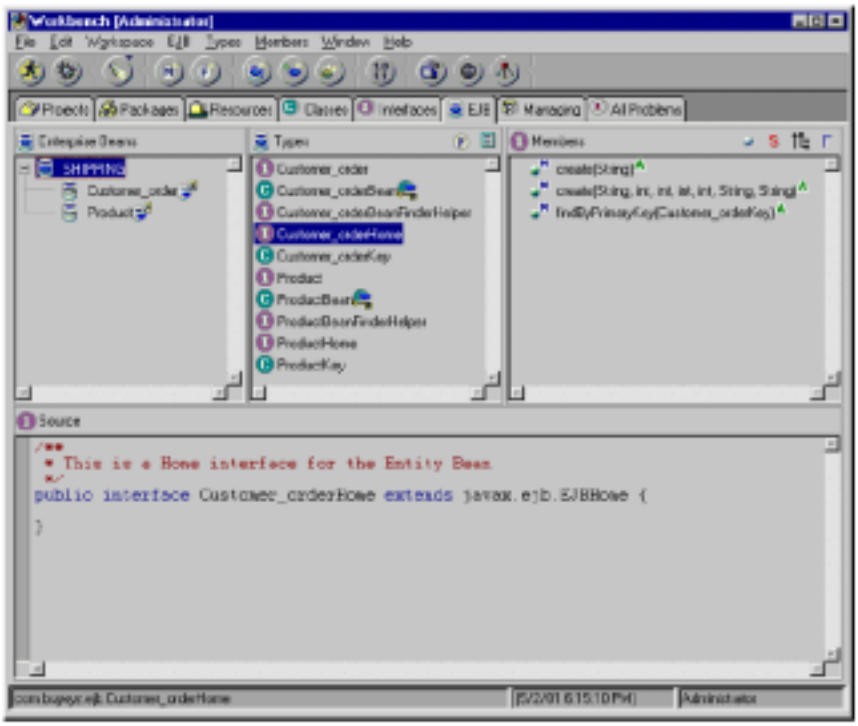

图 7-8 已修改的 EJB 组 SHIPPING

我们现在可以使用 EJB 测试客户机来测试 EJB。为了测试 EJB，我们应在 VisualAge for Java 中安装测试环境。

1. 首先，我们将生成应用代码。在“ Enterprise Beans ”窗格中选择 EJB 组 SHIPPING。选择 EJB -> 生成应用代码（Generate Deployed Code）。
2. 启动 WebSphere 测试环境。在工作台上，选择 Workspace -> Tools -> WebSphere Test Environment...，将出现该 WebSphere 测试环境。启动永久名服务器，确认该服务器提供了一个有效数据库以便初始化名称服务表。在“ Console ”窗口中，您可以检查该操作的结果。
3. 现在，我们将创建一个 EJB 服务器。转到“ Enterprise Beans ”窗格中所选的“ EJBSHIPPING ”组。在工作台中选择 EJB -> 添加（Add To）->服务器配置（Server Configuration），将出现 EJB 服务器配置浏览器窗口（图 7-9）。

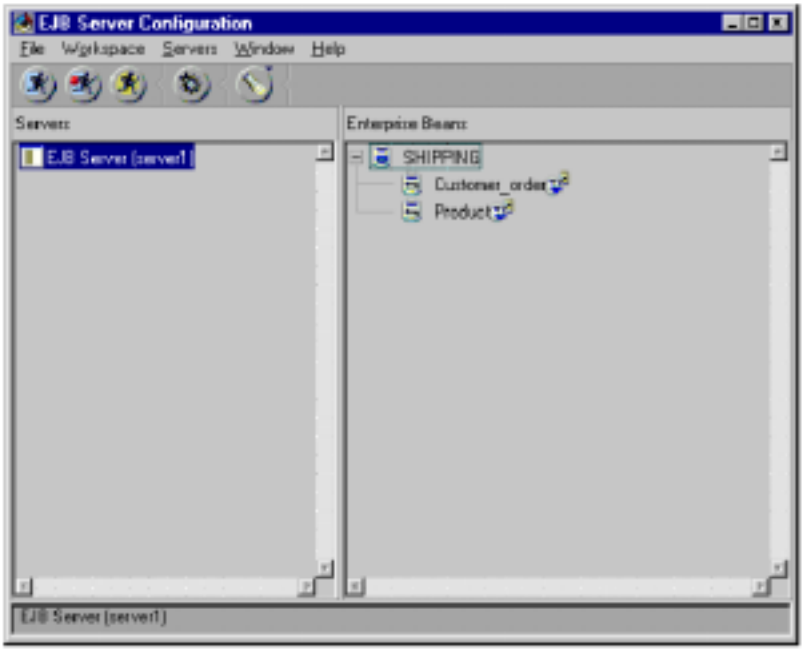

图 7-9 EJB 服务器配置浏览器

4. 选择服务器（server1）实例，然后右键单击该实例并选择**属性（Properties）**，将出现属性窗口（图 7-10）。指定数据库属性。

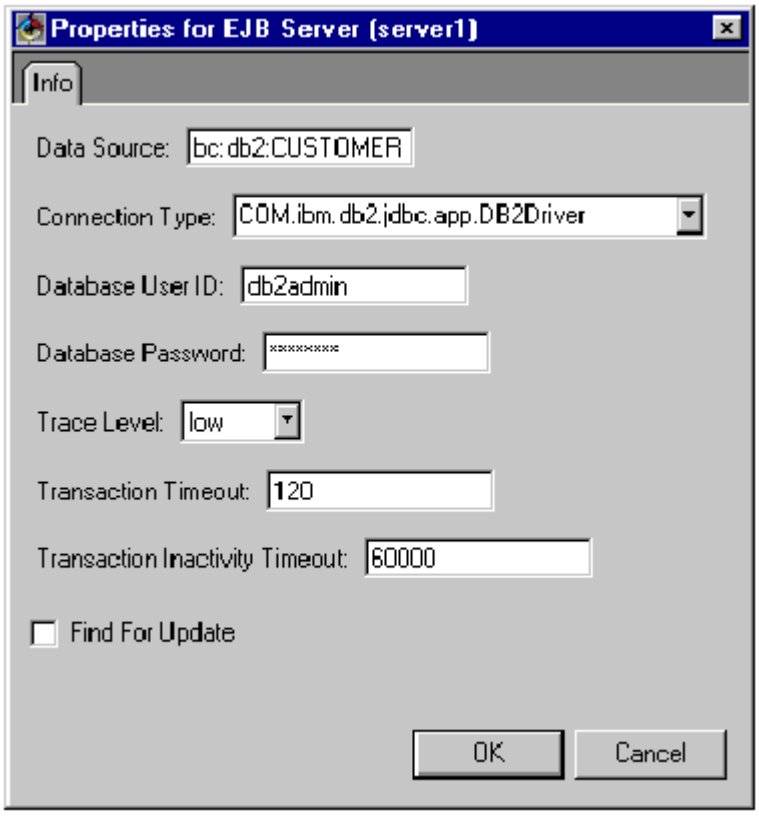

图 7-10 EJB 服务器属性

5. 在 EJB 服务器配置浏览器中启动 EJB 服务器。
6. 现在，我们可以运行 EJB 测试客户机了。为了运行该客户机，转到 EJB 服务器配置浏览器，选择并且右键单击 **EJB Group Shipping**，然后选择**运行测试客户（Run Test Client）**，将出现 EJB 测试客户机（图 7-11）。选择您要测试的适当的 JNDI 名。

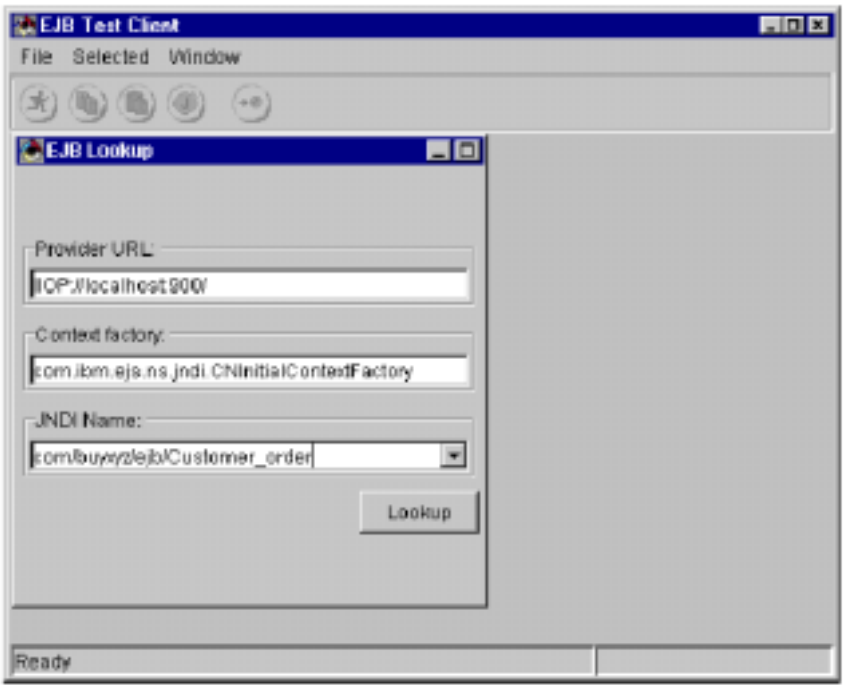

图7-11 EJB 测试客户机

7. 如果查找成功，将出现“ Home ”和“ Remote ”接口（如图 7-12）。您可以选择已修改的 ejbCreate()方法，通过指定输入参数并调用该方法来测试它。

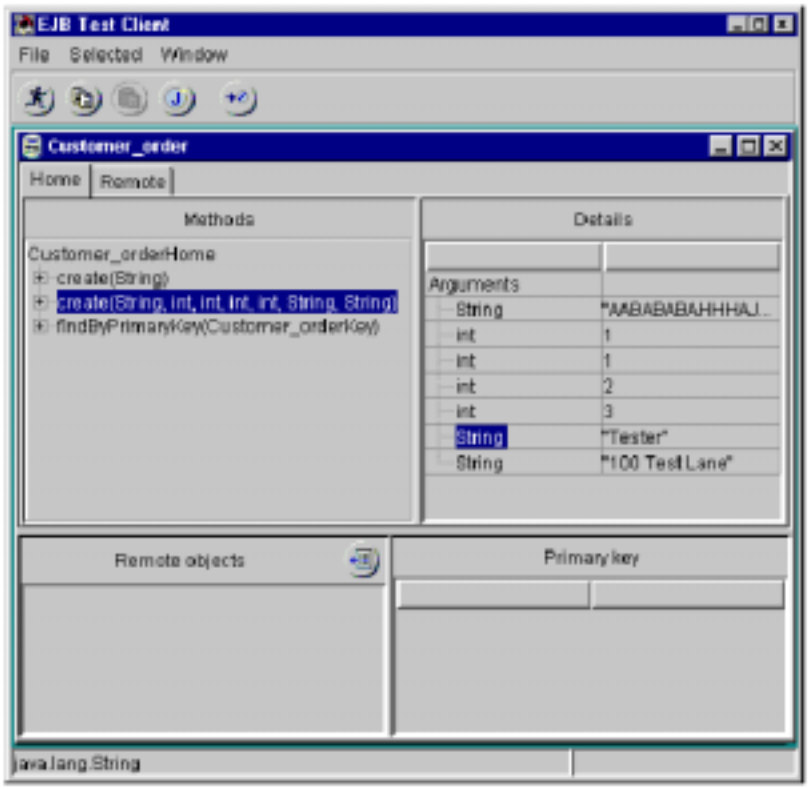

图7-12 EJB“Home”接口

现在，我们准备在 WebSphere 中应用 EJB。为此，我们首先要从 VisualAge 中导出 JAR 文件。

1. 在工作台中，从 EJB 标签的“Enterprise JavaBeans”窗格中选择 **EJB Group**。
2. 选择 **EJB -> Export -> Deployed JAR**。

提示:我们推荐您使用扩展 JAR 文件作为一个可扩展 EJB。扩展 JAR 文件包含 EJBHome 类、EJBObject 类、finder 类和 stub 以及框架文件。WebSphere 应用服务器将发现这些扩展类已经存在，因此它将不会试图重新生成它们。

强烈推荐您使用 VisualAge for Java 来应用用于传统应用程序或需要到一个数据库表的复杂映射的 bean。如果您在 WebSphere 管理控制台中使用自动应用进程，将不能保证生成表中的订单和列名与传统应用程序所需的表配置相匹配。

7.2.2 在WebSphere中应用EJB

在本节中，我们将完成把 EJB 应用到 WebSphere 应用服务器的高级步骤。关于更多详细信息，请参考《WebSphere V3.5 Handbook》编号：SG24-6161。

以下是在 WebSphere 中应用 EJB 的步骤：

1. 确认 IBM HTTP 服务器和 IBM WS AdminServer 服务已启动，同时 VisualAge for Java EJB 服务器已停止。
2. 通过选择 **Start -> Programs -> IBM WebSphere -> Application Server V3.5 -> Administrator's Console** 启动 WebSphere 管理控制台。
3. 通过单击向导按钮（如图 7-13 圈示）或单击 **Console -> Tasks -> Create Data Source** 启动创建数据源向导。

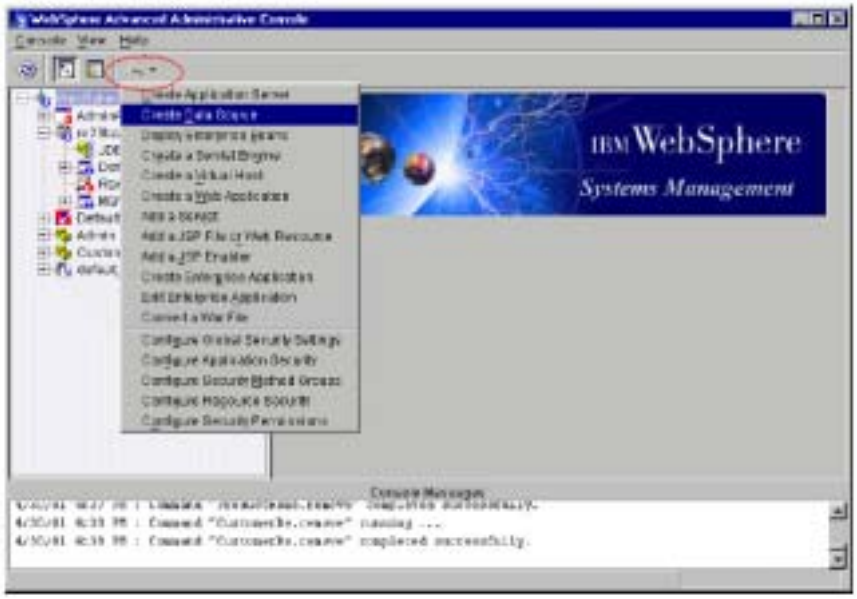

图 7-13 数据源定义

4. 选择 JDBC 驱动器。此案例中，我们使用现有的 Admin DB 驱动器（见图 7-14）。

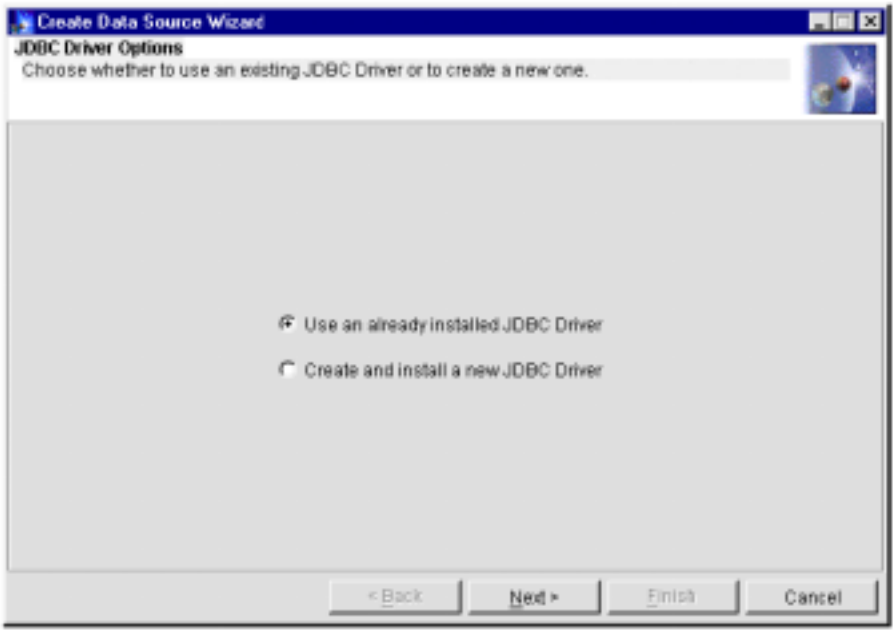

图 7-14 选择 JDBC 驱动器

5. 在如图 7-15 所示的窗口中，选择适当的驱动器并指定所需的数据源名和数据库名。单击**完成（Finish）**完成该设置。

图 7-15 创建数据源向导

现在已经定义了数据源，我们可以开始导入 EJB 了。

6. 将扩展 JAR 文件复制到 WebSphere\AppServer\deployableEJBs 目录中。
7. 从 WebSphere 高级管理控制台的拓扑视图中选择**<hostname>** ->**默认服务器（Default Server）** ->**默认容器（Default Container）**，您将看到如图 7-16 所示。

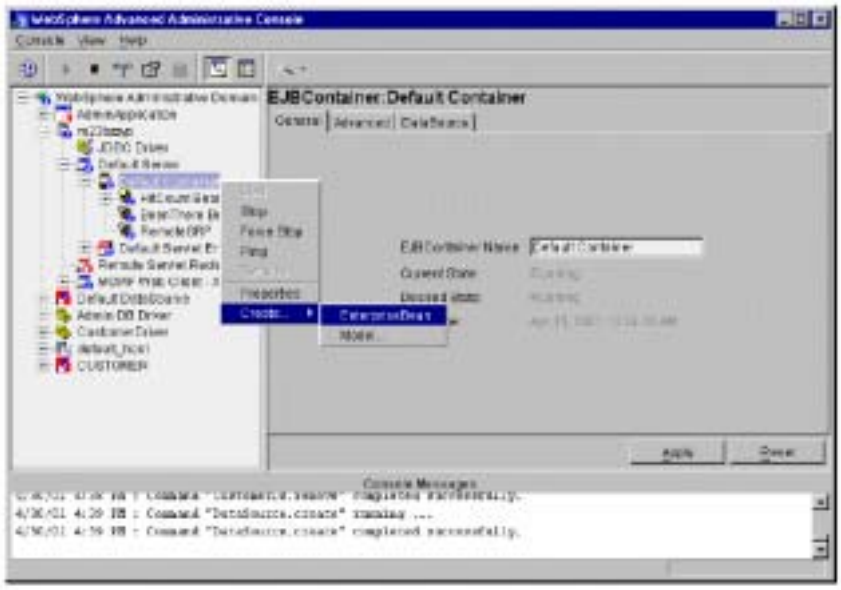

图 7-16 创建 EnterpriseBean

8. 右键单击**默认容器（Default Container）**并选择 **Create... -> EnterpriseBean**，将出现如图 7-17 所示的创建 EnterpriseBean 窗口。

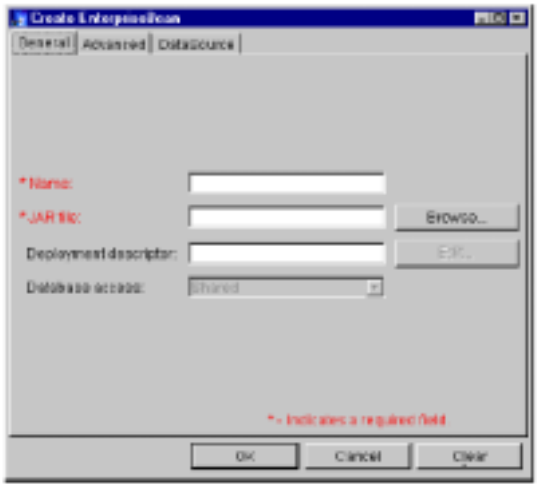

图 7-17 EnterpriseBean 设置

9. 从该窗口中选择 **Browse...**并找到存放 JAR 文件的目录。双击 JAR 文件，您将看到图 7-18。

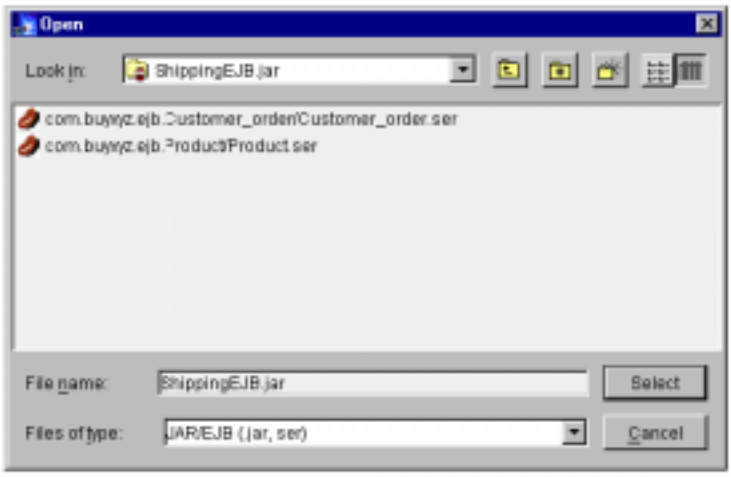

图 7-18 选择 JAR 文件

10. 选择与可扩展 EJB 相关的.SER 文件，您将看到 EJB 的参数（见图 7-19）。

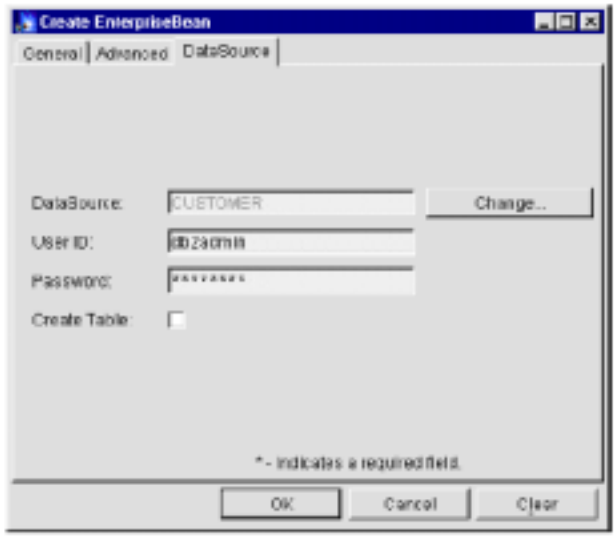

图 7-19 .SER 文件中的设置

11. 选择 **Datasource** 标签并且通过选择 **Change...** 按钮指定数据源，如图 7-20 所示。

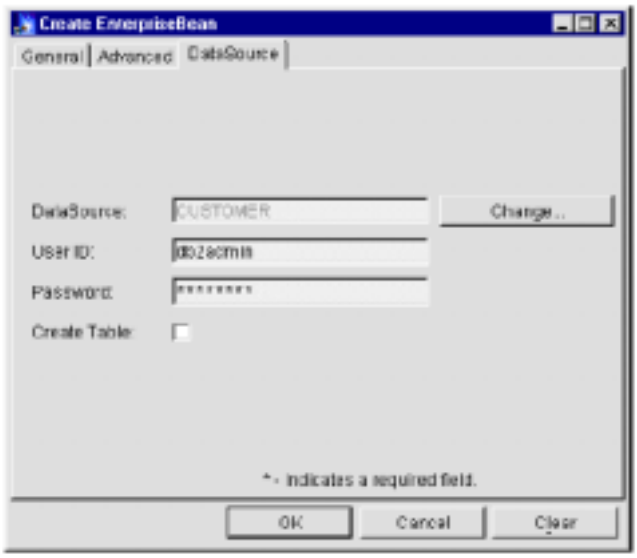

图 7-20 数据源访问设置

12. 在点击**确定（OK）**之后，该可扩展 EJB 将出现在“ 默认 ”容器中。

重复图 7-11 所示的步骤来应用其它 EJB。在应用成功后，您将看到图 7-21。

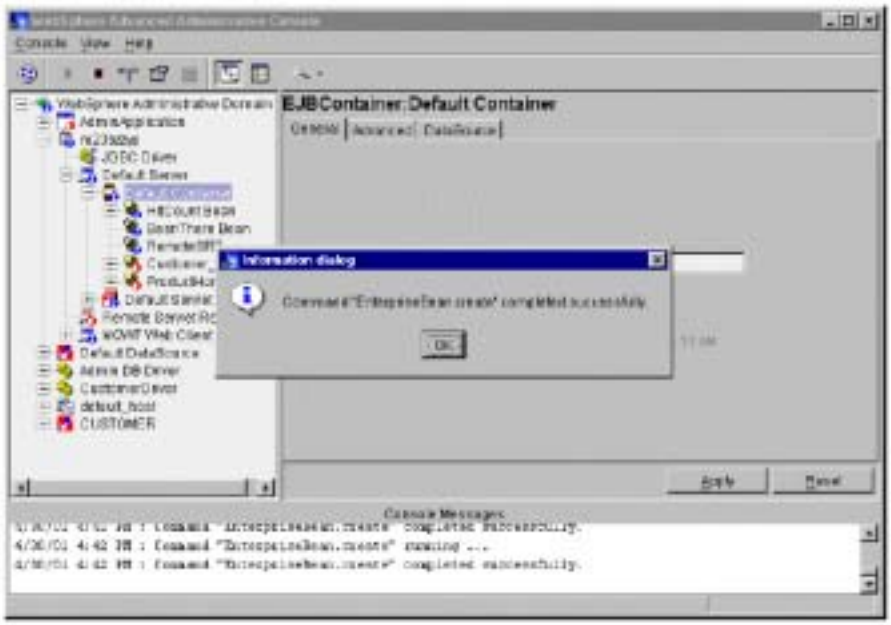

图 *7-21 创建和应用的 Enterprise JavaBean*

最后，在管理控制台中启最新应用的 EJB。

7.3 集成EJB的两种方法

集成 EJB 和 MQSeries Workflow 存在两种方法。第一种方法是执行读取 MQSeries Workflow XML 消息并调用 WebSphere 应用服务器外部 EJB 的 UPES。该 UPES 程序将充当 EJB 客户机程序。该方法的更多细节将在下节讨论。图 7-22 显示了这些组件怎样在一起工作。

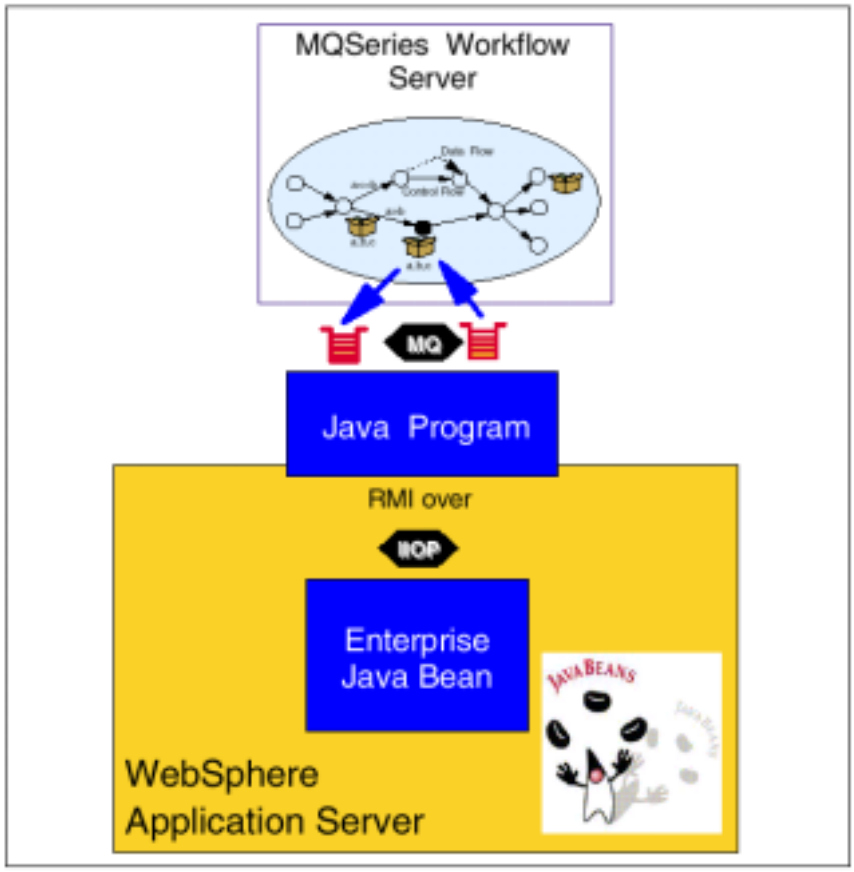

图 7-22 外部 EJB 客户机正在调用 WebSphere 应用服务器的 EJB

另一种方法是在 WebSphere 应用服务器中完整实施 EJB，但需要一个仅由 WebSphere Business Integrator 提供的组件。缺少的组件称为 JMSListener，它与 WebSphere 应用服务器中的 MQSeries Adapter Offering 正常运行时环境一起工作。图 7-23 显示了不同组件如何在一起工作。

JMSListener接收JMS消息，然后初始化“工人”消息bean并将该消息传给它。“工人”消息bean是无状态会话EJB。它将该消息转换成消息存储器对象。然后“工人”bean 分析消息以找出它必须调用哪个目标适配器。分析由处理 MQAK 头或 ApplicationID、bodyType 和 bodyCategory 的默认值组成。对于 EJB 适配器，EJB 处理程序将决定相应的服务会话 bean（本地接口）。然后，将应用程序数据转换成适当的数

据结构，如在微流中所示，并调用服务会话 bean。

JMSListener 将和无状态会话 bean 一起提供类似由 EJB 2.0 规范所定义的消息驱动 bean 的功能。

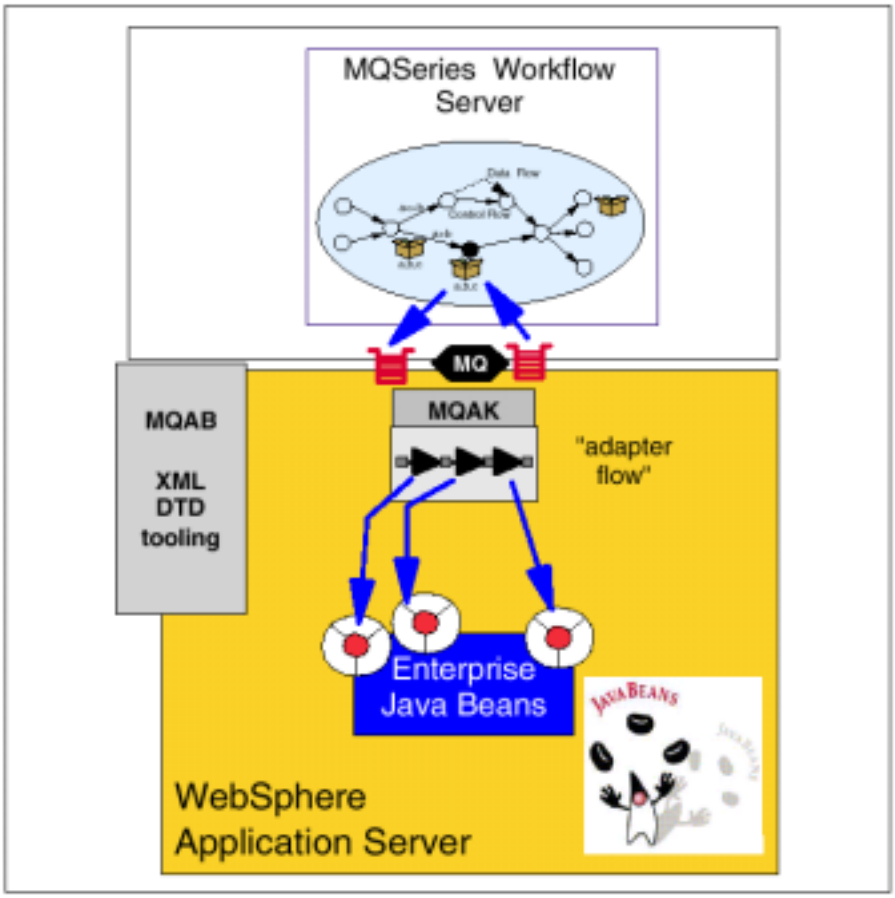

图 7-23 JMSListener 和会话 bean 调用 EJB

7.4 用Java开发UPES

因为还没有为该解决方案安装 WebSphere Business Integrator，所以我们将使用另一种方法来集成工作流中的 EJB。我们将开发一个运行于 WebSphere 应用程序服务器外部的简单 JMS listener 类。对于一条接收消息来说，JMS listener 充当客户机并且实例运送 EJB。

现在，我们有了 EJB 在数据库中存储所需的数据。在本节中，我们将略述 JMS listener 的开发。JMS listener 将侦听 UPES 输入队列，在接到消息之后，它将使用 SAX 解析器解析该消息并执行带有摘录字段的 EJB 方法。

我们将从 IncrementEJBUpes.java 代码开始叙述。

1. 我们需要定义称为“ ShippingEJBUpes ”的类。而且还将指定专用数据类“ Data ”来容纳引入字段，并且指定 EJB 本地变量来表达 7.2 节“ 开发和应用 EJB ”（302 页）中创建的 EJB。

实例 7-1 ShippingEJBUpes 类

```
public class ShippingEJBUpes implements ContentHandler
{
     /**Subclass to hold the parsed data */
     private class Data
     {
       /**Name of the 工作流 activity */
       public String activityName ="";
       /**ActImplCorrelID needed to correlate the XML request with the reply */
       public String actImplCorrelID ="";
       /**Name of the 工作流 process instance */
       public String instanceName ="";
       public String ProgramRC ="";
       /**Input data container */
       public int CustID =0;
       public int ProdID =0;
       public int ProdQty =0;
       public int ProdPrice =0;
       public String CustName ="";
       public String CustAddress ="";
  }
  /**EJB home */
  private Customer_orderHomeorderHome =null;
  private ProductHome prodHome =null;
}
```

2. 修改 SAX API 调用并在“ endElement ”方法中包含运送变量。实例 7-2 显示了该方法的相关部分。

实例 7-2 endElement SAX 解析方法

```
//Only a few elements are of interest ()
  if (latestTag.equals("_ACTIVITY"))data.activityName =currentString;
  else if (latestTag.equals("ActImplCorrelID"))data.actImplCorrelID =
currentString;
  else if (latestTag.equals("_PROCESS"))data.instanceName =currentString;
  else if (latestTag.equals("CustID"))
  {
       Integer i =new Integer(currentString);
       data.CustID =i.intValue();
  }
  else if (latestTag.equals("ProdID"))
  {
       Integer i =new Integer(currentString);
       data.ProdID =i.intValue();
  }
  else if (latestTag.equals("ProdQty"))
  {
       Integer i =new Integer(currentString);
       data.ProdQty =i.intValue();
  }
  else if (latestTag.equals("CustName"))data.CustName =currentString;
  else if (latestTag.equals("CustAddress"))data.CustAddress =currentString;
```

3. 指定构造器方法，修改 EJB Home 检索部分。

实例 7-3 ShippingEJBUpes 构造器

```
//Retrieve the EJB home ()

Object obj =cxt.lookup("com/buyxyz/ejb/Customer_order");
System.out.println(obj.toString());//just for debugging purposes
orderHome =(Customer_orderHome)javax.rmi.PortableRemoteObject.narrow(obj,
Customer_orderHome.class);

Object objp =cxt.lookup("com/buyxyz/ejb/Product");
System.out.println(objp.toString());//just for debugging purposes
prodHome =(ProductHome)javax.rmi.PortableRemoteObject.narrow(objp,
ProductHome.class);
```

4. 修改调用 EJB 方法的“ executeCommand ”方法。

首先，我们将调用“ Customer_order ” EJB 的创建方法来保存运送信息。

然后，我们将使用“ findByPrimaryKey ”和“ getStock ”方法来检索实际产品库存并使用“ setStock ”方法来更新库存。

实例 7-4 executeCommand 方法

```
Customer_order order;
Productprod;
ProductKeyprodkey;
int stock;

order =orderHome.create(data.actImplCorrelID,data.ProdID,data.CustID,
data.ProdQty,data.ProdPrice,data.CustName,data.CustAddress);

prodkey =new ProductKey(data.ProdID);
prod =prodHome.findByPrimaryKey(prodkey);
stock =prod.getStock();
stock =stock -data.ProdQty;
prod.setStock(stock);
```

5. 修改“ writeResponse ”方法以便在 XML 应答消息中使用“ DefaultDataStructure ”。

7.5 运行运送UPES

为了启动运送 UPES，我们需要做以下工作：

1. 在命令提示符下，运行 **setup_environment.cmd** 来初始化类路径。
2. 运行 **register_upes.cmd** 来注册 UPES。
3. 运行 **start_upes.cmd** 来调用应用程序。

关闭运送 UPES：

1. 在另一个命令提示符下，调用 **shutdown.cmd** 将“ QUIT ”消息写入队列。
2. 一旦 UPES 关闭，调用 **unregister_upes** 来释放 JMS 资源。

UPES 运行在 WebSphere 设备上并访问集群队列‘ MQSI.INPUT.SH ”,这些对象已在 5.9.1 节“ 定义 MQSeries 对象 ”（243 页）中定义过。

8

扩展模型：处理异常业务

在本章中，我们将扩展工作流模型以便当主供应商不能供货或没有按时应答时仍可以使用备用供应商。

当 MQSeries Workflow 发现“ OrderActivity ”到期时，它将发送另一条带有请求查寻提供特定产品的备用供应商的 MQSeries Workflow XML 消息给 MQSeries Integrator。

8.1 设计增加备用供应商的流程

如图 8-1 所示，在改进的解决方案中，我们增加了一个备用供应商以防主供应商不能满足需求或不能及时应答。

我们希望以一种可重用思想设计该方案。换句话说，我们不想彻底改造 MQSeries Integrator “供应订单”消息流。备用供应商必须能够以 BOD 标准格式接收供应订单。

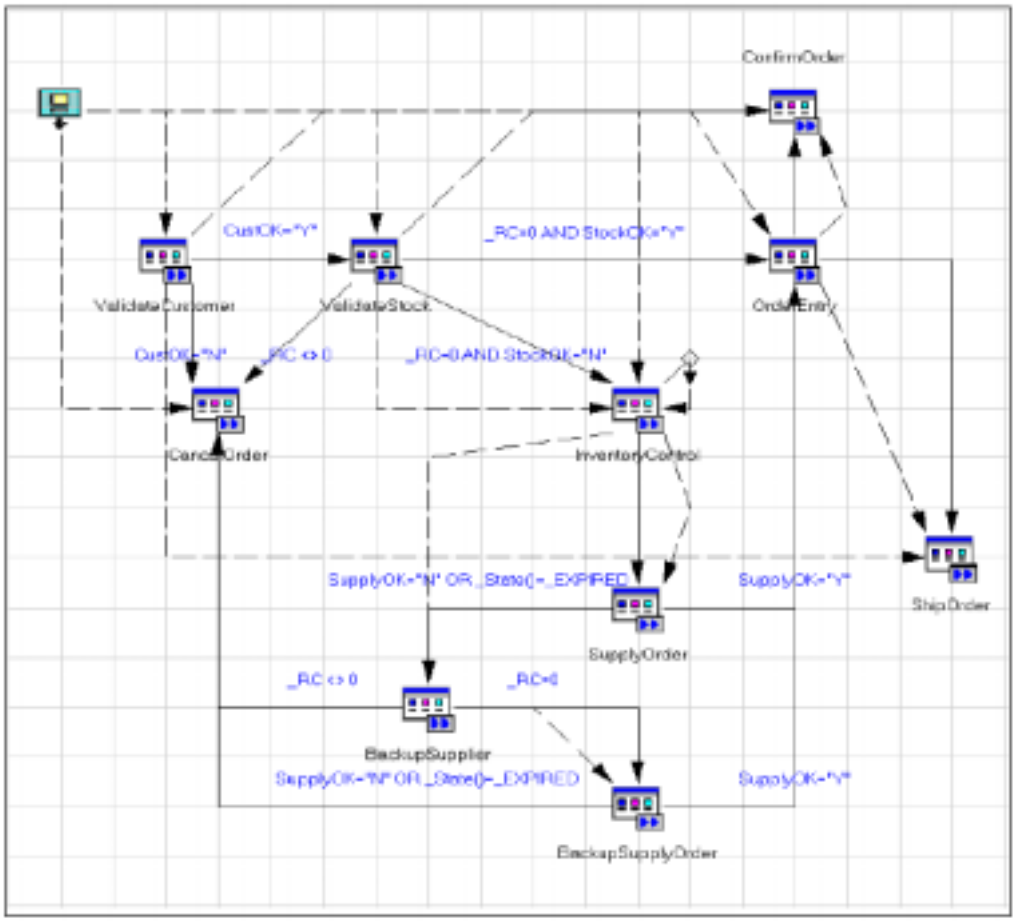

图 8-1 带有备用供应商的改进工作流

改进的工作流中包含额外的数据库查寻活动，它将决定是否需要一个备用供应商。如需要，它将调用“ SupplyOrder ” UPES 应用程序来使用备用供应商。

8.1.1 数据需求

数据库模式已经包含了一个表明该供应商是否是主供应商的字段。我们只需通过在表“ SUPPLIER_PRODUCT ”中添加行即可确定备用供应商。我们可以使用现有的数据结构来查询“ BackupSupplier ”查寻活动。“ BackupSupplyOrder ”活动将与“ 供应订单 ”活动完全相同。

8.1.2 添加新的流程到构建时模型

现在，我们将在构建时环境中修改订单流程。

活动定义

如 131 页“ 活动定义 ”所述，添加另外两个活动。我们将把这两个活动分别命名为“ BackupSupplier ”和“ BackupSupplyOrder ”。

“ BackupSupplier ”将使用“ SupplyInput ”数据结构作为输入和输出容器。“ BackupSupplyOrder ”将使用“ SupplyInput ”作为输入数据容器，“ SupplyValid ”作为输出数据容器。我们最初将使用哑元程序测试新的流程。

控制和数据流连接器

现在，我们将把新的活动与流程连接起来。参考本书第 136 页“ 控制流连接器 ”和第 139 页“ 数据流连接器 ”获得详细说明。

首先，我们将模拟一个控制流。

1. 删除从“ SupplyOrder ”到“ CancelOrder ”的控制连接器。
2. 添加从“ SupplyOrder ”到“ BackupSupplier ”的控制连接器。添加条件为“ SupplyOK="N"OR _State()=_EXPIRED ”。
3. 添加从“ BackupSupplier ” 到“ CancelOrder ” 带有条件“ _RC<>0 ”的控制连接器和从“ BackupSupplier ” 到“ BackupSupplyOrder ” 带有条件“ _RC=0 ”的控制连接器。
4. 添加从“ BackupSupplyOrder ” 到“ CancelOrder ” 带有条件“ SupplyOK="N"OR _State()=_EXPIRED ”的控制连接器和从“ BackupSupplyOrder ” 到“ OrderEntry ”带有条件“ SupplyOK=”Y ” ”的控制连接器。

这样就完成了控制流程。

现在，我们将模拟一个数据流特征。

1. 添加从“ InventoryControl ”到“ BackupSupplier ”的数据连接器。
2. 添加从“ BackupSupplier ”到“ BackupSupplyOrder ”的数据连接器。

因为输出和输入容器是相同的，所以没有必要做额外的数据映射。

完成的模型应与本书第 328 页图 8-1 所示的模型相似。

导出模型

从构建时导出新的流程模型并将文件 FDL 导入正常运行时环境。

提示：在将新的 FDL 文件导入正常运行时环境之前，请在步骤“ 存储控制 ”中留下几个流程实例。

8.2 测试新的流程和条件

现在，我们将使用本书第 153 页 4.2.3 节“ 使用带有默认 Web 页的 Web 客户机测试流程 ”所述的方法来测试新的流程。

注意，旧的流程实例不会试图利用备用供应商。现有的流程实例总是在实例构建时使用存在于正常运行时数据库中的模板。对于短期流程，这一般不成问题。但对于长期流程，这可能就有问题了。我们可以通过使用子流并动态指定子流程调用来解决该问题。同时，可以在现有实例正常运行时替换流程模板这一事实证明您可以非常迅速地对业务条件改变做出反应。

8.3 为备用供应商创建新的UPES

在工作流中加入 UPES 还需要两个步骤。

备用供应商 UPES 定义

按照已在本书第 168 页 4.4 节“ 定义 UPES 活动 ”中略述过的步骤，我们将创建一个称为“ XYZBACK ”的新 UPES，它使用队列“ MQSI.INPUT.BS ”来支持备用供应商数据库查寻。

修改活动定义

现在，我们将用“ UPES_Program ”来替换新活动中的哑元程序。我们将使用新的“ XYZBACK ” UPES 来支持“ BackupSupplier ”活动并重新使用现有的“ XYZSUPP ” UPES 来支持“ BackupSupplyOrder ”活动。

参考本书第 175 页的“ 修改活动定义 ”来获得该步骤的详细说明。

8.4 为备用供应商创建MQSI UPES应用程序

为了选择备用供应商，我们将设计一个新的流程，并保持现有的流程不变。

图 8-2 显示了完整消息流。

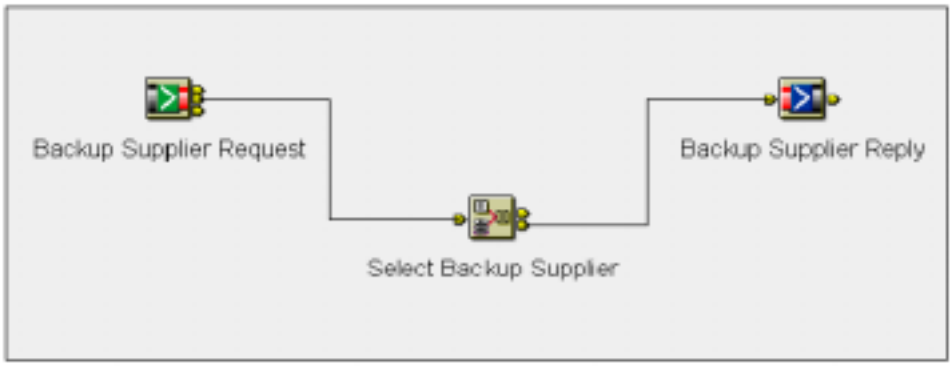

图 8-2 消息流概览

该消息流的任务是在入站消息中选择一个提供指定产品的备用供应商。该备用供应商应不同于以前的供应商。为了避免同一供应商再次被选中，以前所选的供应商标识符将成为引入消息的一部分。

8.4.1 输入和输出数据结构

我们使用同样的容器数据结构（SupplyInput）作为输入和输出消息。这是执行新活动的最快也是最容易的方法。

实例“ SupplyInput ” XML 输入消息的“ ProgramInputData ”文件夹如下所示。

```
<ProgramInputData>
<_ACTIVITY>BackupSupplier</_ACTIVITY>
<_PROCESS>Order Process</_PROCESS>
<_PROCESS_MODEL>Order Process</_PROCESS_MODEL>
<SupplyInput>
<SupplierID>1</SupplierID>
<ProdID>1</ProdID>
<ProdDesc>CD1</ProdDesc>
<ProdQty>21</ProdQty>
</SupplyInput>
</ProgramInputData>
```

在输出数据结构中，“ SupplierID ”字段已更改为备用供应商。

关于完整的输入和输出消息实例，请参考本书第 397 页附录 B “ 实例应用程序安装 ”。

8.4.2 消息流细节

本节将描述消息流“ BuyXYZ_Backup_Supplier ”中每个节点的细节。

MQInput 节点“ Backup Supplier Request ”

图 8-3 和图 8-4 显示了“ MQInput ”节点的配置面板。

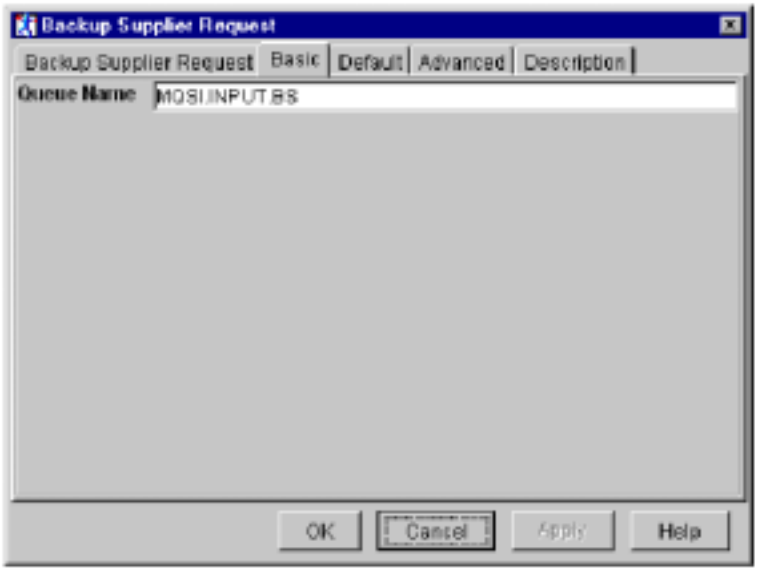

图 8-3 “ MQInput ”节点细节

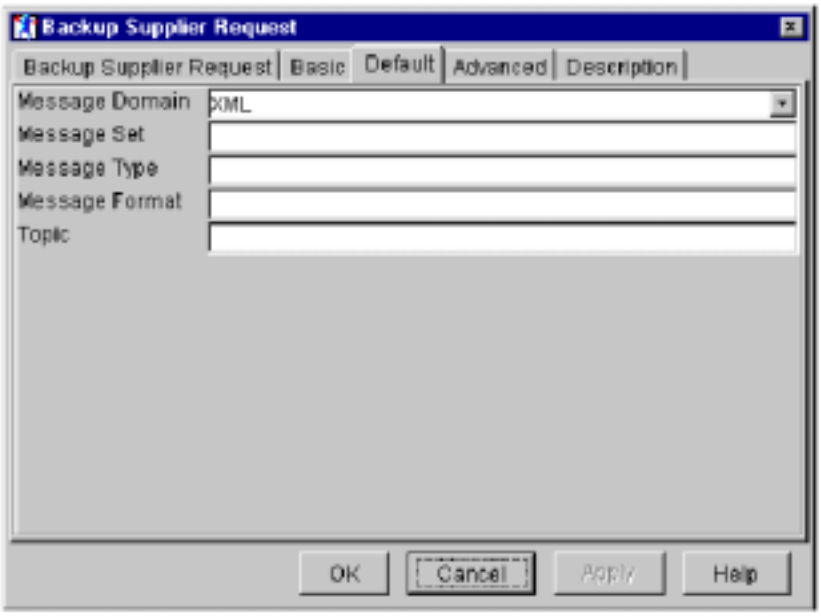

图 8-4 “MQInput”节点细节

该“MQInput”节点具有默认设置。它将使用第.1 节“活动执行概述”（180 页）中所述的方法从给定的输入队列中读取 XML 消息。

计算节点“Select Backup Supplier”

图 8-5 显示了该计算节点的属性。

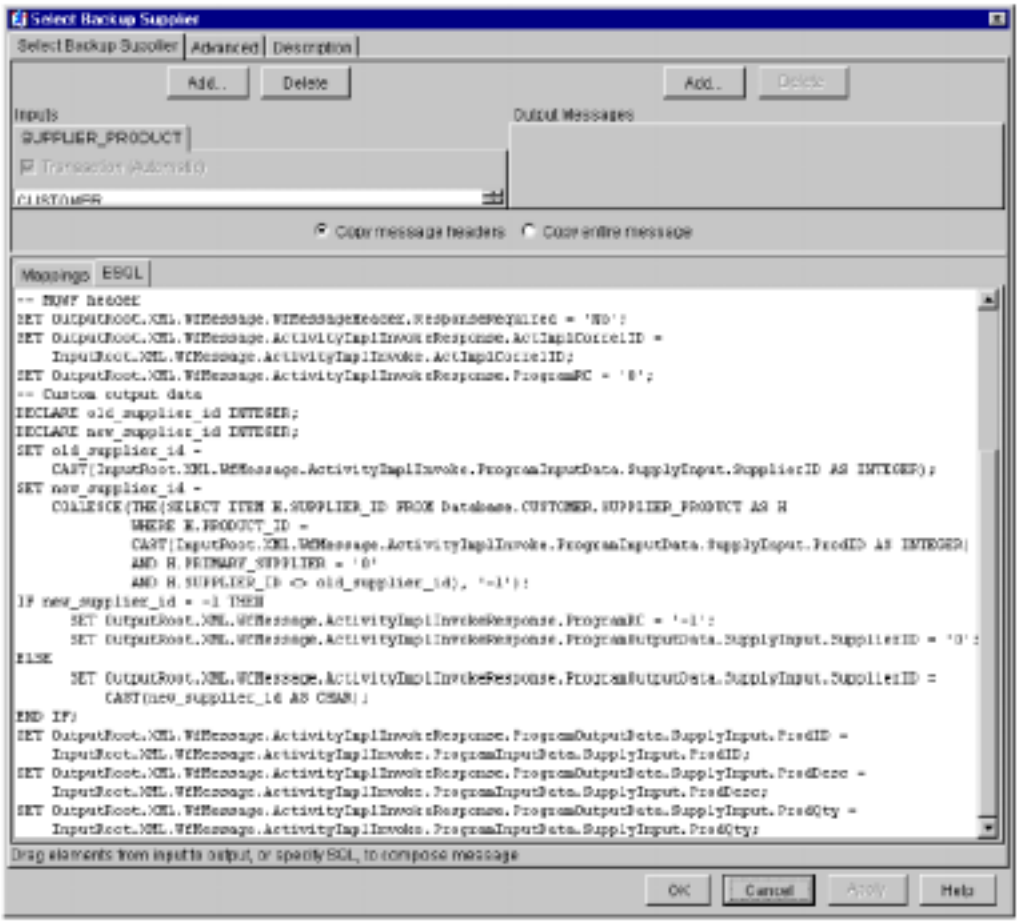

图8-5 计算节点细节

该计算结点接收产品标识符和供应商标识符作为输入并检索给定产品的另一个供应商，该供应商不同于先前的供应商。

```
SET old_supplier_id =
CAST(InputRoot.XML.WfMessage.ActivityImplInvoke.
ProgramInputData.SupplyInput.SupplierID AS INTEGER);
SET new_supplier_id =
COALESCE(THE(SELECT ITEM H.SUPPLIER_ID FROM Database.CUSTOMER.SUPPLIER_PRODUCT
AS H
WHERE H.PRODUCT_ID =
```

```
CAST(InputRoot.XML.WfMessage.ActivityImplInvoke.
ProgramInputData.SupplyInput.ProdID AS INTEGER)
AND H.PRIMARY_SUPPLIER ='0'
AND H.SUPPLIER_ID <>old_supplier_id),'-1');
```

数据库查询的结果将是备用供应商的供应商标识符或-1（在没有其它供应商时）。

该节点将检查查询结果并设置“ ProgramOutputData ” XML 消息字段的数值。在这种情况下，输入和输出数据结构是相同的，因此唯一要做的事情就把其它消息字段复制到输出消息的“ ProgramOutputData ”文件夹中。

MQOutput 节点“ Backup Supplier Reply ”

图 8-6 和图 8-7 显示了该“ MQOutpout ”节点的“ Basic ”和“ Advanced ”属性。

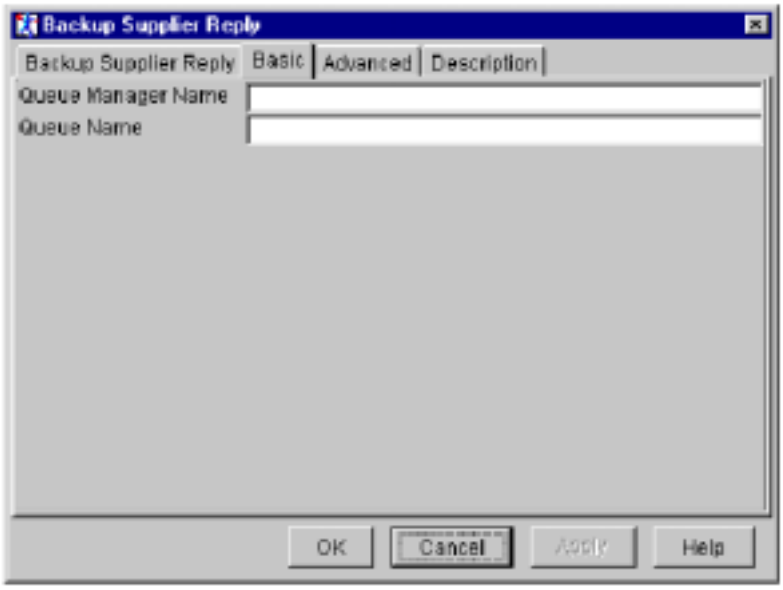

图 8-6“ MQOutput ”节点细节

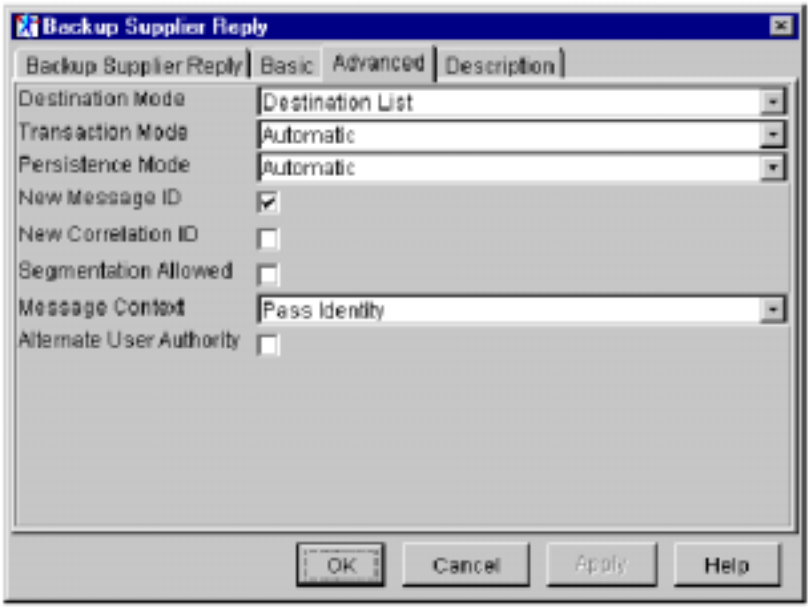

图 8-7“MQOutput”节点细节

该“MQOutput”节点使用“Destination List”作为“Destination Mode”把消息传到适当的队列。

使用这一附加的消息流，当第一个订单过期或主供应商不能交付所需的数量时，我们的实例应用程序能够把购买订单发给其它的供应商。

8.5 新问题——价格上涨

备用供应商已经提高了定价，这将危及我们的利润。我们将在价格过高时引入一个审批步骤。在本书第 337 页第九章“扩展模型：实现基于角色的交互”中，我们将看到如何处理另一个异常业务。

9

扩展模型：实现基于角色的交互

因为备用供应商不会总是给我们最合适的价格，所以我们将再次扩展该模型以便在供应商的价格过高时包含一个审批步骤。

为了防止审批请求在某人的审批目录中停留过长时间（如果这样，我们将不能满足客户服务水平协议的要求），审批活动将以这样一种方法来执行，即在确定的时间范围内，较低级管理者不批准或拒绝该请求时，审批请求将自动转给更高级管理者。

9.1 职员决议和异常管理

在新的方案中，我们将引入由 MQSeries Workflow 提供的一些异常管理和职员决议特性。现在，我们将演示基于角色的动态路由和自动异常处理。

改进的工作流应标记供应商向我们索取的价格比我们向客户索取的价格更高的条目，并且将这些条目发送给支持审批的异常队列。如果异常条目在指定的时间内没有被响应，它们将被转给更高级的管理者。

9.1.1 角色定义

我们将为 BuyXYZ 公司定义两个角色：

1. 从构建时中选择 Staff 标签。
2. 右键单击 Roles 并选择 New Role，将出现如图 9-1 所示的窗口。
3. 在名称区域中键入 BuyXYZ_Clerk。

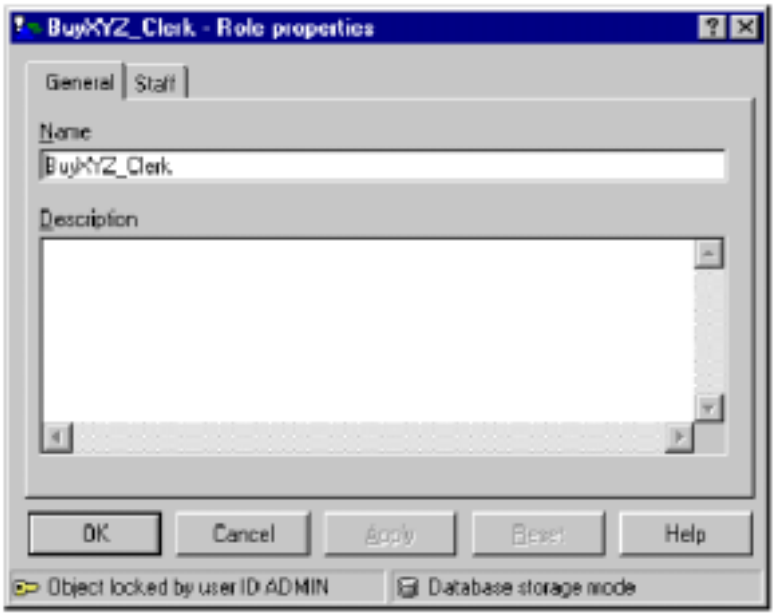

图 9-1BuyXYZ 职员角色

重复这些步骤来创建 BuyXYZ 管理者角色（见图 9-2）。

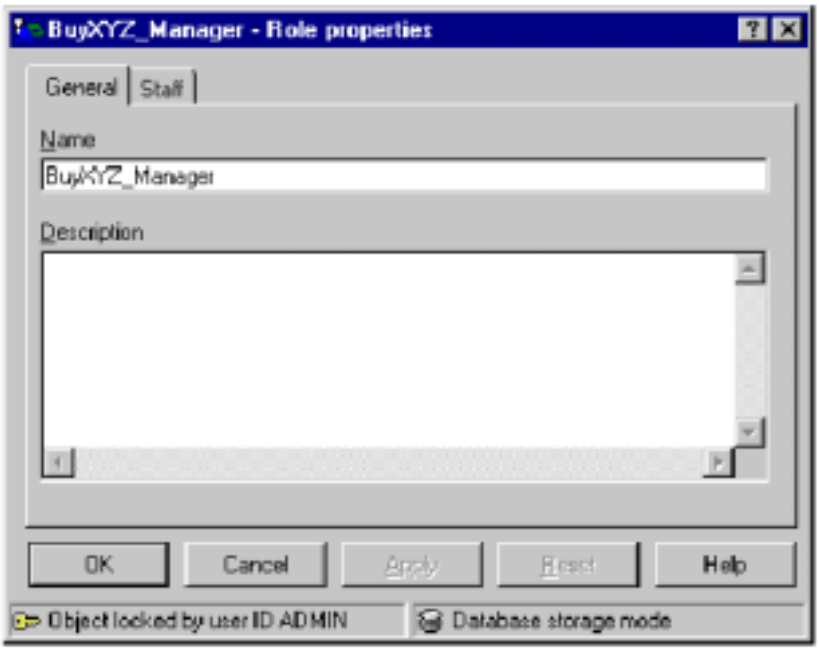

图 9-2 BuyXYZ 管理者角色

9.1.2 全体职员定义

现在，我们将定义在 BUYXYZ 订单流程中执行手动活动的人。我们将以一般名称代替真实名称。

1. 从构建时中选择 Staff 标签。
2. 右键单击 Persons 并选择 New Person，将出现如图 9-3 所示的窗口。
3. 在名称区域中键入 BUYXYZ_CLERK。
4. 指定默认口令。

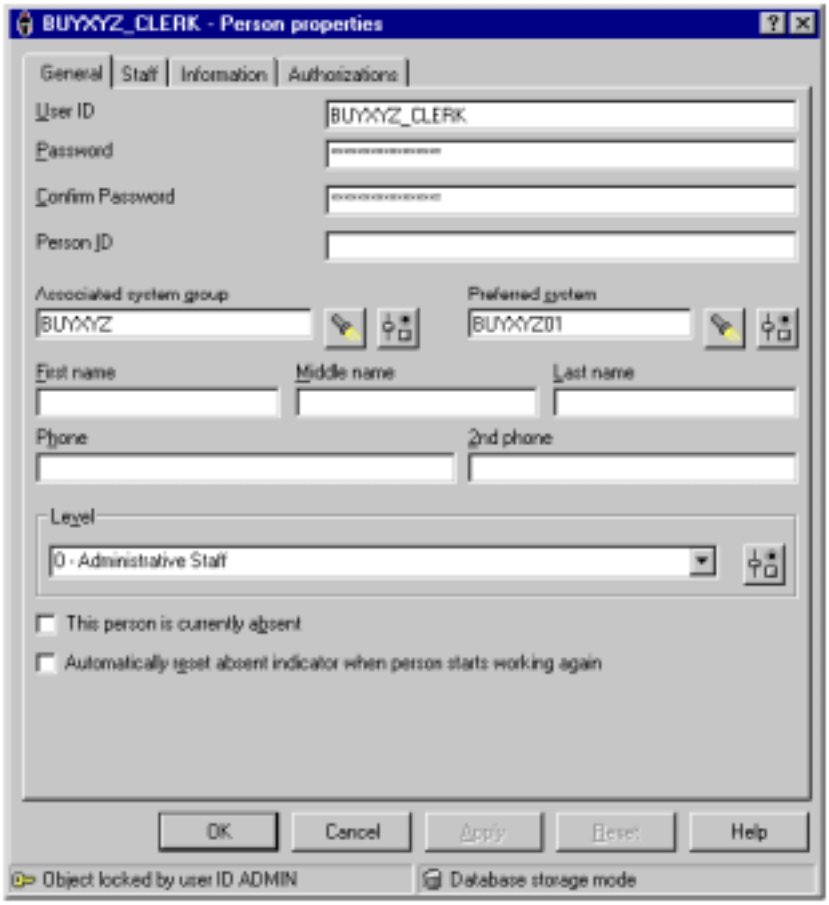

图 9-3 人称属性通用标签

提示：级别名可以通过扩展级别树来修改，然后双击您希望修改的级别号。

5. 选择 Staff 标签。
6. 单击“ Member of roles ”边框中的手电筒图标（箭头所指），如图 9-4 所示，将出现“ Find Role ”窗口。
7. 选择 BuyXYZ_Clerk，结果如图 9-4 所示。

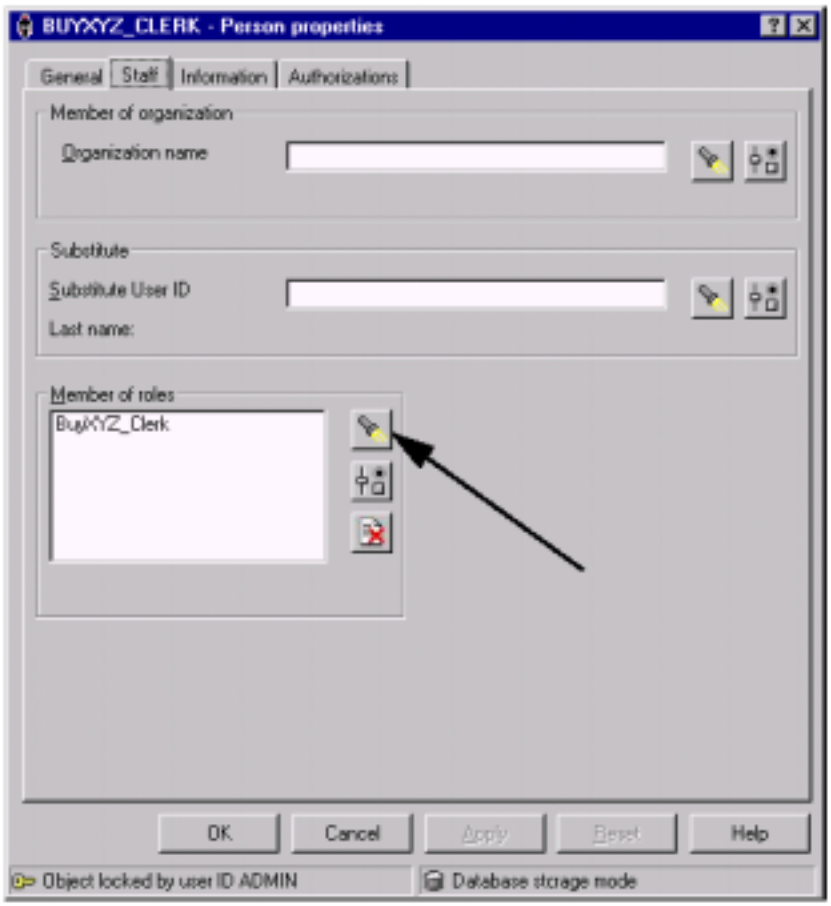

图 9-4 人称属性中的职员标签

8. 选择认证（Authorizations）标签，如图 9-5 所示。
9. 单击“ Categories ”边框中的手电筒图标，将出现“ Find Category ”窗口。
10. 选择 BUYXYZ Processes。

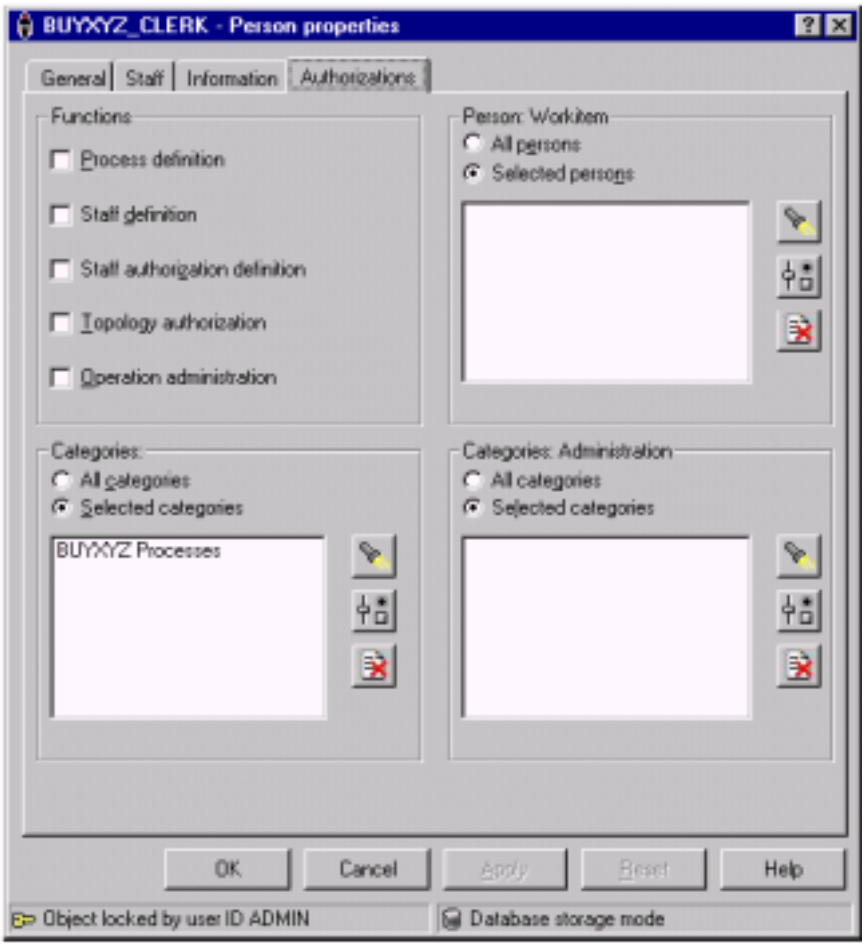

图 9-5 人称中的认证标签

11. 单击确定（OK）。

现在，我们将重复这些步骤来创建一些管理者，并为每位管理者分配不同的级别。

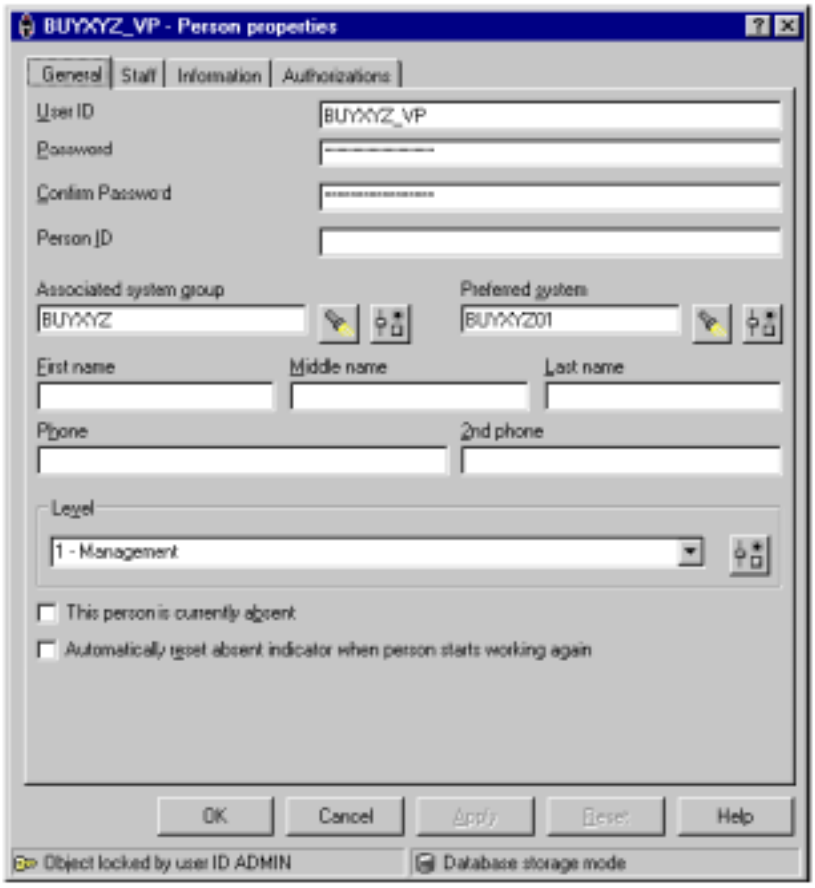

图 9-6 BuyXYZ 管理者 1 定义

1. 如图 9-6 所示，选择 Level 1，它是我们给定的管理人员描述。
2. 单击 Staff 标签并选择 BuyXYZ_Manager。

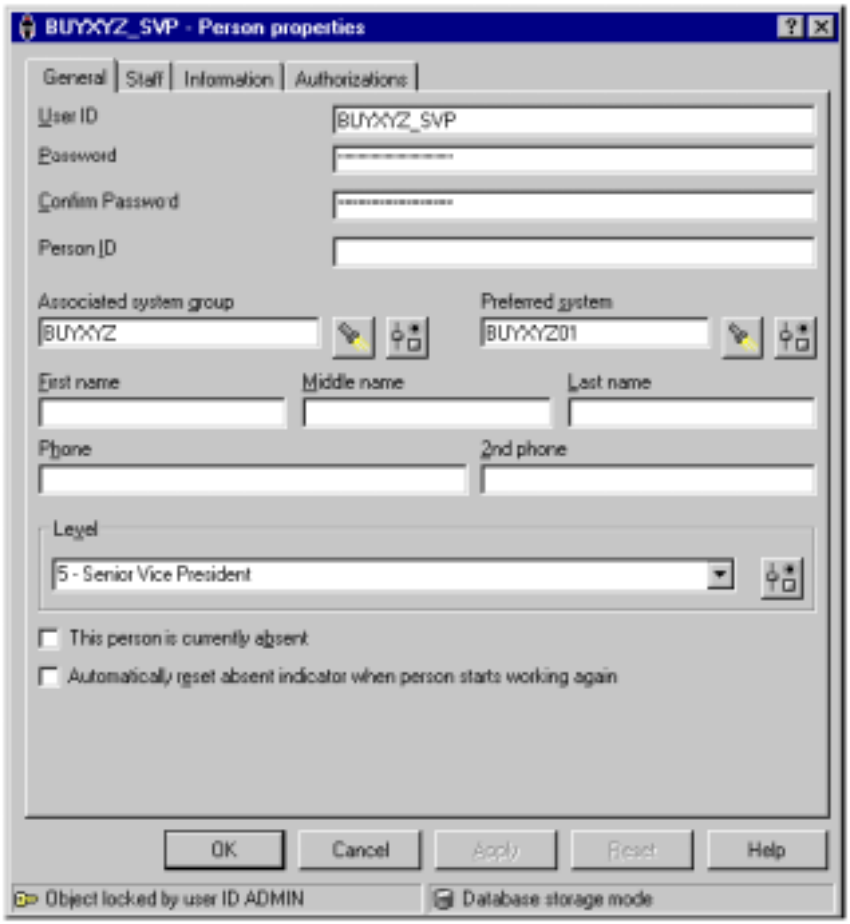

图 9-7 BuyXYZ 管理者 2 定义

3. 如图 9-7 所示，选择 Level 5，它是我们给定的高级副总经理描述 Senior Vice President.。
4. 单击 Staff 标签并选择 BuyXYZ_Manager。

9.1.3 活动staffing定义

现在，我们将修改活动定义来利用基于角色的手动活动分配。

1. 打开“ CancelOrder ”活动的活动属性窗口，如图 9-8 所示。
2. 选择 Staff 1 标签。
3. 选择 Dynamic assignment from page 2..。

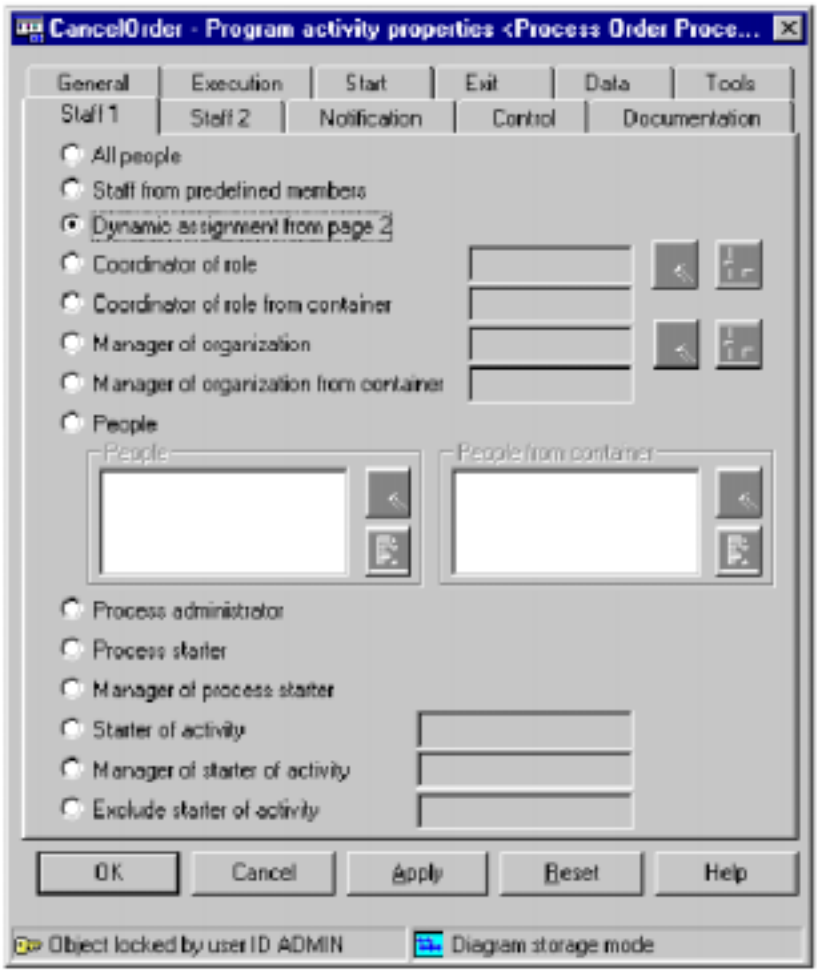

图 9-8 Staff 1 设置

4. 选择 Staff 2 标签，如图 9-9 所示。
5. 单击“ Member of roles ”边框上的手电筒图标，将出现“ Find roles ”窗口。
6. 选择 BuyXYZ_Clerk。

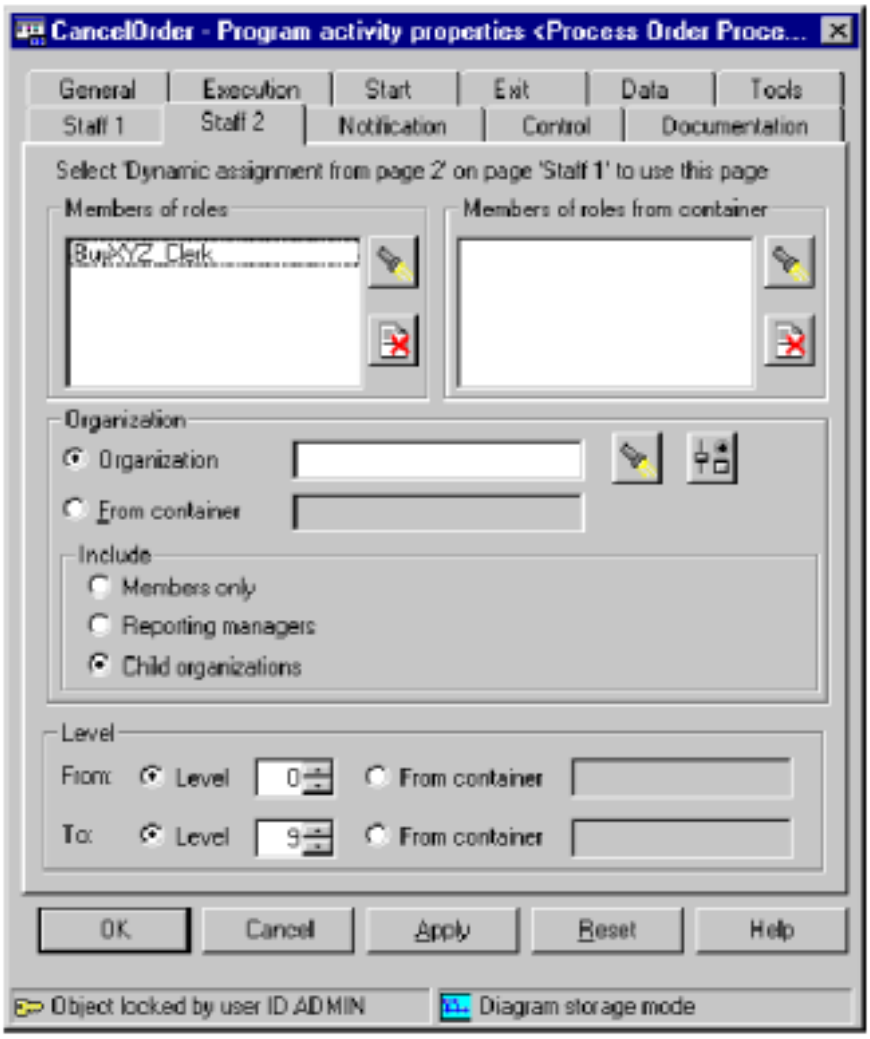

*图*9-9 *Staff 2 设置*

7. 单击确定（OK）。

对确认订单活动重复这些步骤。现在，任一作为角色“ BuyXYZ_Clerk ”成员的用户都将收到关于取消订单和确认订单活动的条目。

我们将对存储控制活动再次重复这些步骤，但有以下不同：

- 选择 BuyXYZ_Manager 作为角色。
- 将级别设为 4（见图 9-10）。

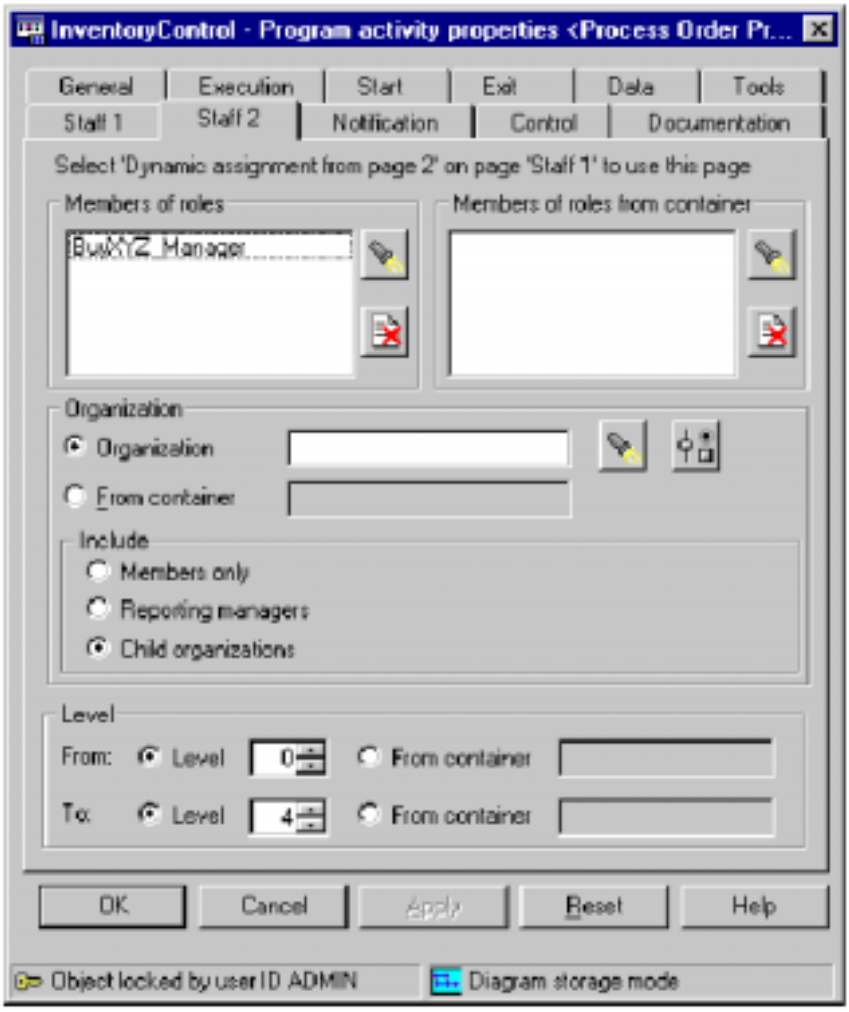

图 9-10 存储控制活动的职位分配。

这将把存储控制活动路由给任一级别为 4 级或 4 级以下的管理者。

9.2 设计定价异常流程

如果供应商所定价格高于或等于我们向客户索取的价格，将会触发新的审批流程。

9.2.1 数据需求

为了判断该条件，我们将需要在“ OrderInfo ”数据结构中指定产品价格。一种选择是在现有的数据结构（即“ SupplyInput ”和“ SupplyValid ”）中加入“ ProdPrice ”字段。然而，如果选择该选项，我们将不得不修改使用现有数据结构的 UPES 应用程序。所以，我们将创建一个新的数据结构“ CheckPrice ”，它将包含价格检查所需的字段（图 9-11）。

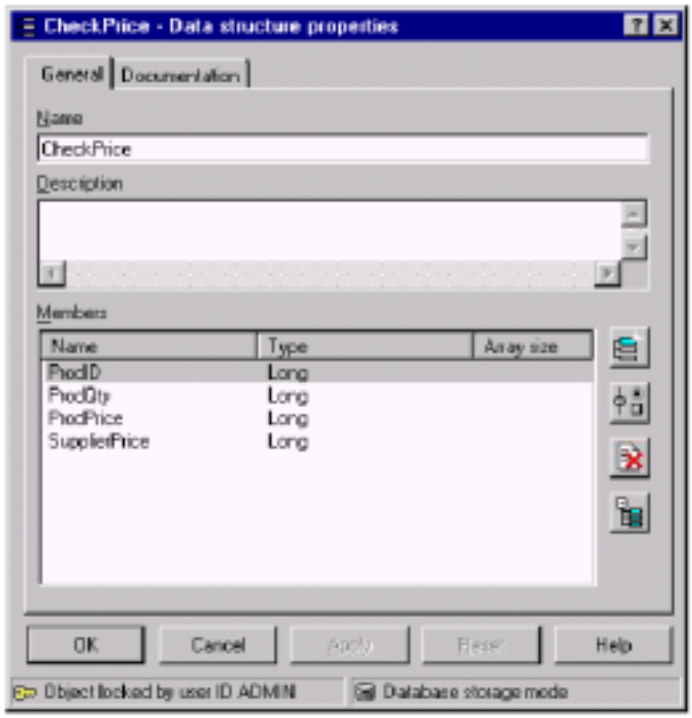

图 9-11 新数据结构“ CheckPrice ”

我们需要一个哑元价格检查活动，以便我们能够将“ SupplyValid ”数值连同“ ProdPrice ”从“ OrderInfo ”映射到新数据结构“ CheckPrice ”中去。我们将使用默认数据连接器来把输入数值传给输出容器，以便我们能够执行有条件的价格检查。

9.2.2 活动定义

如 131 页“ 活动定义 ”所述，另外添加三个活动。我们将之分别命名为“ CheckPrice ”、“ Approval ”和“ Escalation ”。

“ CheckPrice ”活动事实上只是将数据与“ CheckPrice ”输入容器结合并将内容传给输出容器以进行判断。然而，我们仍需指定一个程序。我们可以指定一个执行“ fmcnop.dll ”的哑元程序执行，但这将需要运行 PEA (Program Execution Agent)登录用户来自动执行该活动。我们将代之以创建一个哑元 UPES 应用程序。按照已在第 4.4 节“ 定义 UPES 活动 ”（168 页）中概述过的步骤，创建一个称为“ XYZDUMMY ”的新 UPES 并将其与“ CheckPrice ”活动相关联。对于该活动我们不需要应答，所以我们可以在“ Execution ”标签中把模式设为异步模式。参看第 9.3 节“ 创建 BuyXYZ_Dummy 消息流 ”（352 页）以获取关于哑元 MQSI 消息流的细节。我们将使用“ CheckPrice ”数据结构作为输入和输出容器。

“ Approval ”和 “ Escalation ”活动将使用“ CheckPrice ”数据结构作为输入容器，“ SupplyValid ”作为输出容器。我们最初将指定哑元程序来模拟新的流程。随后，我们将使用在 9.4 节“ 为审批活动创建 JSP ”（353 页）中创建的 JSP 来更新该流程。

“ Approval ”活动的职员分配需要任一级别 0 到级别 4 的 BuyXYZ_Manager 角色。对于“ Escalation ”活动，它将被设为任一级别 5 到级别 9 的 BuyXYZ_Manager 角色。

我们也将为“ Approval ”活动设置终止期限。

9.2.3 控制和数据连接器

我们可以在流程中添加新的活动。请参考第 136 页的“ 控制流连接器 ”和“ 数据流连接器 ”来获取详细介绍。

现在，我们将模拟修改控制流和数据流：

1. 删除从“ BackupSupplyOrder ”到“ OrderEntry ”的控制连接器。
2. 添加从“ BackupSupplyOrder ”到“ CheckPrice ”的控制连接器。将转换条件设为 SupplyOK="Y"。
3. 添加从“ BackupSupplyOrder ”到“ CheckPrice ”的数据连接器。映射“ SupplierPrice ”。
4. 添加从“ Source Node ”到“ CheckPrice ”的数据连接器。映射“ ProdID ”、“ ProdQty ”和“ ProdPrice ”字段。
5. 添加默认数据连接器到“ CheckPrice ”活动。
6. 添加从“ CheckPrice ”到“ OrderEntry ”的控制连接器。将转换条件设为“ ProdPrice >SupplierPrice ” 。

7. 添加从“ OrderEntry ”到“ Approval ”的控制连接器。将转换条件设为“ ProdPrice <=SupplierPrice ”。
8. 添加从“ CheckPrice ”到“ Approval ”的数据连接器和从“ CheckPrice ”到“ Escalation ”的数据连接器。
9. 添加条件为“ SupplyOK=”Y ” ”的从“ Approval ”到“ OrderEntry ”的控制连接器，和条件为“ SupplyOK=”N” ” 的从“ Approval ”到“ CancelOrder ”的控制连接器。
10. 添加从“ Approval ”到 Escalation “ ”的控制连接器。将转换条件设为“ _State()=_EXPIRED ”。
11. 添加条件为“ SupplyOK=”Y ” ”的从“ Escalation ”到“ OrderEntry ”的控制连接器，和条件为“ SupplyOK=”N” ” 的从“ Escalation ”到“ CancelOrder ”的控制连接器。

在完成这些任务之后，定价异常流程示图将如图 9-12 所示。

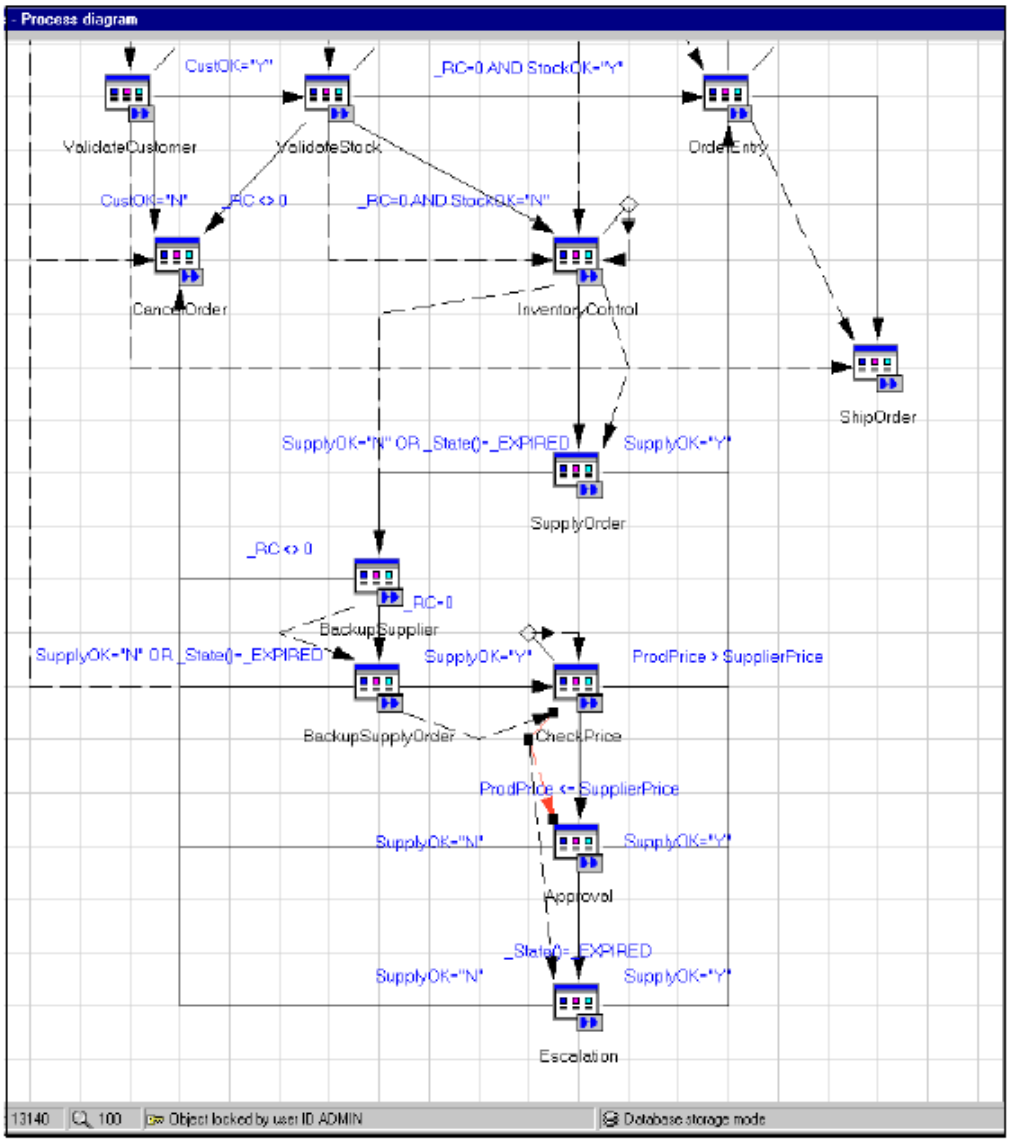

图9-12 修改定价异常流程示图

这样，我们就完成了价格检查和“ approval/escalation ”活动的模拟练习。

9.3 创建BuyXYZ_Dummy消息流

本节将描述“ BuyXYZ_Dummy ”消息流。此消息流可能是最简单的消息流。它只包含一个“ MQInput ”节点，该节点从“ MQSI.DUMMY ”队列接收一条消息并将其丢弃。我们需要执行该哑元消息流以便清空与“ CheckPrice ”活动相关的队列。

图 9-13 显示了此简单的消息流。

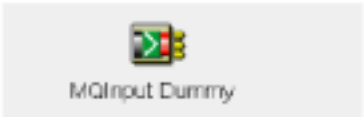

图9-13 消息流概览

在“ MQInput ”节点的“ Basic ”标签中，我们将指定 MQSeries Workflow 把 UPES 消息写入“ MQSI.DUMMY ”队列。

在“ MQInput ”节点的“ Default ”标签中，我们将指定 BLOB 消息域。

在“ MQInput ”节点的“ Advanced ”标签中，我们将把“ Transaction Mode ”设为“ No ”，如图 9-14 所示。

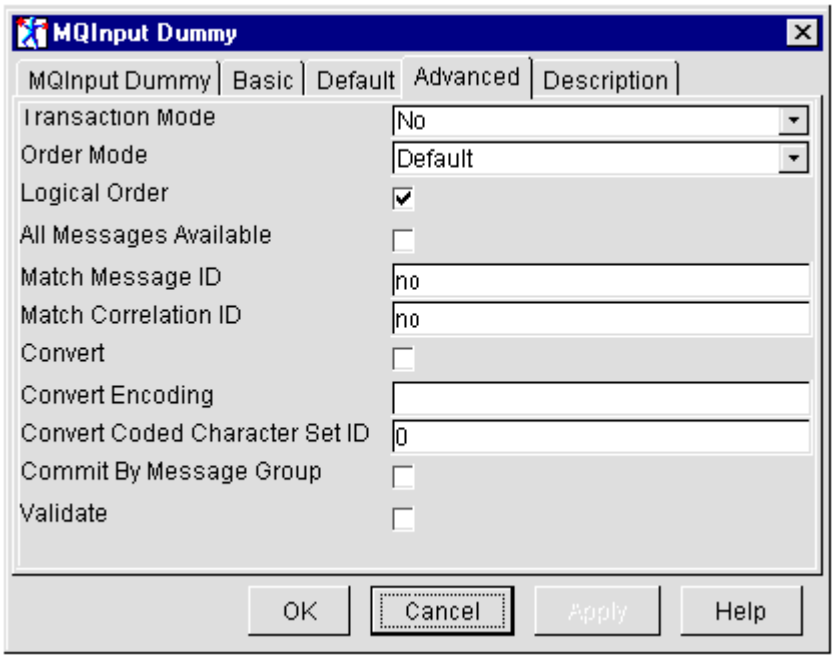

图9-14“MQInput”节点细节

9.4 为审批活动创建JSP

使用如 4.3.2 节“ JSP 快速应用向导 ”（159 页）中所述的方法为审批 Web 窗体创建和应用一个 JSP。

在构建时环境中，更新“ approval ”和“ escalation ”活动的活动定义以便使用最新创建的 JSP。

导出 FDL 并将新的流程定义导入正常运行时环境。

9.5 测试新的流程和条件

我们现在将使用如第 153 页的第 4.2.3 节“ 使用 Web 客户机和默认 Web 页测试消息流 ”中所述的方法来测试新流程。

我们现在将必须作为“ BUYXYZ_CLERK ”登录来执行取消订单和确订单活动。作为“ BUYXYZ_VP ”登录来执行“ InventoryControl ”和“ Approval ”活动。作为“ BUYXYZ_SVP ”登录来执行“ Escalation ”活动。

9.6 主供应商提高价格问题

现在有了异常流程，我们可以很容易地修改业务流程来对主供应商执行价格检查。如图 9-15 所示，简单地在供应订单和“ CheckPrice ”活动之间添加几个连接器，删除从“ SupplyOrder ”活动到订单条目活动的连接器，我们就可以对所有供应商进行异常管理了。

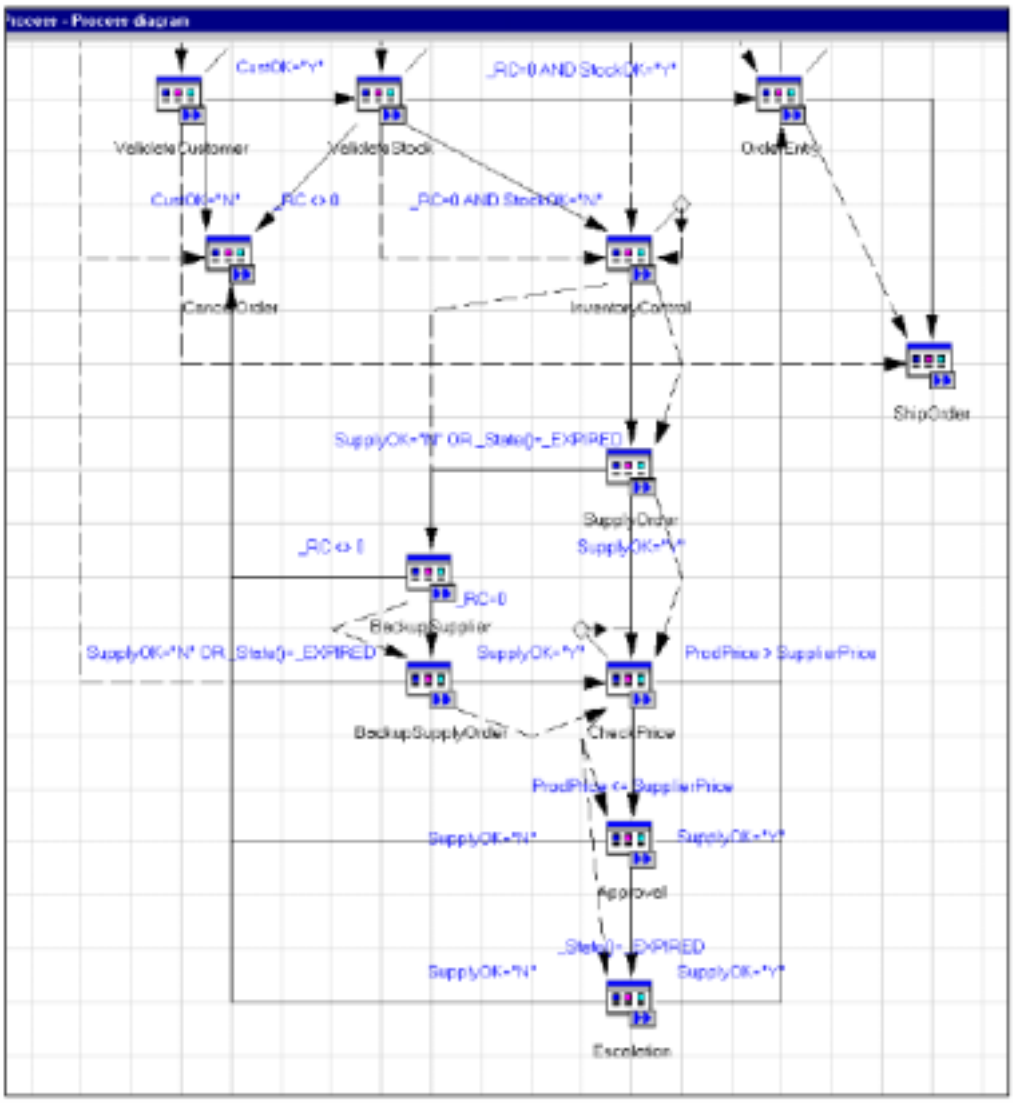

图9-15　执行主供应商价格检查

9.7 下一个挑战

由于我们已经以这样一种方式构建了业务流程模型，即我们可以很容易地使用不只一个供应商来提供特定的产品，所以如果我们要在特定的时刻选择最好的供应商，问题就又出现了。基于供应商最近活动的历史记录，我们可以在正常运行时中动态地选择主供应商。

在本书第 357 页的第十章“ 挖掘审计信息 ”中，我们将会看到这是如何达到的。

A

硬件和软件配置

本附录将给出设备配置的细节。

包括以下部分：

- 硬件细节
- 软件细节

硬件配置

该配置中用于 WebSphere 服务器、MQSeries Workflow 服务器和 MQSeries Integrator 服务器的每个设备都是一台 IBM PC 300PL，型号 6565。这些系统均拥有一个以 667MHz 运行的 Pentium III 处理器，512 MB 内存。它们通过 16Mbps 令牌环卡联接网络，并配有 20.4GB EIDE 硬盘。

用作数据库服务器的设备是 IBM Netfinity 3000。该系统拥有一个以 350MHz 运行的 Pentium II 处理器和 256 MB 内存。它通过 16 Mbps 令牌环卡连接网络，并配有 SCSI 硬盘。

软件环境

以下部分将逐条描述本书中所有设备的软件配置。

操作系统

所有设备都运行 Windows NT Server 版本 4.0（Build 1381: Service Pack 6a）。

产品软件

本书中描述的每台设备所安装的全部软件环境如下：

Web 服务器

- M23BZZYP：
  - IBM WebSphere 应用服务器版本 3.5.3
  - Windows NT 版本 5.2 的 IBM MQSeries+支持包 MA88
  - IBM Java 开发工具包版本 1.2.2
  - 具有 FixPak 1 的 Windows NT 版本 7.1 的 IBM DB2
  - IBM MQSeries Workflow 版本 3.3

工作流服务器 1

- M23CABYG：
  - Windows NT 版本 5.2 的 IBM MQSeries +支持包 MA88
  - 具有 FixPak 1 的 Windows NT 版本 7.1 的 IBM DB2

- IBM MQSeries Workflow 版本 3.3

工作流服务器 2

- M23CAAAD：
  - Windows NT 版本 5.2 的 IBM MQSeries +支持包 MA88
  - 具有 FixPak 1 的 Windows NT 版本 7.1 的 IBM DB2
  - IBM MQSeries Workflow 版本 3.3

数据库服务器

- M23M1773：
  - Windows NT 版本 5.2 的 IBM MQSeries +支持包 MA88
  - 具有 FixPak 1 的 Windows NT 版本 7.1 的 IBM DB2
  - Windows NT 版本 2.01 的 IBM MQSeries Integrator + CSD 1（U200132）
  - IBM MQSeries Workflow 版本 3.3

MQSI 服务器 1

- M23CABWZ：
  - Windows NT 版本 5.2 的 IBM MQSeries +支持包 MA88
  - 具有 FixPak 1 的 Windows NT 版本 7.1 的 IBM DB2
  - Windows NT 版本 2.01 的 IBM MQSeries Integrator + CSD 1（U200132）
  - IBM MQSeries 适配器内核版本 1.1.1

MQSI 服务器 2

- M23CAAXY：
  - Windows NT 版本 5.2 的 IBM MQSeries +支持包 MA88
  - 具有 FixPak 1 的 Windows NT 版本 7.1 的 IBM DB2
  - Windows NT 版本 2.01 的 IBM MQSeries Integrator + CSD 1（U200132）
  - IBM MQSeries 适配器内核版本 1.1.1

B

应用程序安装实例

本附录将总结本红皮书中重建已有解决方案所需的全部定义、程序和数据文件。

第三章中的定义、命令文件和数据文件

在表 11-1 中，我们总结了第三章“ 技术组件配置 ”（第 17 页）中重建已有解决方案所需的所有定义、命令和数据文件。

表 11-1 第三章解决方案文件列表

文件名	描述
create_dbsqm.cmd	在 M23M1773 设备上创建 DBSQM 队列管理器。
_dbsqm.cfg	在 M23M1773 设备上的 DBSQM 队列管理器对象配置文件，create_dbsqm.cmd 执行时调用该文件。
create_mqsi01qm.cmd	在 M23CABWZ 设备上创建 MQSI01QM 队列管理器。
_mqsi01qm.cfg	在 M23CABWZ 设备上的 MQSI01QM 队列管理器对象配置文件，create_mqsi01qm.cmd 执行时调用该文件。
create_mqsi02qm.cmd	在 M23CAAXY 设备上创建 MQSI02QM 队列管理器。
_mqsi02qm.cfg	在 M23CAAXY 设备上的 MQSI02QM 队列管理器配置文件，create_mqsi02qm.cmd 执行时调用该文件。
create_mqsiCFGmgr_db.cmd	在 M23M1773 设备上创建 MQSICMDB 配置管理器数据库。
create_mqsi01bk_db.cmd	在 M23CABWZ 设备上创建 MQSI01BK 代理数据库。
create_mqsi02bk_db.cmd	在 M23CAAXY 设备上创建 MQSI02BK 代理数据库。
create_mqsiCFGmgr.cmd	在 M23M1773 设备上创建配置管理器。
create_usernameserver.cmd	在 M23M1773 设备上创建用户名服务器。
create_mqsi01bk.cmd	在 M23CABWZ 设备上创建 MQSI01BK 代理。
create_mqsi02bk.cmd	在 M23CAAXY 设备上创建 MQSI02BK 代理。

第四章中的定义

在表 11-2 中，我们总结了第四章“ 在 MQSeries Workflow 中实现模型 ”（第 109 页）中重建已有解决方案所需的所有定义。

表 11-2 第四章解决方案文件

文件名	描述
buyxyz-04dummy.fdl	导入工作流构建时或/和第四章运行时数据库的 FDL 文件。

第五章中的定义、命令文件和数据文件

在表 11-3 中，我们总结了第五章“ 在 MQSeries Integrator 中执行活动 ”（第 179 页）中重建已有解决方案所需的所有定义、命令文件和数据文件。

表 11-3 第五章解决方案文件列表

文件名	描述
Buyxyz_msgflows.xml	包含我们在本红皮书中创建的所有消息流的文件。这些消息流是： 1. BuyXYZ_Validate_Customer 2. BuyXYZ_Validate_Stock 3. BuyXYZ_Order_Entry_CICS 4. BuyXYZ_Order_Entry_CICSACK 5. BuyXYZ_Supply_Order_PO 6. BuyXYZ_Supply_Order_POACK 7. BuyXYZ_Backup_Supplier 8. BuyXYZ_Store_Audit_Trail 9. BuyXYZ_Dummy 需要通过 MQSI 控制中心将其导入配置管理器数据库。
Buyxyz_msgset.mrp	包含我们在本红皮书中创建的消息设置的文件。它由三个 MRM 消息组成。这些 MRM 信息是： Supply_Order_PO Supply_Order_POACK Order_Entry 需要通过 mqs imrmipexp 命令将其导入配置管理器数据库。
buyxyz_ddl.cmd	在 M23BZZYP 设备上 CUSTOMER 数据库中用于创建表的 DB2 脚本。

文件名	描述
buyxyz_data.cmd	在 M23BZZYP 设备上 CUSTOMER 数据库中用于移植表的 DB2 脚本。
buyxyz_wasqm.cfg	在 M23BZZYP 设备上用于改变 WASQM 队列管理器队列的 MQSeries 脚本。
buyxyz_wf01qm.cfg	在 M23CABYG 设备上用于改变 WF01QM 队列管理器的 MQSeries 脚本。
buyxyz_wf02qm.cfg	在 M23CAAAD 设备上用于改变 WF02QM 队列管理器队列的 MQSeries 脚本。
buyxyz_mqsi01qm.cfg	M23CABWZ 设备上的 MQSI01QM 队列管理器中用于创建 MQ 对象的 MQSeries 脚本。
buyxyz_mqsi02qm.cfg	M23CAAXY 设备上的 MQSI02QM 队列管理器中用于创建 MQ 对象的 MQSeries 脚本。
CustomerInput_2_MQSI.xml	从 MQWF ValidateCustomer 活动到 MQSI BuyXYZ_Validate_Customer 消息流的 XML 信息。
CustomerValid_2_MQWF.xml	从 MQSI BuyXYZ_Validate_Customer 消息流到 MQWF ValidateCustomer 活动的 XML 信息。
StockInput_2_MQSI.xml	从 MQWF ValidateStock 活动到 MQSI BuyXYZ_Validate_Stock 消息流的 XML 信息。
StockValid_2_MQWF.xml	从 MQSI BuyXYZ_Validate_ Stock 消息流到 MQWF ValidateStock 活动的 XML 信息。
OrderInfo_2_MQSI.xml	从 MQWF OrderEntry 活动到 BuyXYZ_Order_Entry_CICS 消息流的 XML 信息。
OrderInfo_2_MQWF.xml	从 MQSI BuyXYZ_Order_Entry_CICSACK 消息流到 MQWF OrderEntry 活动的 XML 信息。

第六章中的定义、程序和数据文件

在表 11-4 中，我们总结了第六章“ 使用 MQSeries Adapter Offering 创建 BOD 消息 ”（第 257 页）中重建已有解决方案所需的所有定义、程序和数据文件。

表 11-4 第六章解决方案文件列表

文件名	描述
mqsi_c_bindings	使用 MQSeries Integrator 生成 C 标题文件目录
IntraBPM_ch6.zip	已导出的 MQSeries 适配器生成器工作空间
mqbjget.java 和 mqbjget.class	MQSeries Adapter Offering 中用于导入消息集的 Java 程序
mqbjput.java 和 mqbjput.class	MQSeries Adapter Offering 中用于导入消息类型的 Java 程序
Simple_Process_POACK_TA	本目录包含用于目标适配器的生成类
Simple_Process_PO_SA	本目录包含用于源适配器的生成类
aqmsetup	MQSeries Adapter Offering 配置文件
aqmconfig.xml	MQSeries Adapter Offering 配置文件
parseopt.c	供应商模拟器 parseopt.c 文件
parseopt.h	供应商模拟器 parseopt.h 文件
readme.txt	供应商模拟器 readme.txt 文件
respond.c	供应商模拟器 respond.c 文件
respond.exe	供应商模拟器 respond.exe 文件

程序 respond.exe 用于简化 MQSeries Adapter Offering 适配器的使用和企业对企业通信的商业伙伴协议管理器的使用。

第七章中的定义和数据文件

在表 11-5 中，我们总结了第七章“ 在工作流中调用企业 JavaBean ”（第 301 页）中重建已有解决方案所需的所有定义和数据文件。

表 11-5 第七章解决方案文件列表

文件名	描述
buyxyz-07upes.fdl	用于导入第七章工作流构建时或/和运行时数据库的 FDL 文件。
Shipping_2_EJB.xml	从 MQWF ShipOrder 活动到 Shipping EJB 的 XML 消息。
ShippingEJB.zip	用于 ShippingEJB 的 Java 文件。

第八章中的定义和数据文件

在表 11-6 中，我们总结了第八章“ 扩展模型：处理商业异常 ”（第 327 页）中重建已有解决方案所需的所有定义和数据文件。

表 11-6 第八章解决方案文件列表

文件名	描述
buyxyz-08backup.fdl	用于导入第八章工作流构建时或/和运行时数据库的 FDL 文件。
SupplyInput_BackupSupplier_2_MQSI.xml	从 MQWF BackupSupplier 活动到 MQSI BuyXYZ_Backup_Supplier 消息流的 XML 消息。
SupplyInput_BackupSupplier_2_MQWF.xml	从 MQSI BuyXYZ_Validate_Customer 消息流到 MQWF ValidateCustomer 活动的 XML 消息。

第九章中的定义

在表 11-7 中，我们总结了第九章“ 扩展模型：实现基于角色的交互 ”（第 337 页）中重建已有解决方案所需的所有定义。

表 11-7 第九章解决方案文件列表

文件名	描述
buyxyz-09price.fdl	用于导入第九章工作流构建时或/和运行时间数据库的 FDL 文件

第十章中的定义、命令行和数据文件

在表 11-8 中，我们总结了第十章“ 挖掘审计信息 ”（第 357 页）中重建已有解决方案所需的所有定义、命令行和数据文件。

表 11-8 第十章解决方案文件列表

文件名	描述
buyxyz-10audit.fdl	用于导入第十章工作流构建时或/和运行时数据库的 FDL 文件。
AuditSample.xml	从 MQWF 审计工具到 MQSI BuyXYZ_Store_Audit_Trail 消息流的 XML 消息。
buyxyz_audit_ddl.cmd	在 CUSTOMER 数据库中创建 AUDIT_TRAIL 表的 DB2 脚本。
byxyz_audit_view.cmd	在 CUSTOMER 数据库 AUDIT_TRAIL 表中创建视图的 DB2 脚本。
BuyxyzAudit.html	用于审计跟踪报告的实例 HTML 页。
Audit_Results.jsp	用于审计跟踪报告的实例 JSP 页。

C

其他资料

本红皮书中提及的其他资料可按如下方法从互联网下载。

定位Web资料

与本红皮书相关的 Web 资料在互联网上以软件拷贝形式存在于 IBM 红皮书 Web 服务器中。将您的 Web 浏览器指向：

ftp://www.redbooks.ibm.com/redboks/SG246173

或者，您可以访问 IBM 红皮书网站：

ibm.com/redbooks

选择其他资料（Additional materials）并打开与本红皮书编号 SG246173 相应的目录。

使用Web资料

与本红皮书相关的其他 Web 资料包括下列文件：

文件名	*描述*
SG246173.zip	每章的压缩文件

下载和使用Web资料的系统需求

该解决方案安装在总共六台设备上，这些设备的规范已在附录 A“ 硬件和软件配置 ”（第 393 页）中给出。我们也可在单独的设备上重新创建解决方案，该设备仅配有一个 MQSeries Workflow 服务器和一个 MQSeries Integrator 代理。

推荐配置如下：

硬盘空间	2 GB 用于软件和配置
操作系统	Windows NT 版本 4+ServicePac 6a
处理器	Pentium III 处理器
内存	512 MB

如何使用Web资料

在您的工作站上创建子目录（文件夹），把 Web 资料压缩文件的内容解压缩到该文件夹内，这将为各章创建其子目录。请参考第 397 页附录 B “ 实例应用程序安装 ”来理解每个文件的使用。

相关出版物

本节所列出版物有助于了解本红皮书所包含主题的更多详细信息。

IBM红皮书

有关订购这些出版物的详细信息，请参见第 408 页的 “ 如何获取 IBM 红皮书 ”。

- 《WebSphere 版本 3.5 手册》编号：SG24-6161
- 《使用 WebSphere Studio 和 VisualAge for Java 编写服务件和 JSP》编号：SG24-5755
- 《VisualAge 的 *EJB* 开发用于 WebSphere 应用服务器的 Java》编号：SG24-6144
- 《设计和实现 IBM WebSphere 应用服务器的服务件、JSP 和 EJBs》编号：SG24-5754
- 《Windows NT 的 MQSeries Workflow 入门》编号：SG24-5848
- 《WebSphere 可测试性：使用 WebSphere 应用服务器高级版的 WLM 和群集》编号：SG24-6153
- 《使用 DB2 UDB EEE 管理 VLDB》编号：SG24-5103

其他资料

- 《多平台 MQSeries 适配器内核：快速入门》编号：GC34-5855
- 《使用 Java》编号：SC34-5657
- 《MQSeries Integrator 2.0.1 安装指南》编号：GC34-5600
- 《快速掌握 MQSeries Workflow3.3 构建时》编号：SH12-6286
- 《MQSeries Workflow3.3 管理指南》编号：SH12-6289
- 《MQSeries Workflow 3.3 编程指南》编号：SH12-6291

全部 MQSeries 家族产品的使用手册，可从以下网址下载：

http://www-4.ibm.com/software/ts/mqseries/library/manualsa/

供参考的Web站点

如欲进一步了解信息来源，以下 Web 站点也是相关的：

- MQSeries 手册

  http://www-4.ibm.com/software/ts/mqseries/library/manualsa/
- MQSeries 支持包

  http://www-4.ibm.com/software/ts/mqseries/txppacs/txpm1.html
- MQSeries 系列主页

  http://www-4.ibm.com/software/ts/mqseries/
- 最新 Windows 安装软件

  http://www.microsoft.com/downloads/release.asp?ReleaseID=17344
- SOAP 规范

  http://www.w3.org/TR/SOAP
- WSDL 规范

http://www.w3.org/TR/wsdl

- UDDI 网站

  http://uddi.org
- W3C 协会网站

  http://www.w3.org/
- OAG 网站

  http://www.openapplications.org/
- OASIS 网站

  http://www.oasis-open.org/

如何获取IBM红皮书

如欲获得其他红皮书和红皮稿，请从以下红皮书的 Web 站点查看、下载或定购硬拷贝：ibm.com/redbooks。

还可以从该红皮书站点下载补充资料（代码实例或磁盘／光盘图像）。

红皮稿是正在编撰中的红皮书；不是所有的红皮书都能成为红皮稿，有时只以这种方式发布几个章节。目的是以比正式发布过程更快的速度发布信息。

IBM红皮书集

红皮书也可在光盘上使用。单击红皮书 Web 站点上的光盘版按钮即可获得所有光盘版本提供的信息，以及更新和格式信息。

特别声明

在本出版物中所提到的 IBM 产品、程序或服务并不意味着 IBM 将为所有 IBM 运作的国家提供。任何对 IBM 产品、程序或服务的引用并不说明或暗示只能使用 IBM 的产品、程序或服务。凡是同等功能的产品、程序或服务，只要不侵犯 IBM 的知识产权，都可以用来代替 IBM 产品、程序或服务。

本文档中的信息和特定设备的使用介绍结合在一起，但仅限于使用那些特定硬件和软件产品和标准的应用程序。

IBM 可能已经申请或正在申请与本文档有关的各项专利权。提供本文档并不表示允许您使用这些专利。您可以用书面方式将特许查询寄往： IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785。

为了以下目的：(i) 允许在独立创建的程序和其他的程序（包括本程序）之间进行信息交换和 (ii) 允许对已经交换的信息进行相互使用，而希望获取本程序有关信息的合法用户请与下列地址联系： IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA 。

这些信息可以通过适当的条款和条件来得到，包括在一些案例中或支付一定的费用。

任何这些技术的执行是客户的责任，并依靠客户的能力来评估它们，将其整合到客户的操作环境中。如果其中的条款由 IBM 在特定情形下被正确地校订，并不能保证在别处也可以得到相同或相似的结果。如果客户企图修改在他们自己环境下的这些技术，将自己承担风险。

本文档包含的信息并没有提交给任何正式的 IBM 测试并发布。本信息的使用或任何这些技术的执行是客户的责任，并依靠客户的能力来评估它们，将其整合到客户的操作环境中。如果其中的条款由 IBM 在特定情形下被正确地校订，并不能保证在别处也可以得到相同或相似的结果。如果客户企图修改在他们自己环境下的这些技术，将自己承担风险。

本资料中对非 IBM Web 站点的任何引用只是为方便您而提供的，IBM 不以任何方式对这些 Web 站点作保证。

以下术语是其他公司的商标：

Tivoli、Manage. Anything. Anywhere.、The Power To Manage., Anything. Anywhere.、TME、NetView、Cross-Site、Tivoli Ready、Tivoli Certified、Planet Tivoli 和 Tivoli Enterprise 是 Tivoli Systems Inc.（IBM 子公司）在美国和/或其他国家（或地区）的商标或注册商标。在丹麦，Tivoli 是经 Kjøbenhavns Sommer – Tivoli A/S 许可的商标。

C-bus 是 Corollary 公司在美国和/或其他国家的商标。

Java 及所有基于 Java 的商标及徽标均为 Sun 微系统公司在美国和/或其他国家的商标或注册商标

Microsoft、Windows、Windows NT、Windows 商标均为微软公司在美国和/或其他国家的商标。

PC Direct 是 Ziff 通信公司在美国和/或其他国家的商标，并且许可 IBM 公司使用。

ActionMedia、LANDesk、MMX、Pentium 和 ProShare 是 Intel 在美国和/或其他国家的商标。

UNIX 是在美国及其他通过 Open Group 唯一特许的国家的注册商标。

SET、SET Secure Electronic Transaction 和 SET 标志是 SET Secure 电子交易公司的商标。

其他的公司、产品和服务名称可能是其自己的商标或服务标志。

词汇表

活动 组成过程模型的步骤之一。活动可以是程序活动、过程活动或块活动。

适配器 MQSeries 适配器生成器的输出。基本上，用户将针对发给或收自应用程序的一个消息类型来创建每个适配器。因此，适配器本身并不是MQSeries 适配器生成器的一部分。适配器由编译为共享库的C或Java源代码构成。

管理服务器 在MQSeries Workflow系统中执行管理功能的MQSeries Workflow组件。这包括终止和启动系统、执行错误管理和参与管理系统组功能。

API 参看**应用编程接口**。

应用编程接口 由软件产品提供的可使程序能够请求服务的接口。

B2B 参看**企业对企业**。

块活动 由可以与控制和数据连接器连接的一组活动组成的复合活动。

BOD 参看**商业对象文档**。

代理域 共用一个单独配置管理器的一组代理。

代理 参看**消息代理**。

商业对象文档 在组织内部或组织之间流动的标准业务流程的表示法。例如添加购买订单、显示产品可用性和添加销售订单。BOD由OAG使用 XML 定义。

企业对企业 互联网应用模型。其中企业通过互联网彼此互联以执行商业操作。

Cleanup 服务器 从运行时数据库中物理删除信息的 MQSeries Workflow 组件。

群集队列管理器 队列管理器可以是一个群集的成员，也可以是多个群集的成员。

群集队列 由群集队列管理器承载的队列，可由群集中的其他队列管理器使用。

群集队列 由群集队列管理器承载的队列，可由群集中的其他队列管理器使用。

群集队列 由群集队列管理器承载的队列，可由群集中的其他队列管理器使用。

群集 逻辑相关的队列管理器网络。

配置管理器 MQSeries Integrator的组件，充当配置存储库和一组正在执行的代理的之间的接口。

配置存储库 代理和拓朴配置的永久存储。

构建时 一个带有图形用户接口的MQSeries Workflow组件，用于创建和运行工作流模型、管理资源和网络定义。

连接器（MQSeries Integrator） 将一个消息处理节点的输出终端连接到另一个节点的输入终端的实体。

控制连接器（MQSeries Workflow）定义过程中两个节点之间的可能控制流。实际过程由在运行时中基于与控制连接器相关联的转换条件的数值（真或假）来决定。

数据连接器（MQSeries Workflow）定义容器之间的数据流程。

数据容器 存储活动或过程的输入和输出数据。

文档类型定义 为特殊 XML 文档类指定结构的规则。DTD 定义了带有元素、属性和符号的结构。并且它建立了每个元素、属性和符号怎样用于特殊文档类的约束。

域（MQSeries Workflow） 含有相同元模型并共用相同的职员信息和拓朴信息的 MQSeries Workflow 系统组。

DTD 参看**文档类型定义**。

动态职员分配 通过指定例如角色、组织或级别等标准来分配职员的一种方法。当活动就绪时，面临选择标准的用户将接受该活动继续工作。

EJB 参看**企业 JavaBean**。

企业应用集成 IT 行业概念。其中不同的 IT 系统、平台和应用彼此相连以提供集成化的应用操作。

企业 JavaBean 企业 JavaBean 定义了组件模型。该模型提供了基于多层分布式体系结构的 Java 应用程序的开发、应用和执行。它扩展了 JavaBean 组件模型以支持服务器组件，包括服务器功能例如全局命名、分布式处理、持久性和安全性。EJB 特别设计从而使创建健壮的和可扩展的服务器组件更加容易，并且把对系统编程的关注与对业务编程的关注分离开来。EJB 即定义了临时对象（会话 bean）也定义了永久对象（实体 bean）。

实体 bean 永久对象。永久对象从内存中创建、应用、载入或卸载。实体 bean 支持多用户访问和参与事务处理。

执行服务器 在运行时中执行过程实例处理的 MQSeries Workflow 组件。

FDL MQSeries Workflow 文档定义语言。该语言用于在组件之间交换 MQSeries Workflow 信息。它可用于导入和导出函数。

Java 数据库连通性 与 ODBC 具有相同特性

	但，专门设计为由 Java 数据库应用程序使用的应用编程接口。 **Java 开发工具包** 用于编写、编译、调和运行 Java 小程序和应用程序的软件包。 **Java 消息服务** 提供 Java 语言函数用于处理消息的应用编程接口。
Java 命名和目录接口 JNDI 为用 Java 编写的应用程序提供命名和目录功能。这些应用程序可以使用 JNDI API 来访问透明插入的多种命名和目录服务。 **Java 运行时环境** Java 工具包的子集。它允许您运行 Java 应用程序和 Java 小程序。 **JavaBeans** 可重用 Java 组件。使用 JavaBeans 软件引擎可以在图形拖放开发环境中操作和装配 bean。bean 接口揭示了 bean 的事件、属性和方法并使其可由其他程 bean 使用。JavaBeans 提供了对自省、定制和永久性的支持。 **JDBC** 参看 **Java 数据库连通性**。 **JDK** 参看 **Java 开发工具包**。 **JMS** 参看 **Java 消息服务**。 **JNDI 定位器** 使用 JNDI 作为命名服务来注册 Java CORBA 代理，并使用 RMI-IIOP 作为 API 方法调用的传输协议的定位政策。 **JNDI** 参看 **Java 命名和目录接口**。	**消息处理节点** 在消息流中表示明确定义活动的停止点。 **消息队列** 软件组件之间使用异步消息通信的通信技术。 **微流程** 当消息经过一个适配器从其输入流入其输出时，模拟消息处理的示意图。 **节点（MQSeries Adapter Offering）** 微流程中表示任一节点的通用词汇。微流节点的每种类型代表一个明确定义的处理阶段。一组微流程节点由 builder 提供。 **节点（MQSeries Integrator）** 参看**消息处理节点**。 **节点（MQSeries Workflow）** 流程图表中活动的通用名称。也是承载 MQSeries Workflow 系统的操作系统映像。 **OAG** 参看**开放应用程序组**。 **ODBC** 参看**开放式数据库连接性**。

JRE 参看 **Java 运行时环境**。

消息代理 承载一个或多个消息流的一组执行过程。

消息域 消息定义源。

消息流 当消息或事件通过代理时，表示其中一组活动的示意图。消息流由多个消息处理节点组成。

OMG 对象管理组，促进在软件中使用面向对象技术的非赢利性组织。

开放应用程序组 关注最佳练习和基于过程的 XML 电子商务内容和应用集成的非赢利性组织。

开放式数据库连接性 在关系型和非关系型数据库管理系统中访问数据的标准应用编程接口。使用该 API，数据库应用程序可以访问存储在多种计算机中的数据，甚至每一数据库管理系统使用不同数据存储格式和编程接口。ODBC 基于 X/Open SQL 访问组的调用等级接口(CLI)规范。

过程定义 参看**过程模型**。

过程执行代理 管理程序活动执行的 MQSeries Workflow 组件。例如 EXE 和 DLL 文件。

过程实例列表 一组根据用户定义标准选择和分类的过程实例。

过程实例 在运行时环境中正在执行的过程的实例。

过程模型 一组在过程模型中表示的过程。这

程序执行代理 当用户登录 MQSeries Workflow 运行时时正式启动的 MQSeries Workflow 组件。它的职责包括作为程序活动执行的可执行文件和 DLL 的执行。

队列管理器 为应用程序提供队列服务的子系统。它提供应用编程接口以使应用程序可以访问队列管理器拥有和管理的队列上的消息。

队列 MQSeries 对象。应用程序可以把消息加入队列，也可以从队列获得消息。队列由队列管理器拥有和管理。本地队列是一种可以包含等待处理的消息列表的队列类型。其他队列类型不能包含消息，而只能用于指向其他队列。

RMI 可以使您能够编写分布式 Java 程序的 Java API。该 API 允许从其他 Java 虚拟设备访问远程 Java 对象。

RMI-IIOP 客户机/服务器传输协议，用于在 Java 中编译带有 IIOP（CORBA）可靠性和跨

些过程在过程图表中以图形化形式表示。过程模型包含与过程活动相关的职员、程序和数据结构的定义。 **过程模板列表** 一组根据用户定义标准选择和分类的过程模板。 **过程模板** 来自已创建过程实例的过程模型的固定和转化格式。 **过程模板** 来自可以创建的过程实例的过程模型的固定格式。	语言互用性的易用 RMI 风格的方法调用。 **角色** 为职员定义的职责。角色是可以用于动态分配活动的标准之一。 **调度服务器** 基于时间事件（例如恢复暂停的工作项目或检测到期过程）调度活动的 MQSeries Workflow 组件。 **服务件** 由 Java 编程语言编写并在 Web 服务器执行的应用程序。在 Web 页的标记中将出现服务件引用，同样也会出现图形文件引用。Web 服务器将执行服务件并将执行结果(如果存在的话)发送给 Web 浏览器。
会话 bean 会话 bean 是临时对象，它代表过程或充当执行任务的代理。 **简单对象访问协议** 基于 XML 的协议，用于在分布式环境中交换信息。 **Sink** 代表过程或块活动输入容器的符号。 **SOAP** 参看**简单对象访问协议**。 **源** 代表过程或块活动的输出容器的符号。 **系统组** 共用同一运行时数据库的一组 MQSeries Workflow 系统。 **系统** 最小的带有 MQSeries Workflow 域的 MQSeries Workflow 单元。它由一组 MQSeries Workflow 服务器组成。	**工作流管理联盟（WfMC）**由工作流管理系统的供应商和用户组成的非赢利性组织。联盟的任务是促进工作流管理系统标准的制定以实现不同实施之间的互用性。 **工作流管理联盟** 由工作流管理系统的供应商和用户组成的非赢利性组织。联盟的任务是促进工作流标准的制定以允许不同系统间的互用性。 **工作流模型** 参看**过程模型**。 **工作流** 按照企业业务流程执行的行为列。 **工作列表视图** 用户根据过滤标准即工作列表的属性从一组工作项目中选择的工作项目列表和声明。它可以根据是否为该工作列表指定分类标准来分类。

转换　把过程模型转换成运行时过程模板的活动。 **UDDI**　参看**通用描述、发现和集成**。 **通用描述、发现和集成**　一组执行服务代理的 API。 **W3C**　万维网协会。为开发通用协议以促进万维网的发展和互用性而建立的国际性行业协会。 **Web 服务描述语言**　为服务提供者提供用于描述其服务接口方法的 XML 词汇。 **工作项目**　过程实例的活动上下文中所做工作的表示法。	**工作列表**　分配给用户并从工作流管理系统中检索的工作项目列表。 **WSDL**　参看 **Web 服务描述性语言**。 **XML**　可扩展标记语言，一种数据表示法标准。