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。

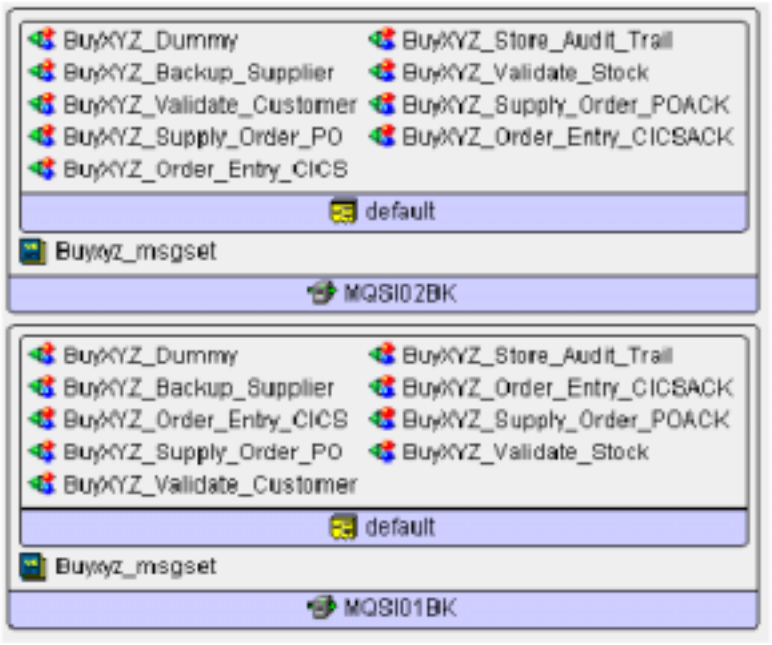

图 5-1 消息流代理分配

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。

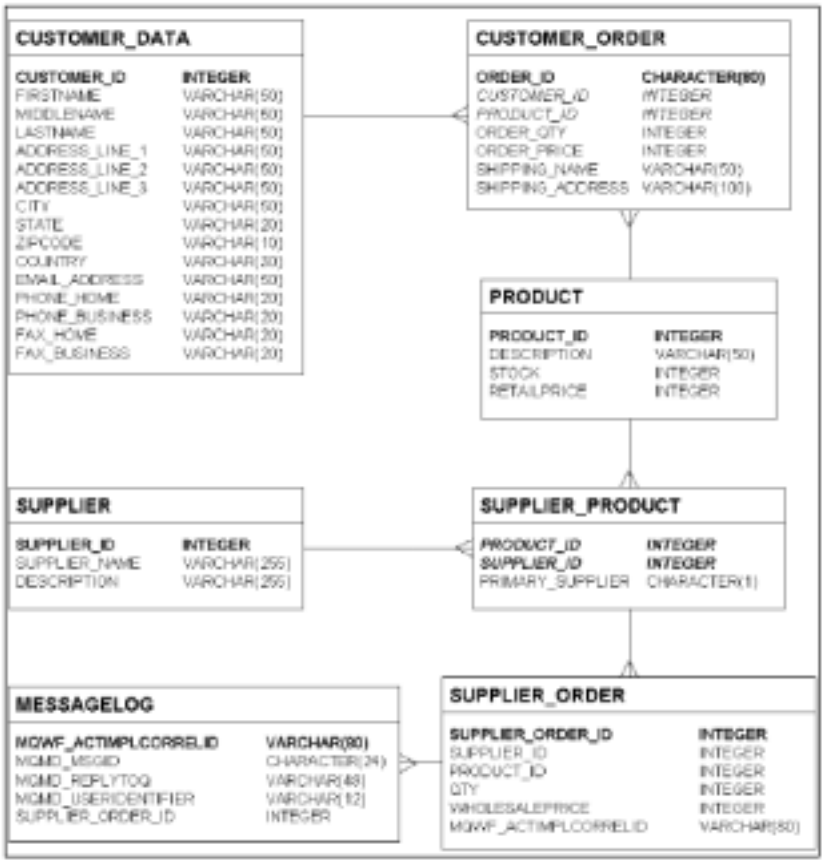

图5-2 逻辑数据库设计

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。

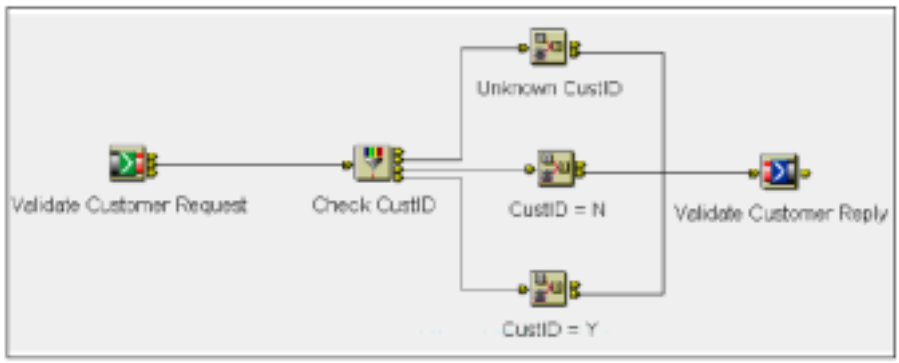

图5-3 确认客户信息流

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。

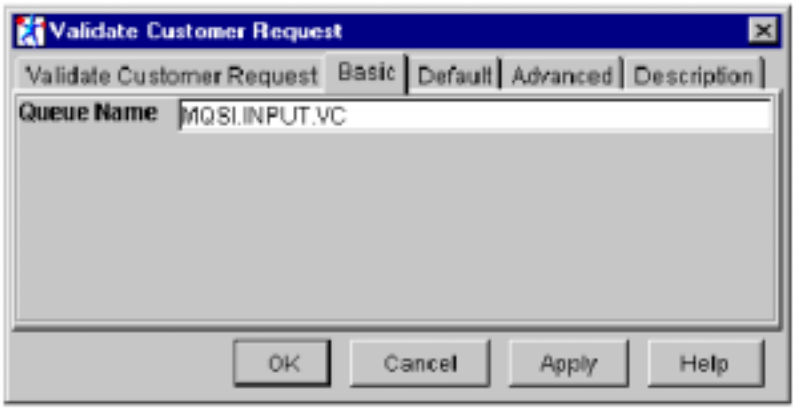

图 5-4 MQInput 节点细节

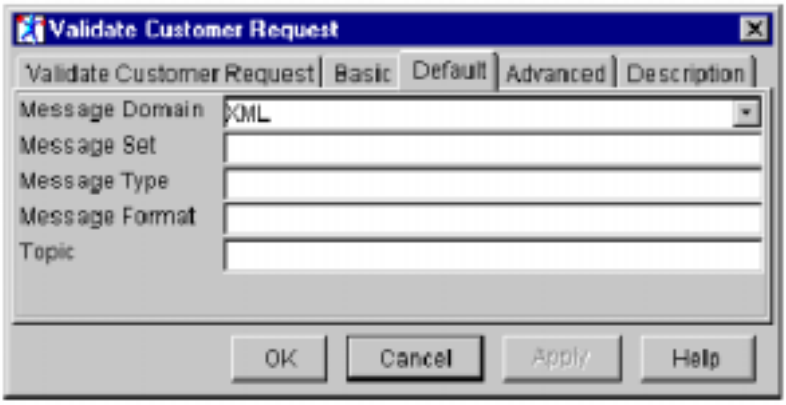

图 5-5 MQInput 节点细节

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。

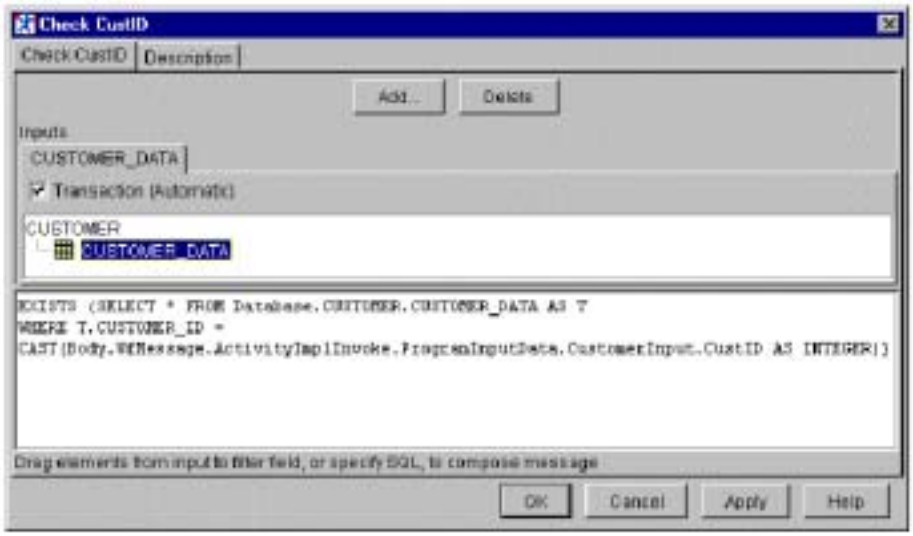

图 5-6 过滤节点细节

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。

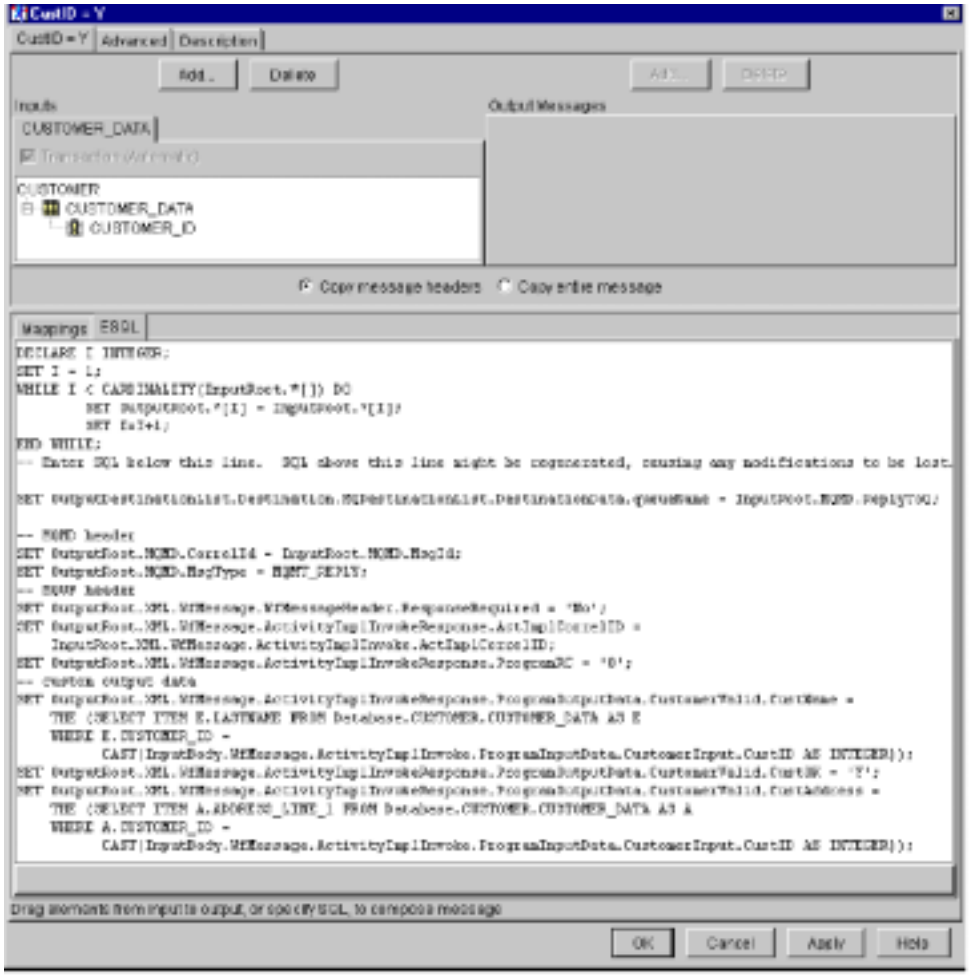
CustID = Y
CustID = Y
Advanced
Description
Add...
Delete
Inputs
Output Messages
CUSTOMER_DATA
Transaction (Automatic)
CUSTOMER
CUSTOMER_DATA
CUSTOMER_ID
Copy message headers
Copy entire message
Mappings
ESQL
DECLARE I INTEGER;
SET I = 1;
WHILE I < CARDINALITY(InputRoot.*[]) DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I=I+1;
END WHILE;
-- Enter SQL below this line. SQL above this line might be regenerated, causing any modifications to be lost.
-- MQMD header
SET OutputRoot.MQMD.CorrelId = InputRoot.MQMD.MsgId;
SET OutputRoot.MQMD.MsgType = MQMT_REPLY;
-- MQRF header
-- Custom output data
THE (SELECT ITEM E.LASTNAME FROM Database.CUSTOMER.CUSTOMER_DATA AS E
WHERE E.CUSTOMER_ID =
CAST(InputBody.MRMessage.ActivityImplInvoke.ProgramInputData.CustomerInput.CustID AS INTEGER));
THE (SELECT ITEM A.ADDRESS_LINE_1 FROM Database.CUSTOMER.CUSTOMER_DATA AS A
WHERE A.CUSTOMER_ID =
CAST(InputBody.MRMessage.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。

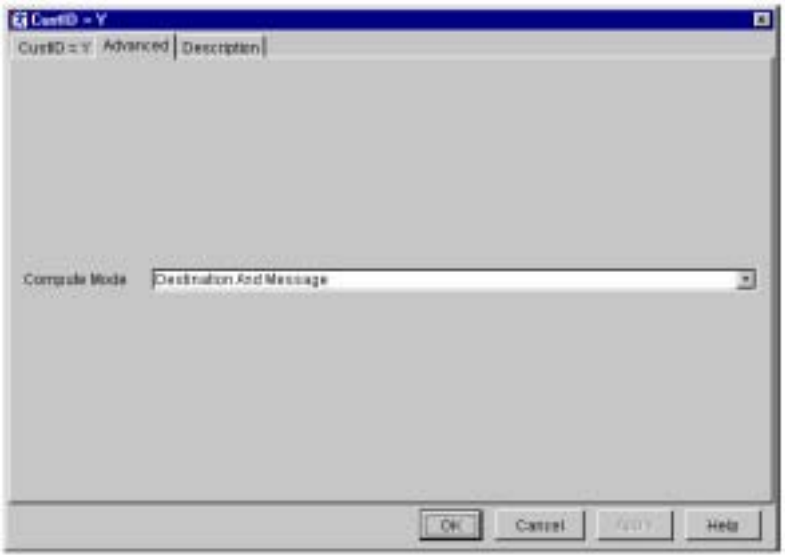

图 5-8 计算节点“CustID=Y”

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。

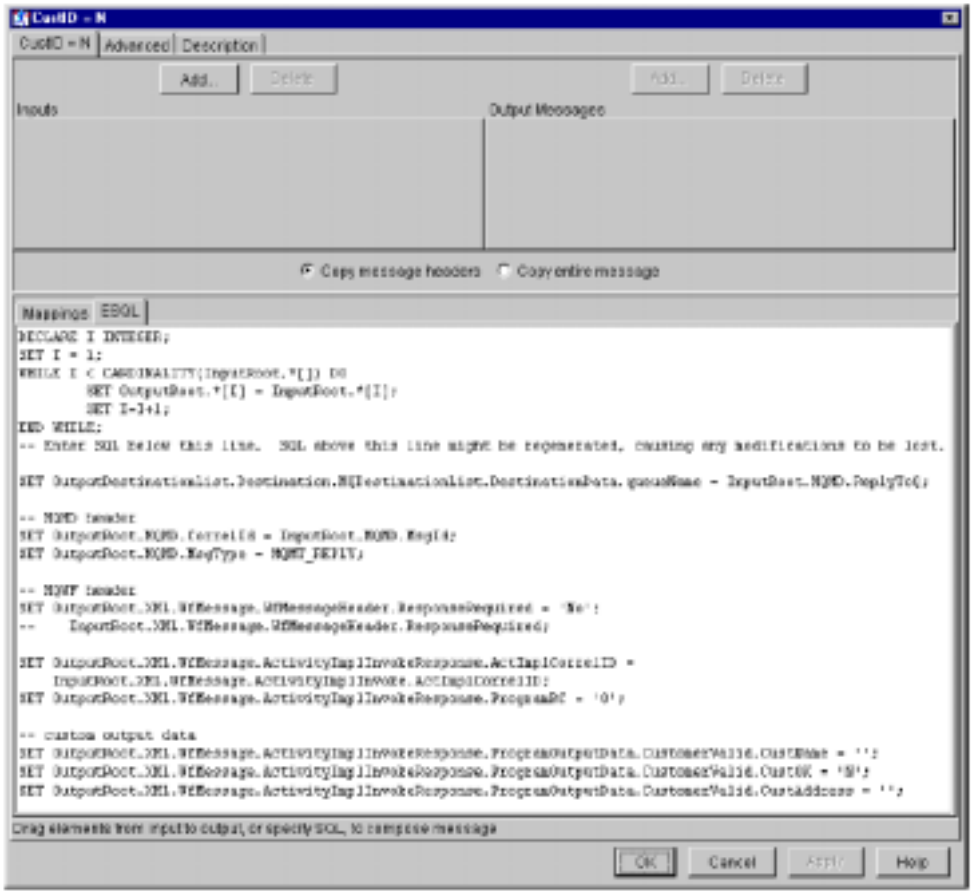

图 5-9 计算节点细节

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。

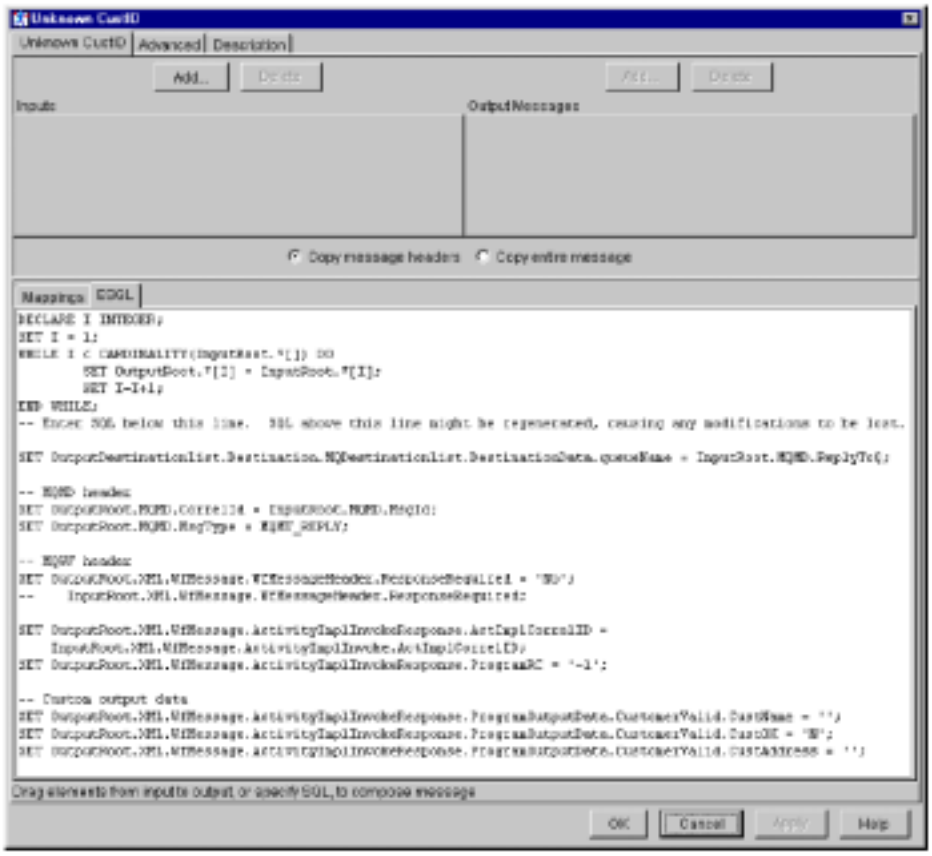

图5-10 计算节点细节

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。

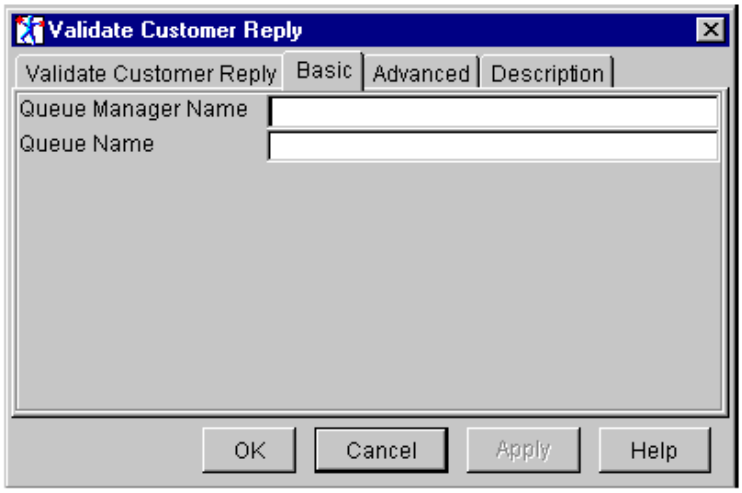

图 5-11 MQOutput 节点细节

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。

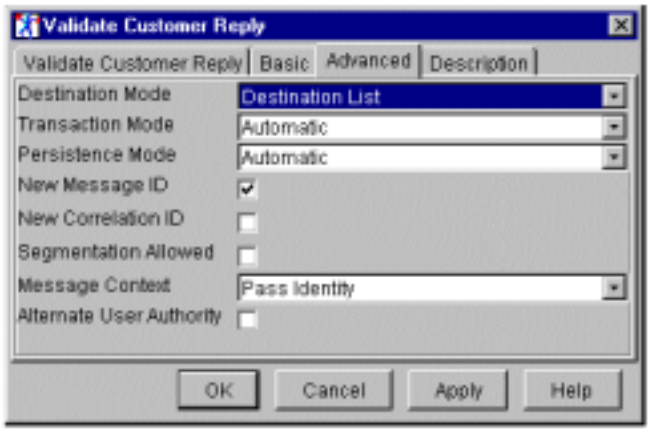

图 5-12　MQOutput 节点细节

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。

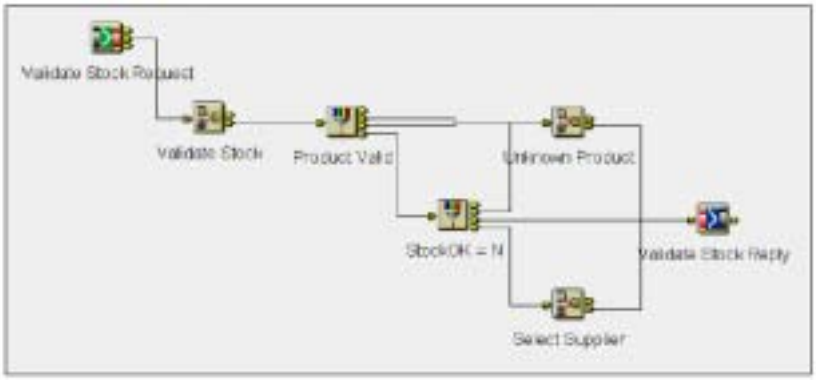

图5-13 确认库存消息流

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。

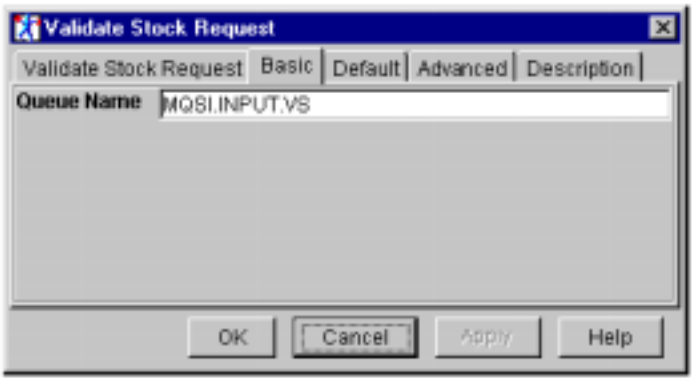

图 5-14 “MQInput ”节点细节

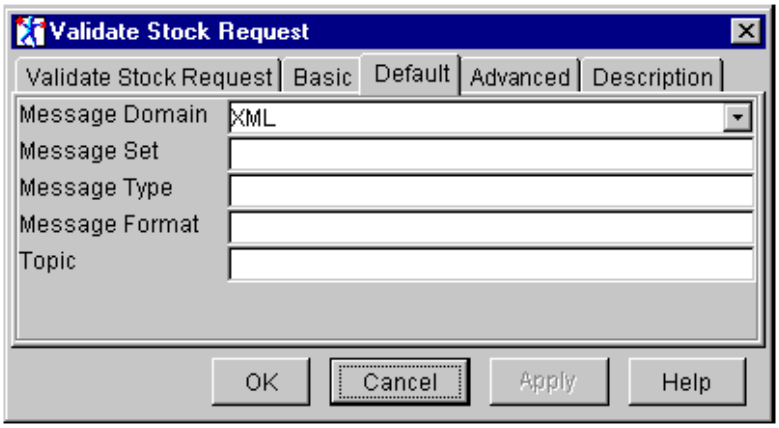

图5-15“MQInput”节点细节

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。

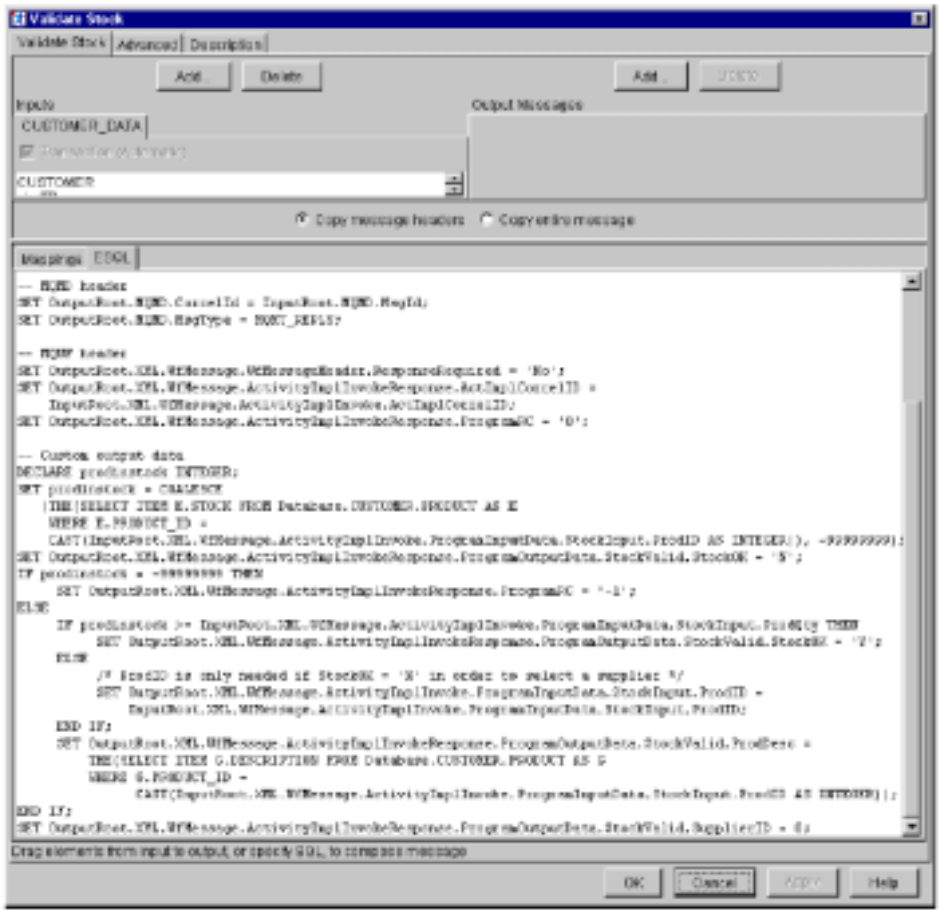

图5-16 计算节点细节

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。

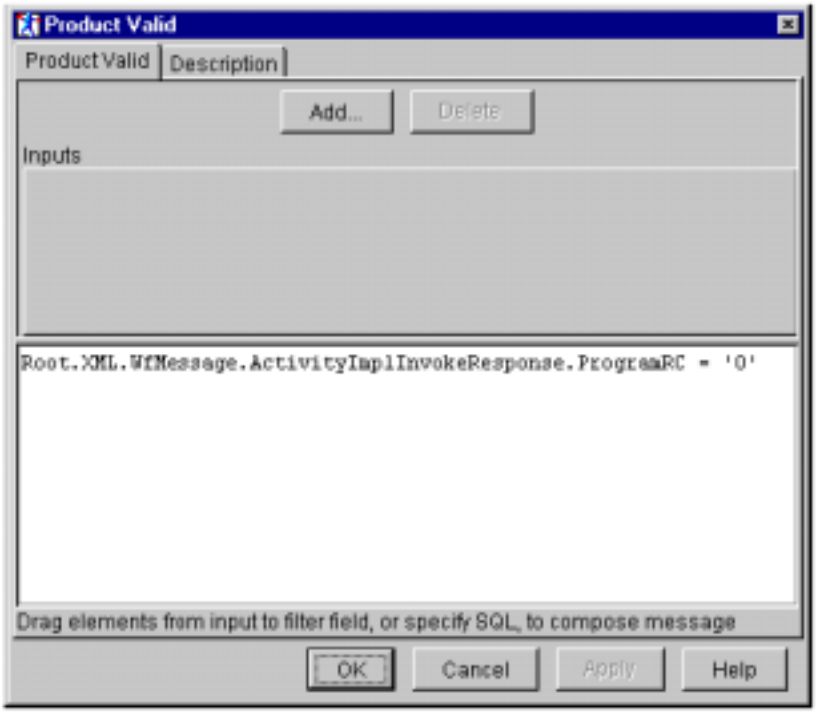

图 5-17　过滤节点细节

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。

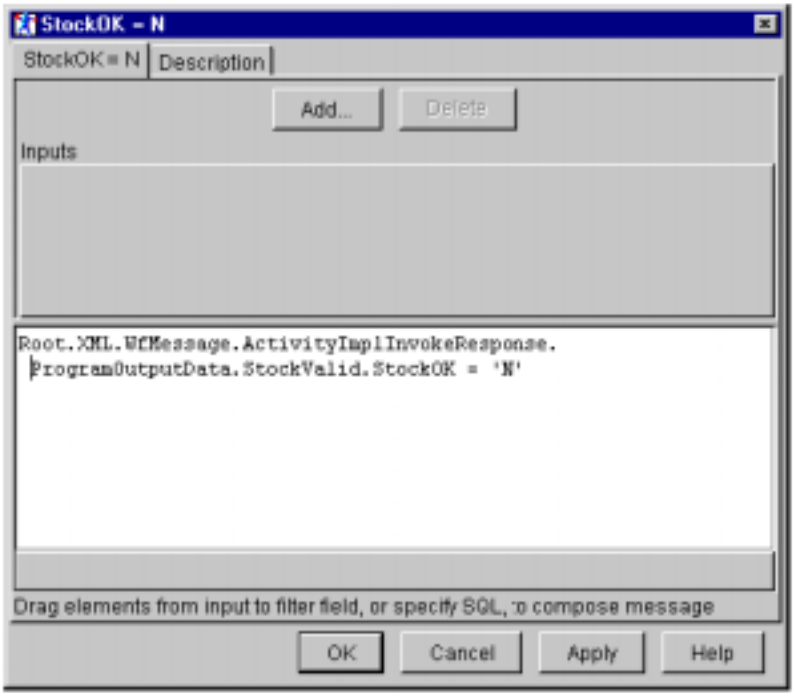

图 5-18　过滤节点细节

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。

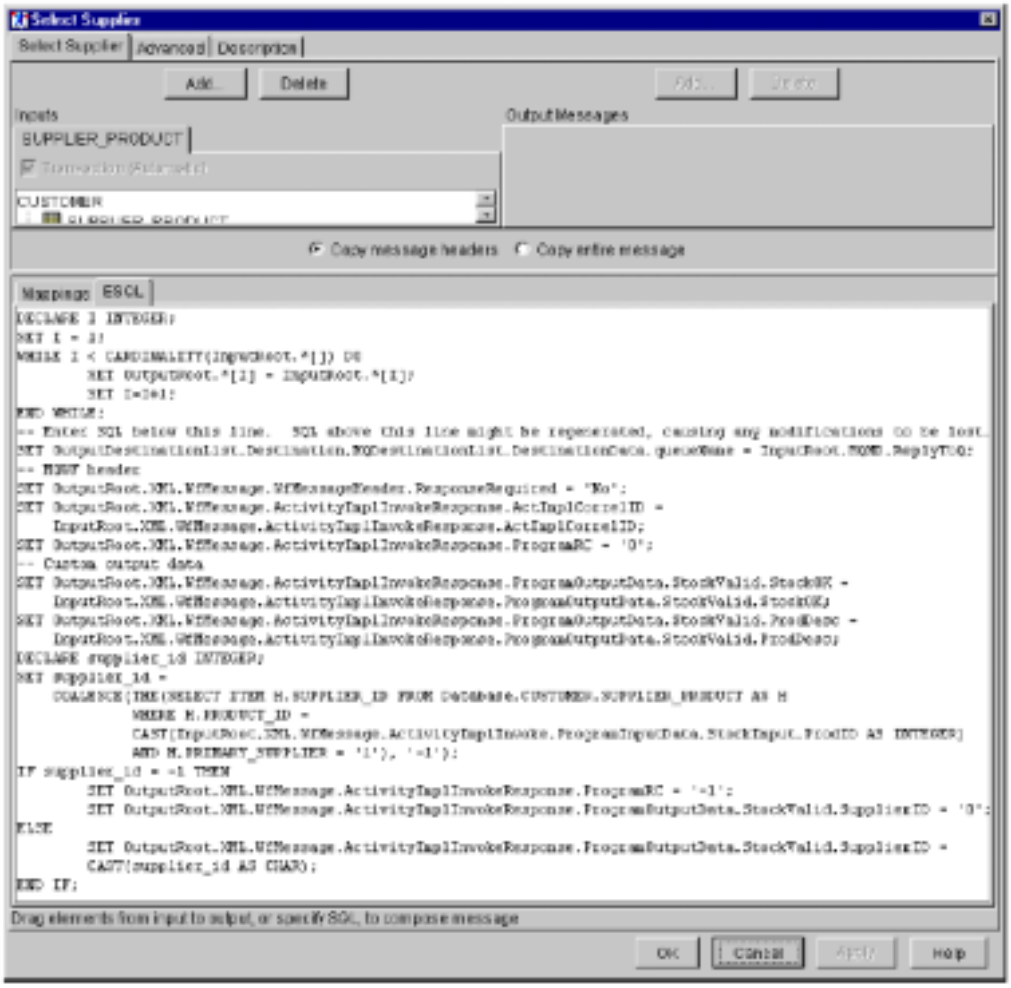
Select Supplier
Select Supplier
Advanced
Description
Add...
Delete
Inputs
SUPPLIER_PRODUCT
CUSTOMER
Output Messages
Copy message headers
Copy entire message
Mappings
ESQL
DECLARE I INTEGER;
SET I = 3;
WHILE I < CARDINALITY(InputRoot.*[]) DO
SET OutputRoot.*[I] = InputRoot.*[I];
SET I=I+1;
END WHILE;
-- Enter SQL below this line. SQL above this line might be regenerated, causing any modifications to be lost.
-- MQMT header
SET OutputRoot.MQMD.MessageHeader.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。

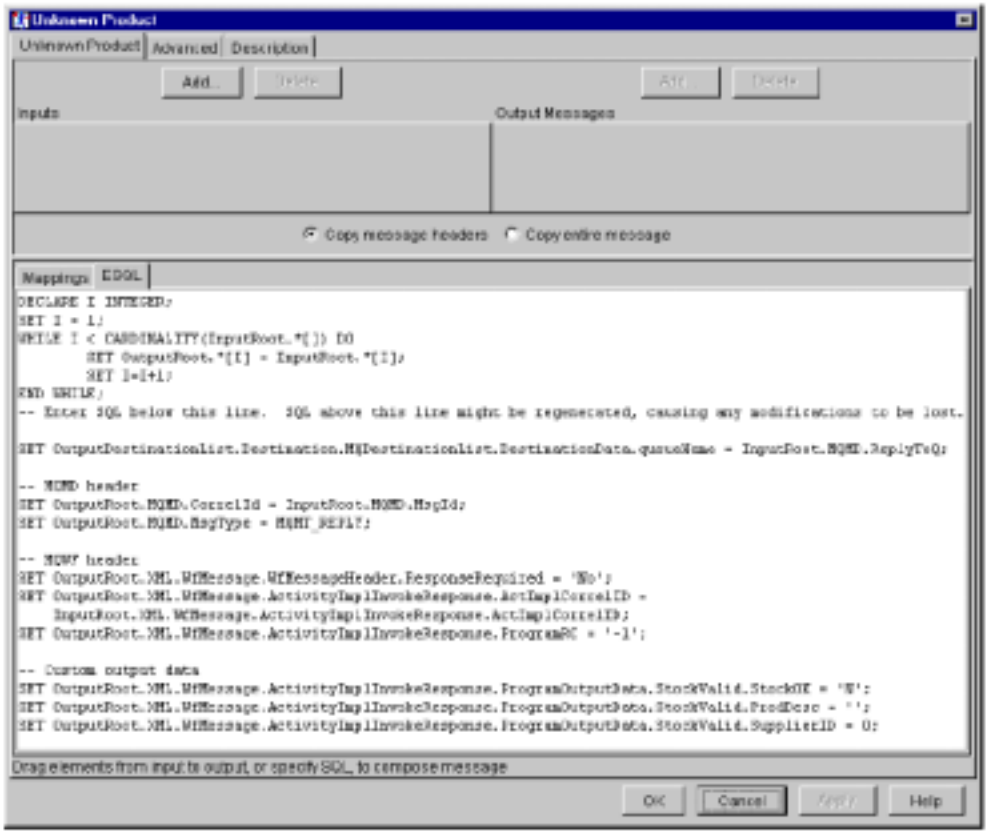

图5-20　计算节点细节

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。

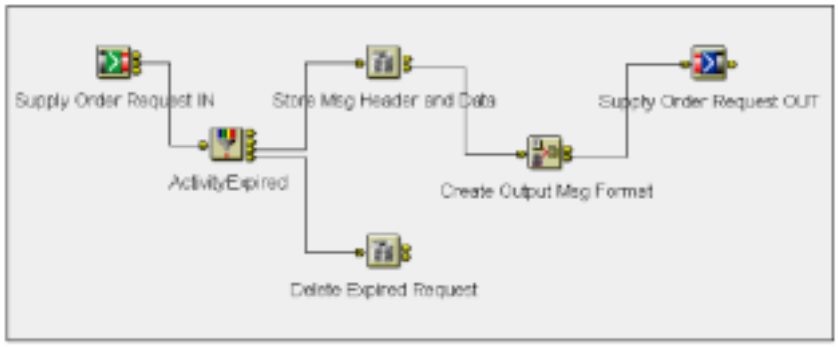

图 5-21 BuyXYZ_Supply_Order_PO 消息流

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。

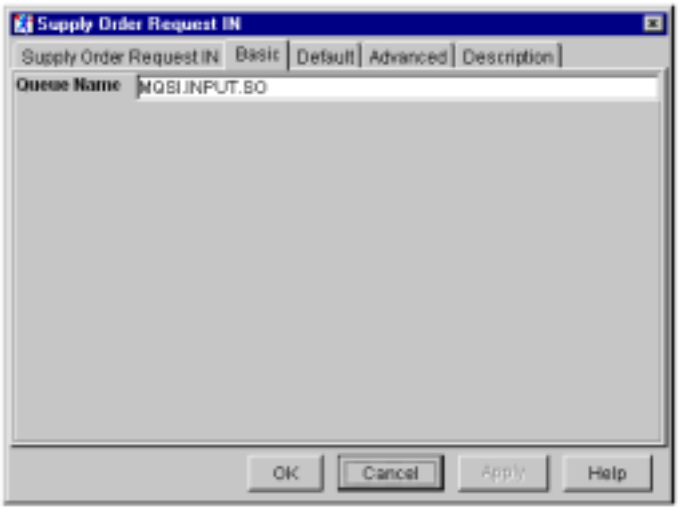

图5-23“MQInput”节点细节

在“默认”标签中（如图 5-24 所示），“消息域”被设置为 XML。

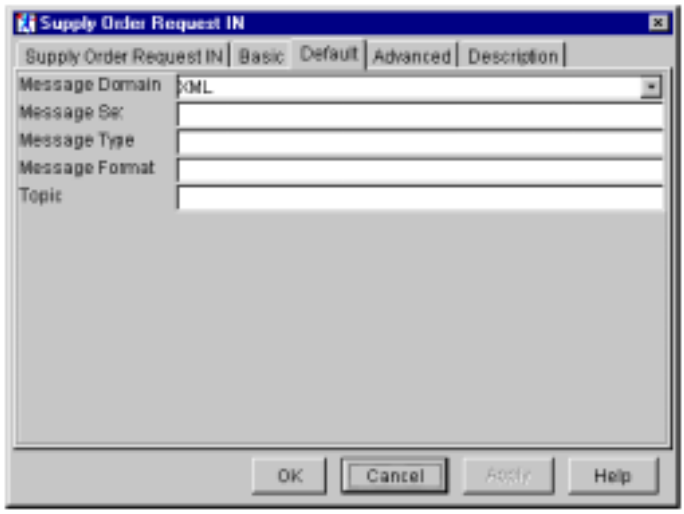

图5-24“MQInput”节点细节

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。

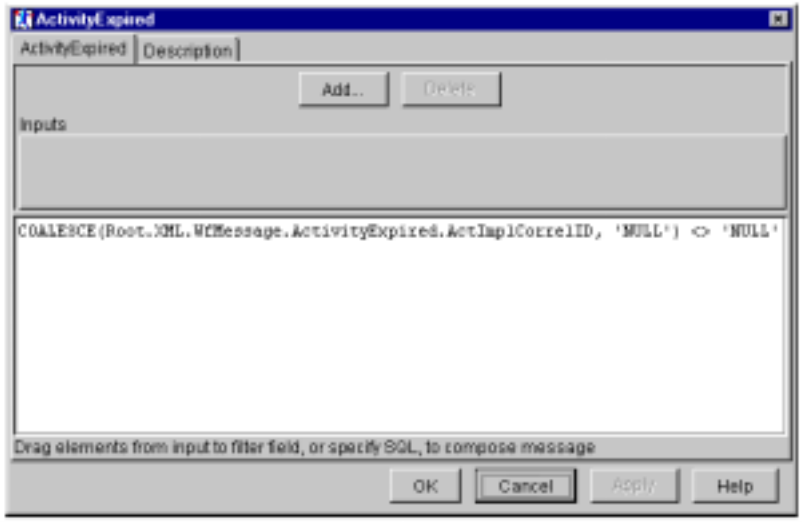

图5-25 过滤节点细节

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。

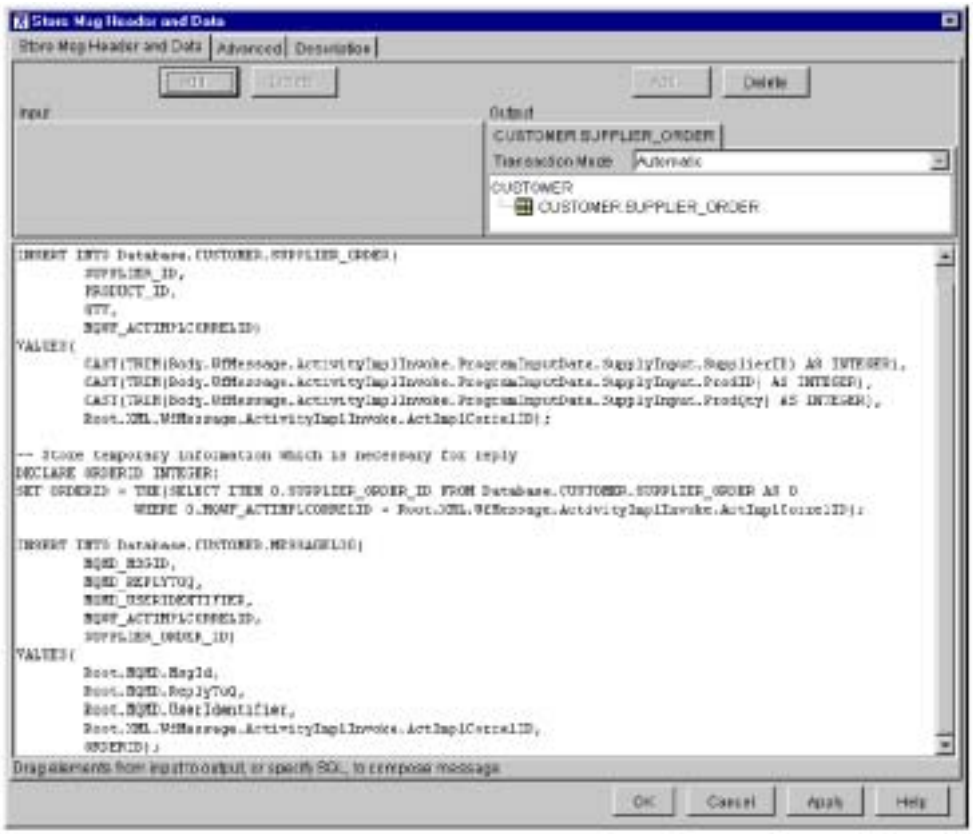

图 5-26 数据库节点细节

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。

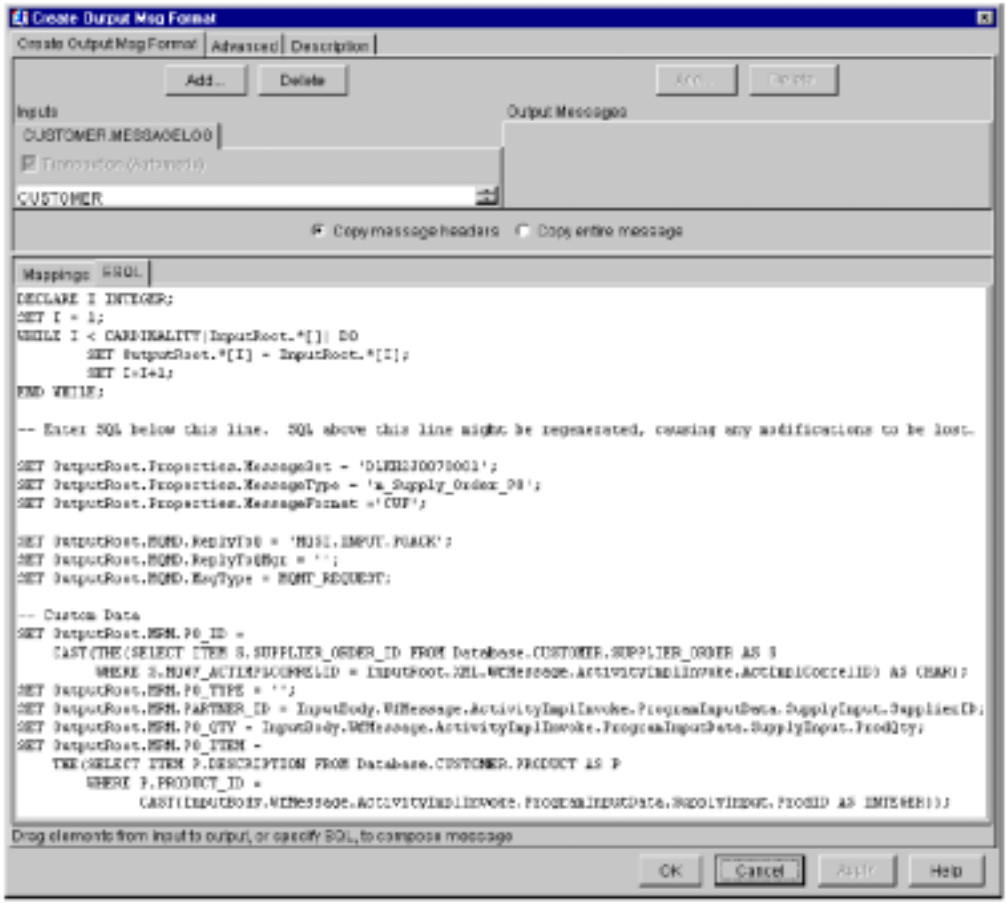

图5-27 计算节点细节

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。

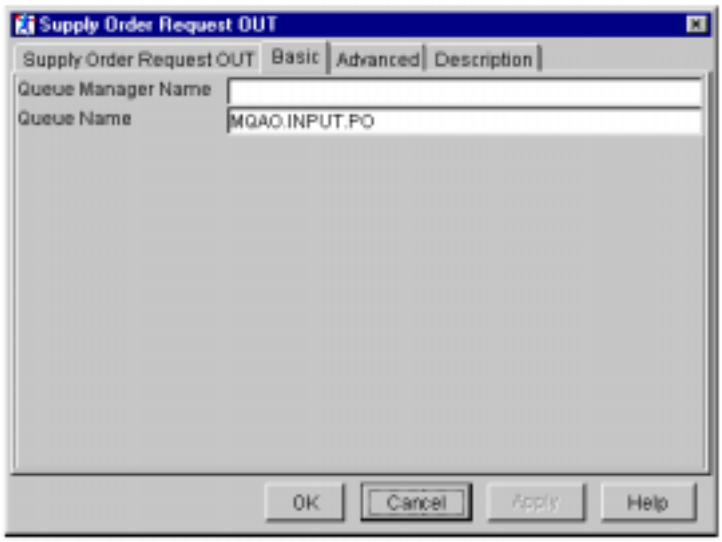

图 5-28MQOutput 节点细节

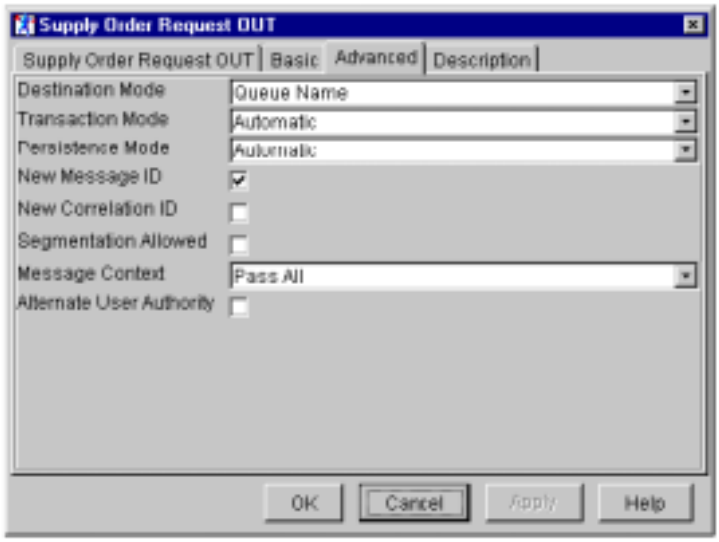

图 5-29MQOutput 节点细节

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。

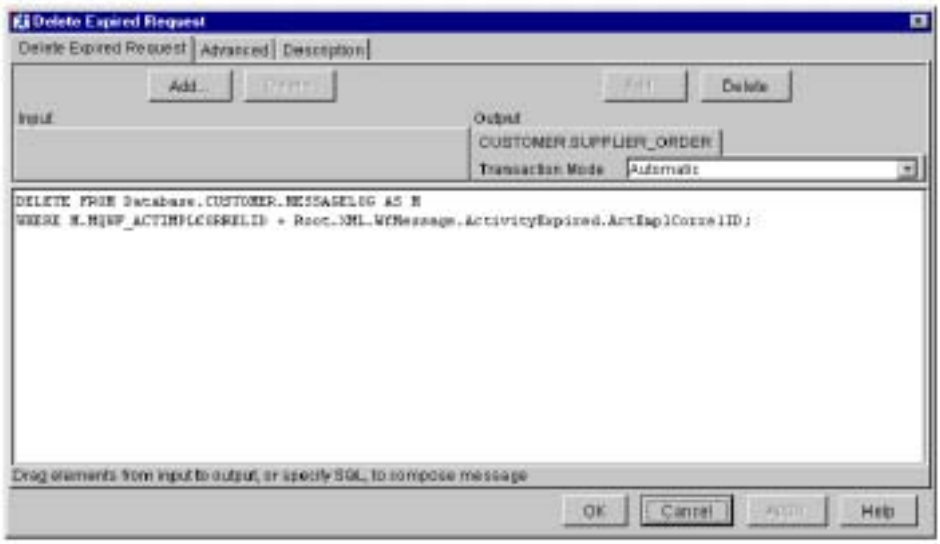

图5-30 计算节点细节

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。

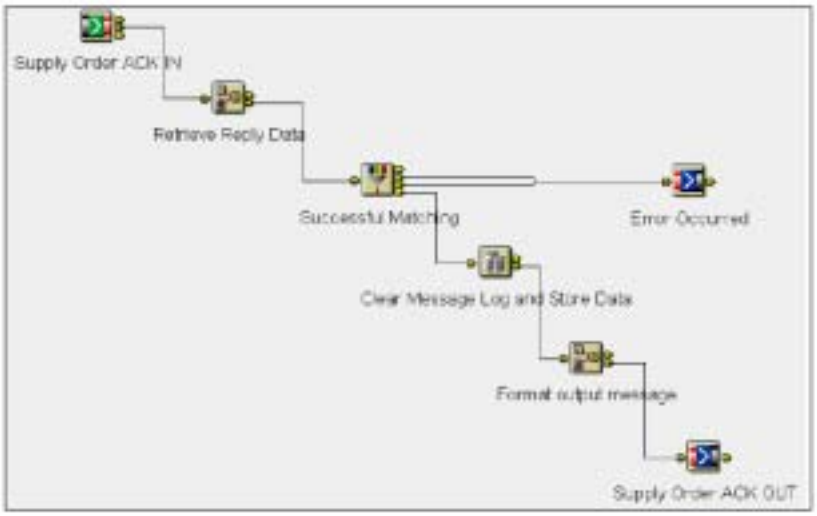

图5-31 消息流概览

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。

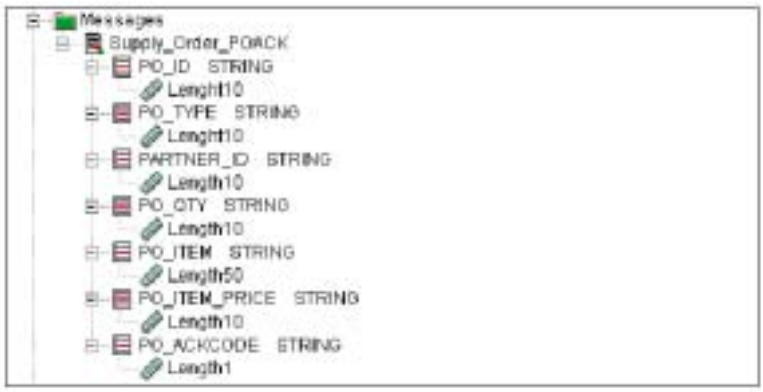

图 5-32 消息类型定义

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。

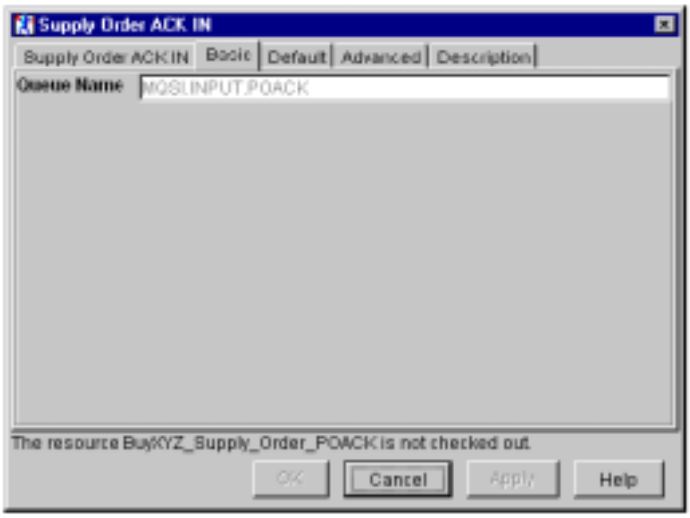

图 5-33MQInput 节点细节

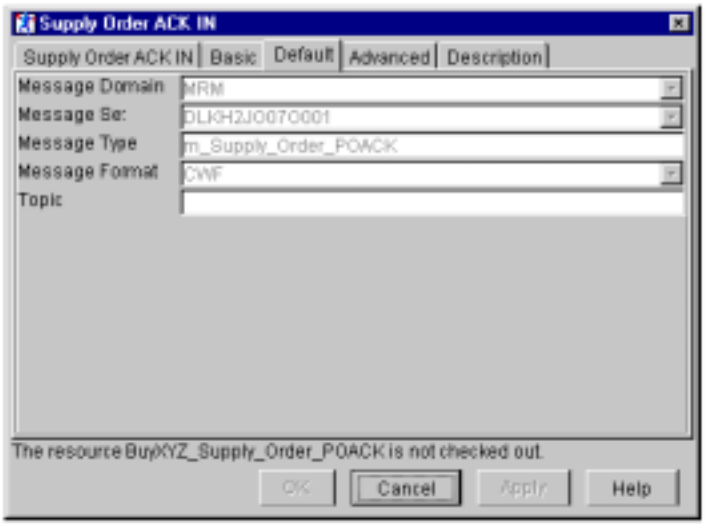

图 5-34MQInput 节点细节

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。

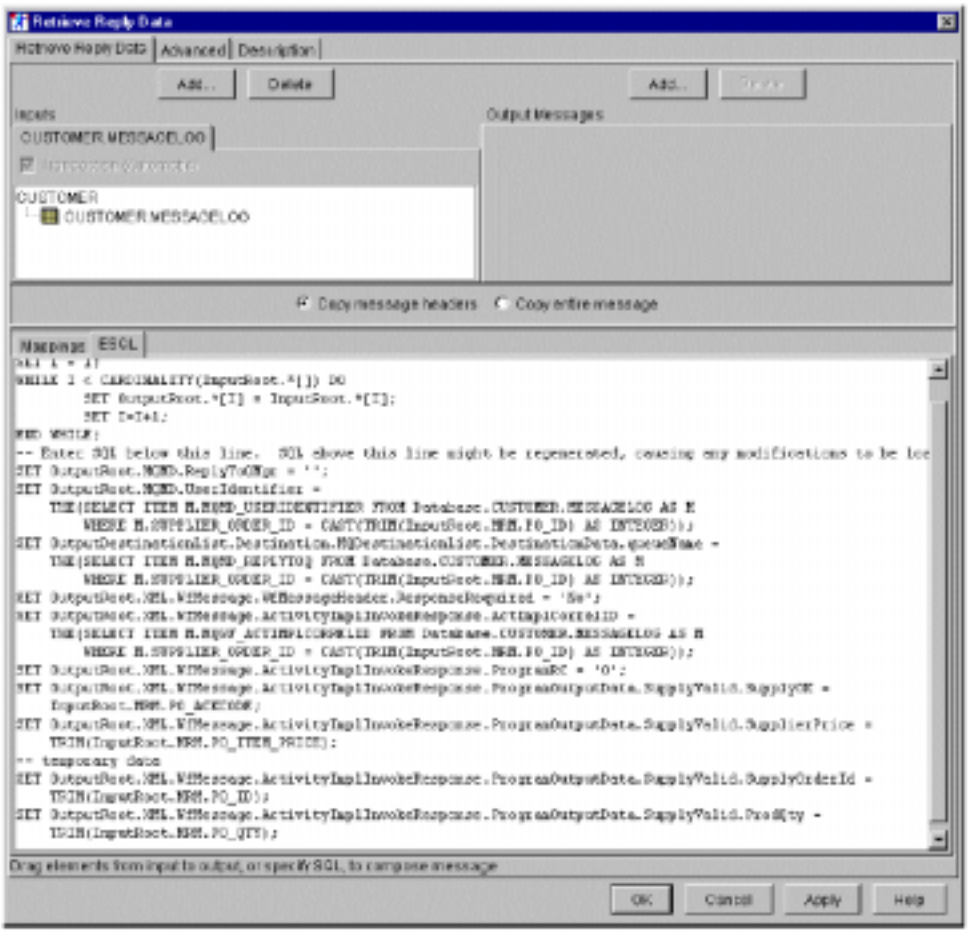

图5-35 计算节点细节

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。

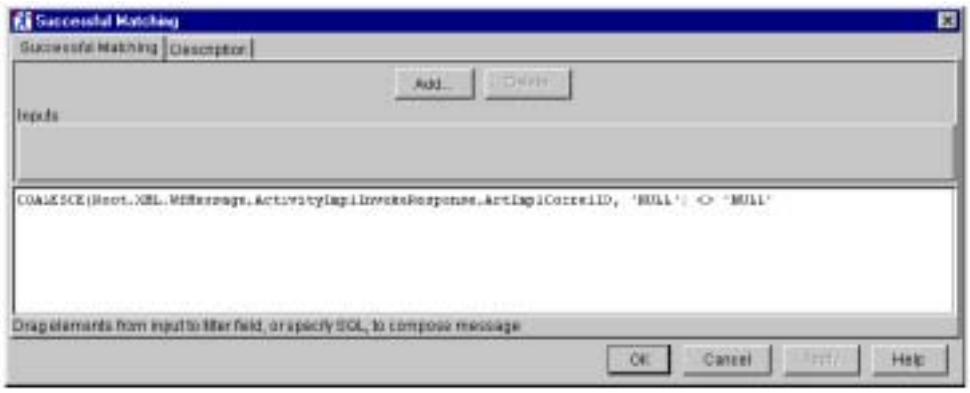

图 5-36 过滤节点细节

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。

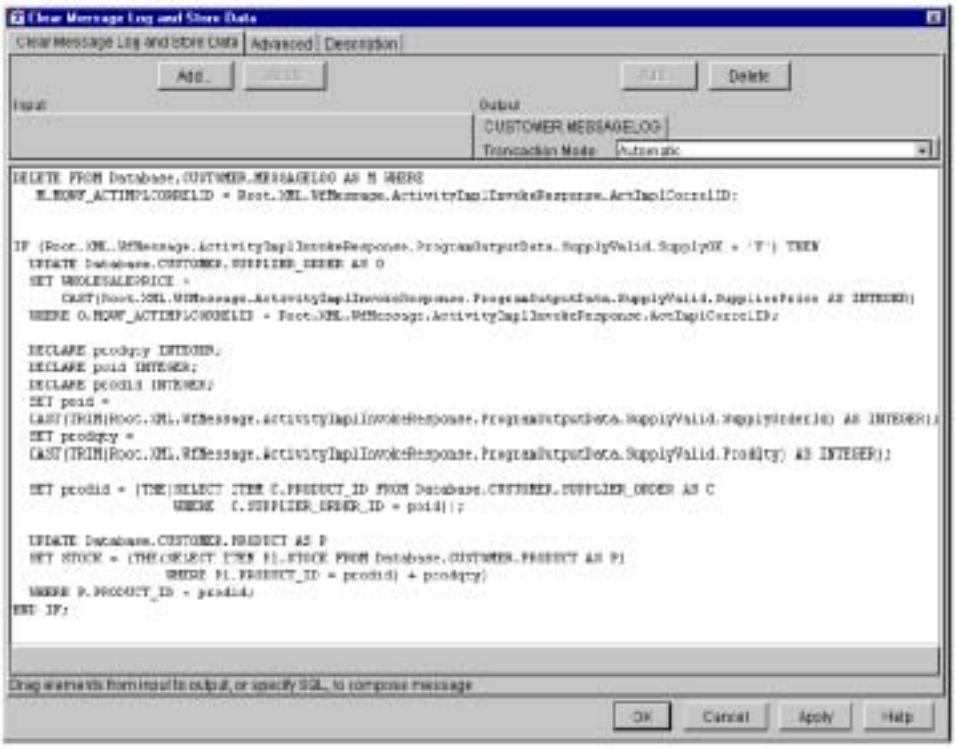

图 5-37 数据库节点细节

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。

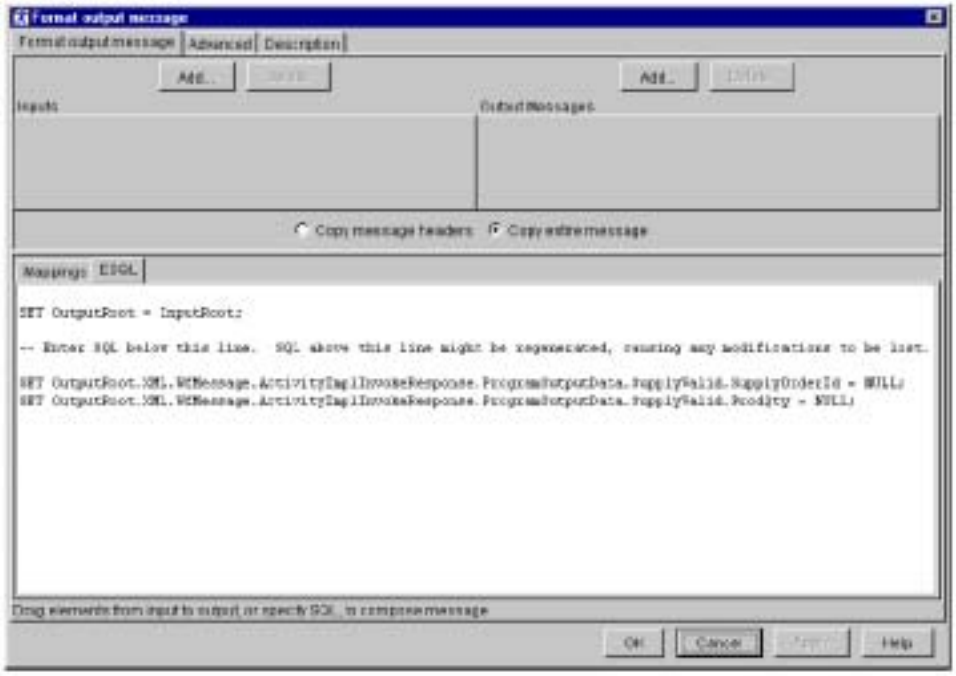

图 5-38 计算节点细节

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。

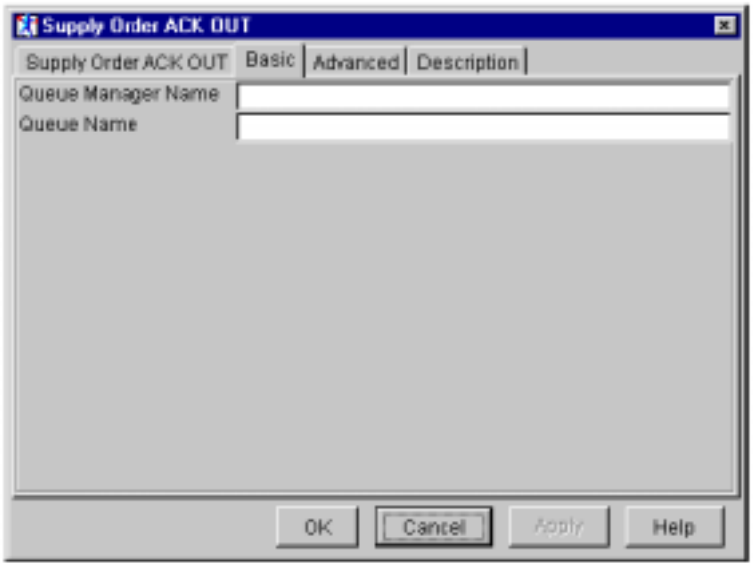

图 5-39MQOutput 节点细节

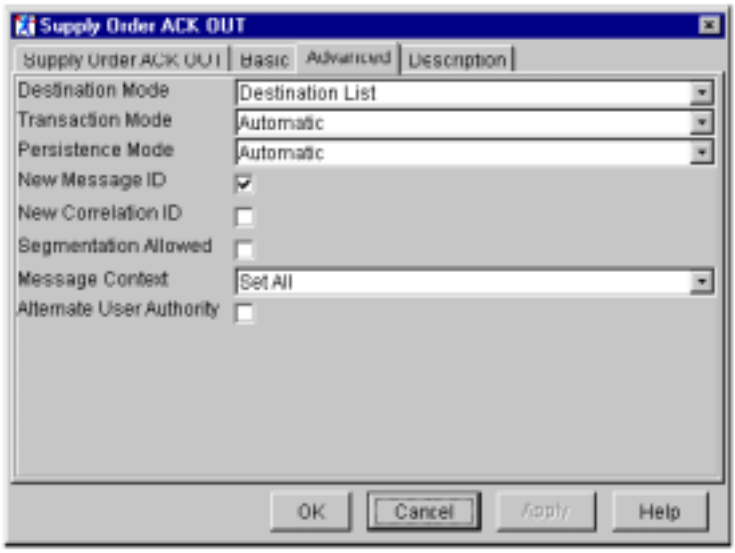

图 5-40MQOutput 节点细节

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。

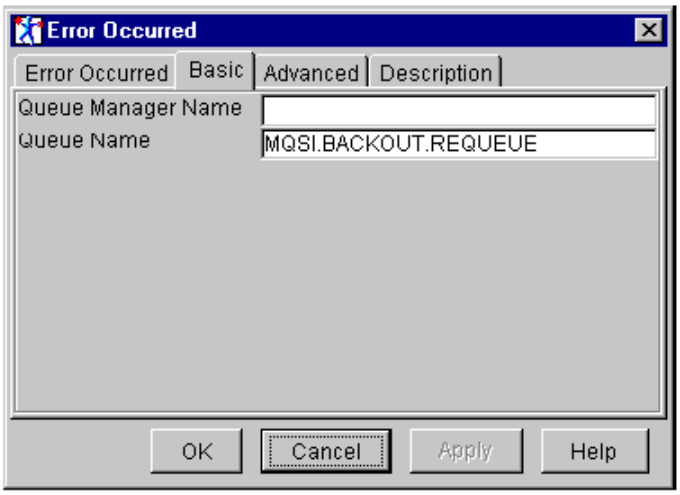

图 5-41MQOutput 节点细节

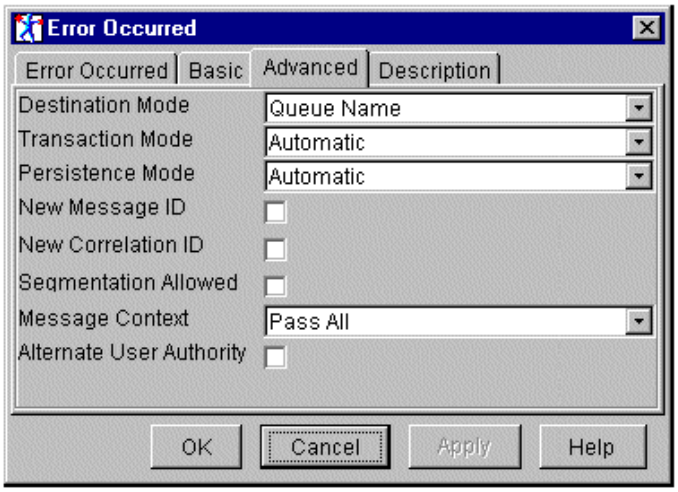

图 5-42MQOutput 节点细节

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。

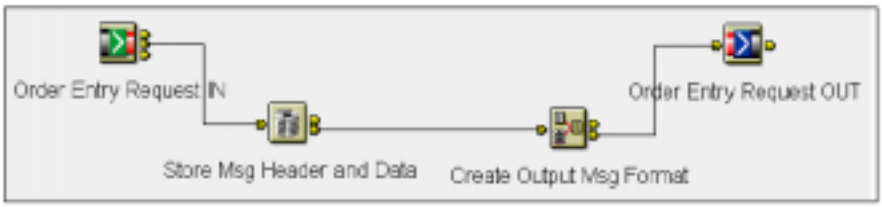

图5-43 消息流概览

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。

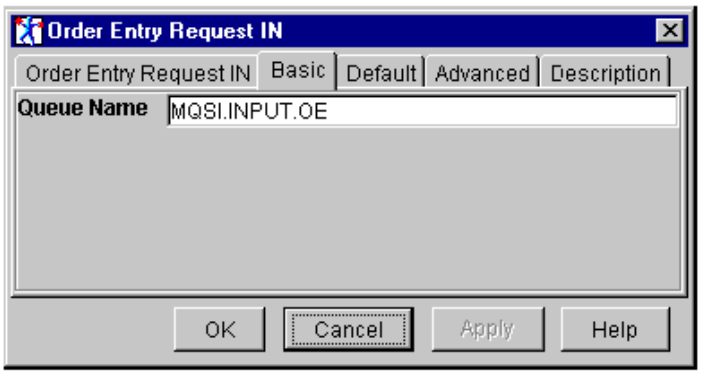

图 5-45MQInput 节点细节

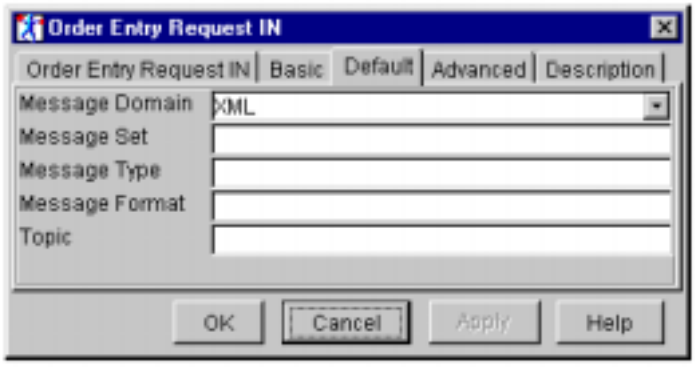

图 5-46MQInput 节点细节

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。

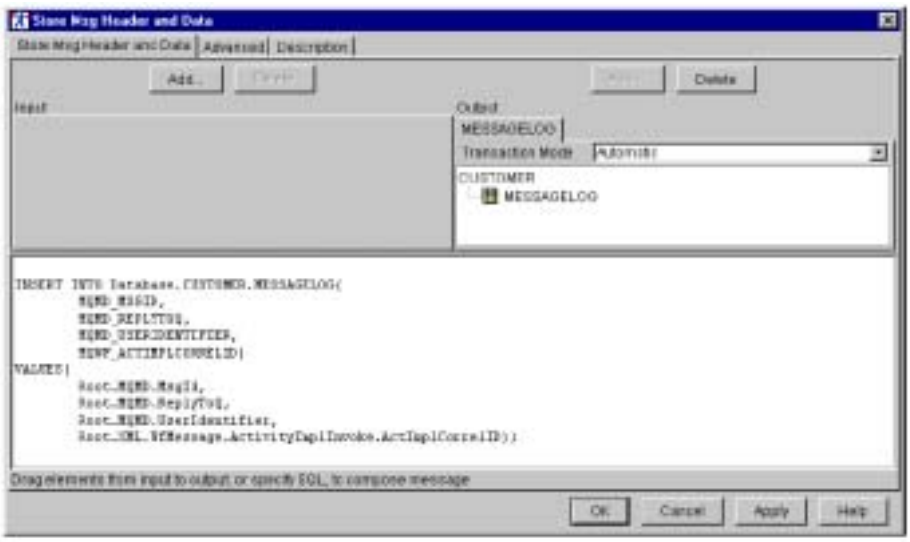

图 5-47 数据库节点细节

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。

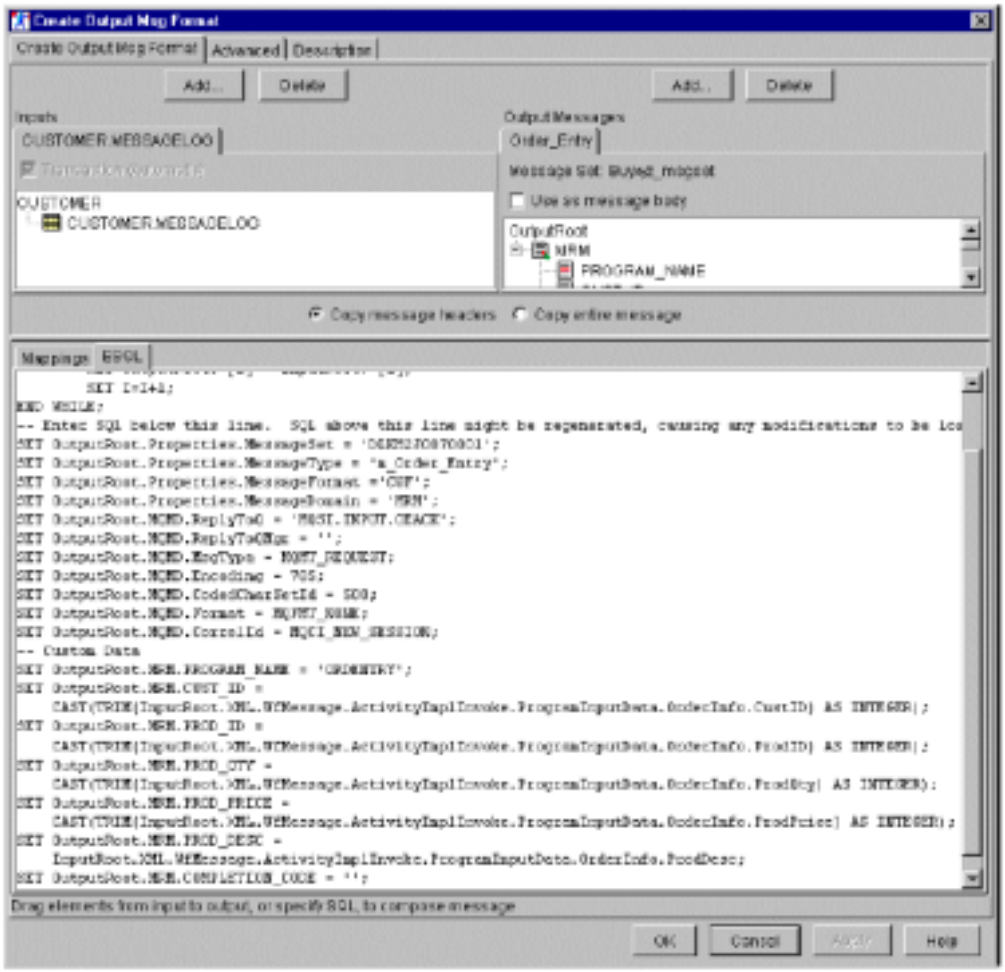

图5-48 计算节点细节

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。

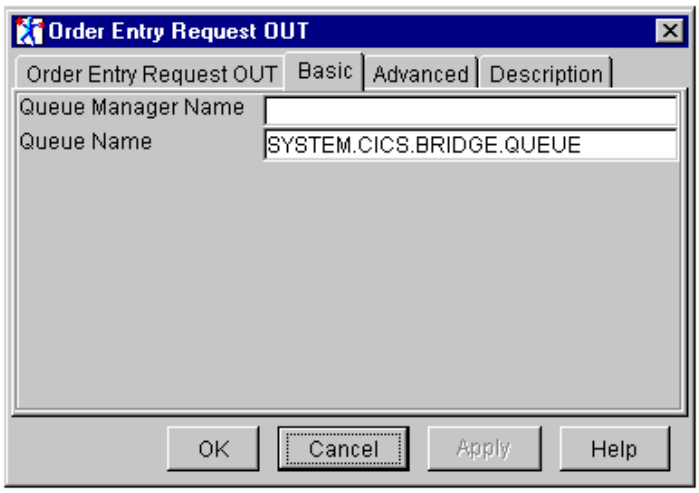

图 5-49 过滤节点细节

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。

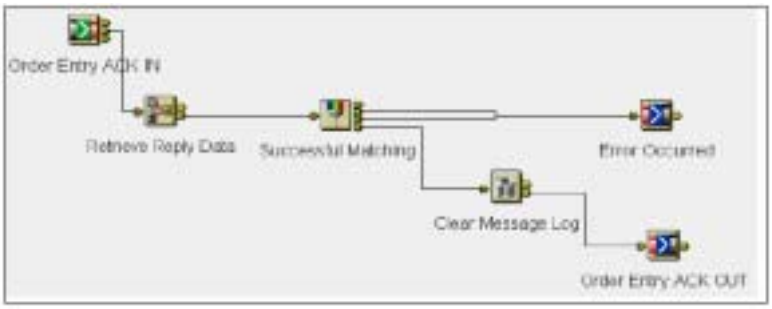

图5-50 消息流概览

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。

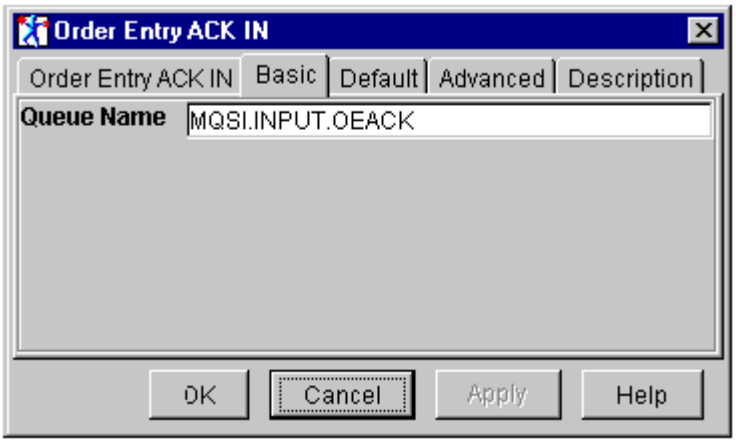

图 5-51MQInput 节点细节

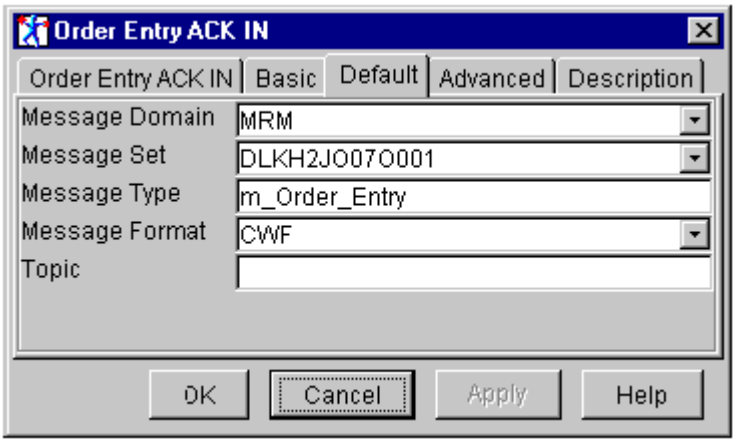

图 5-52MQInput 节点细节

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。

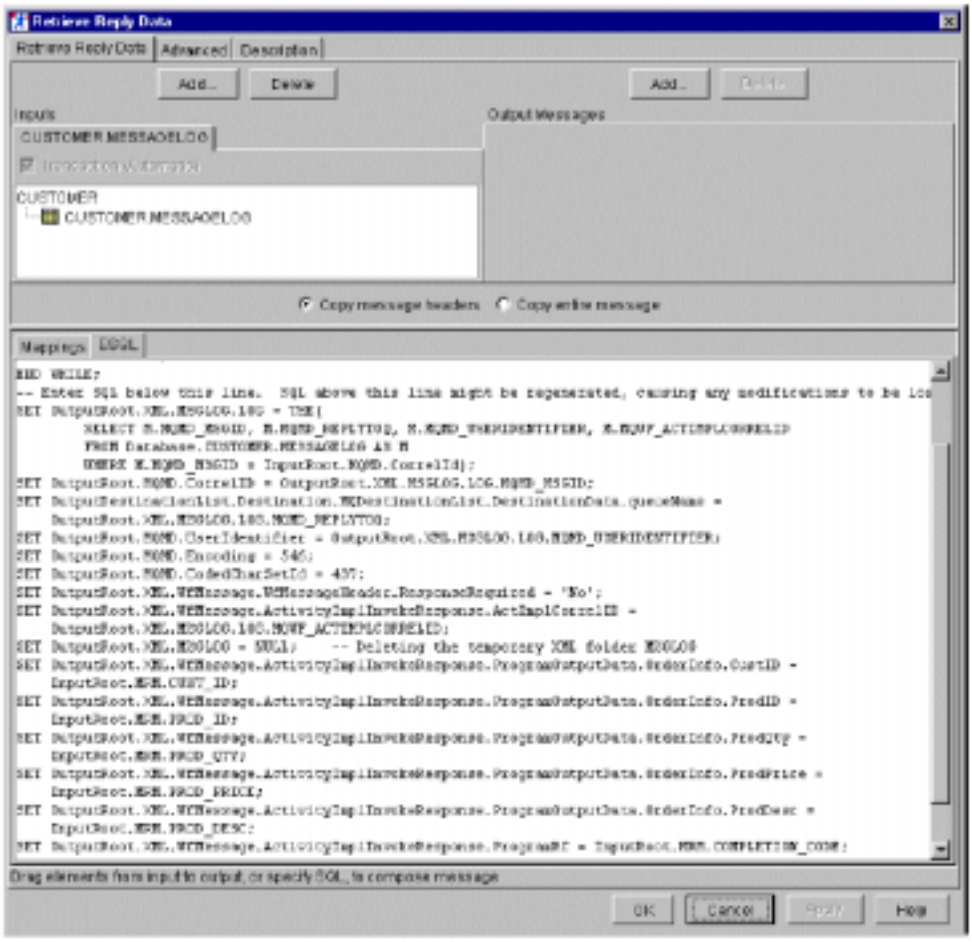

图 5-53 计算节点细节

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。

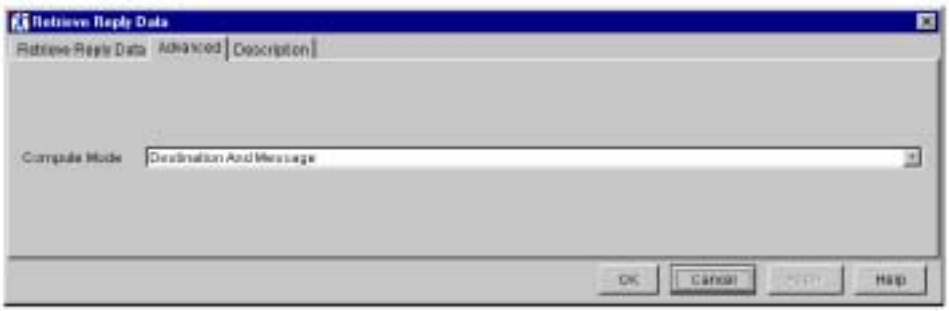

图5-54 计算节点细节

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。

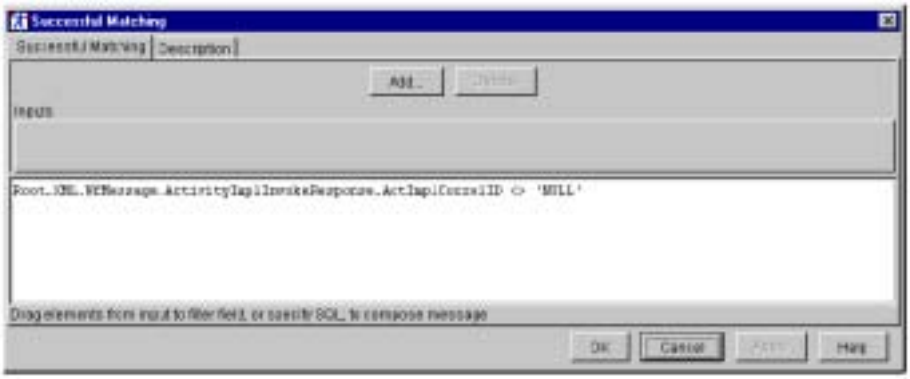

图5-55 过滤节点细节

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。

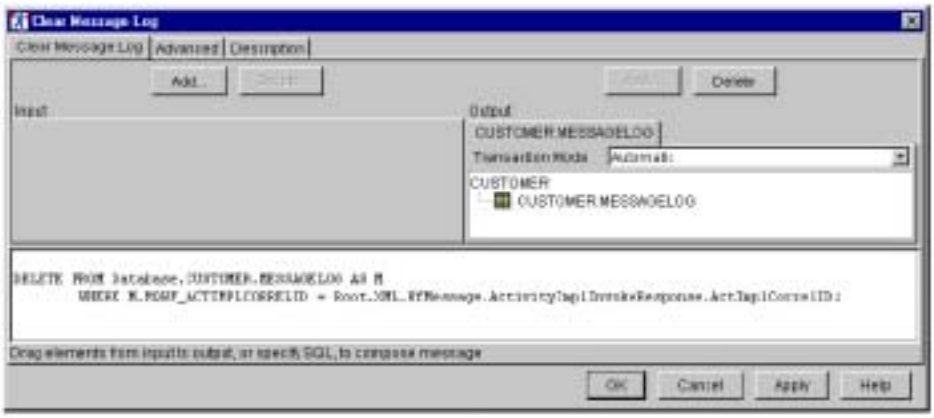

图 5-56 数据库节点细节

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。

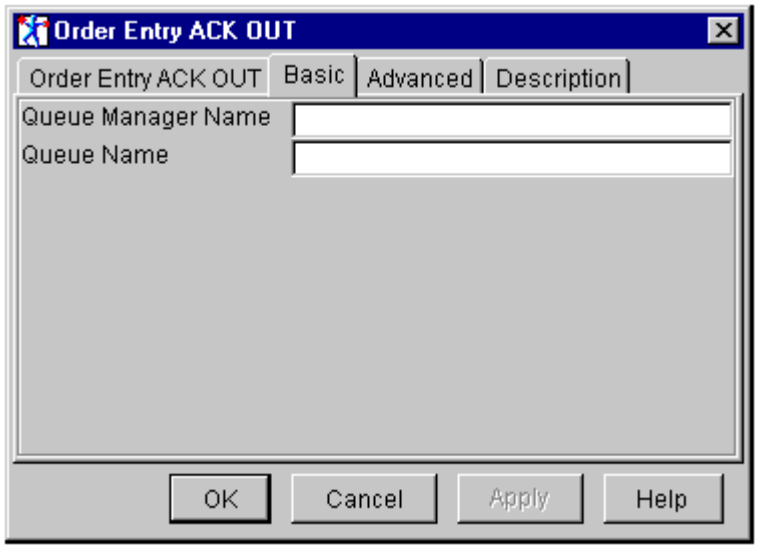

图 5-57MQOutput 节点细节

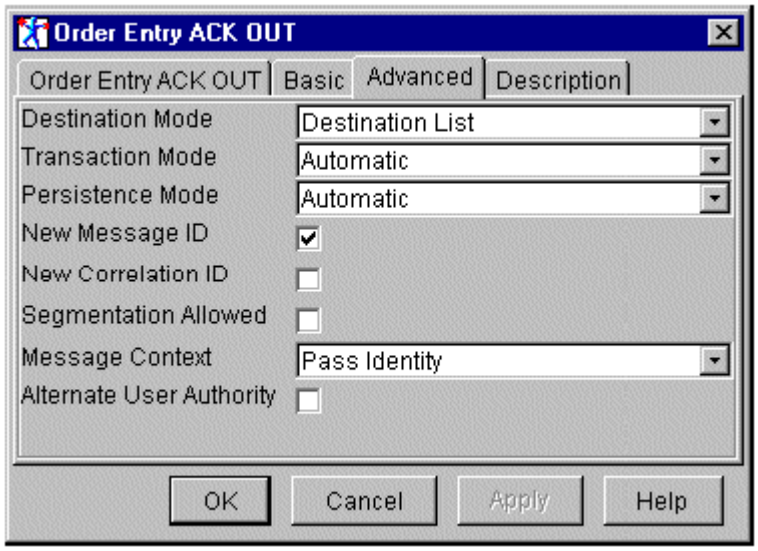

图 5-58MQOutput 节点细节

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。

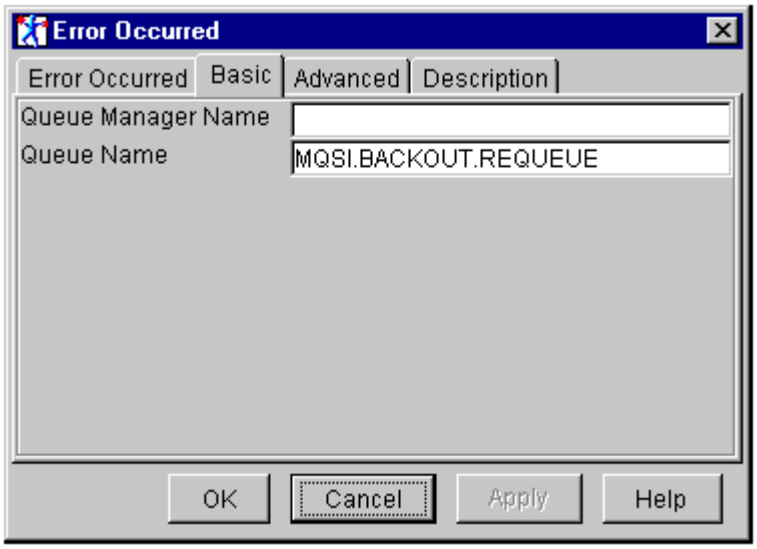

图 5-59 MQOutput 节点细节

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。

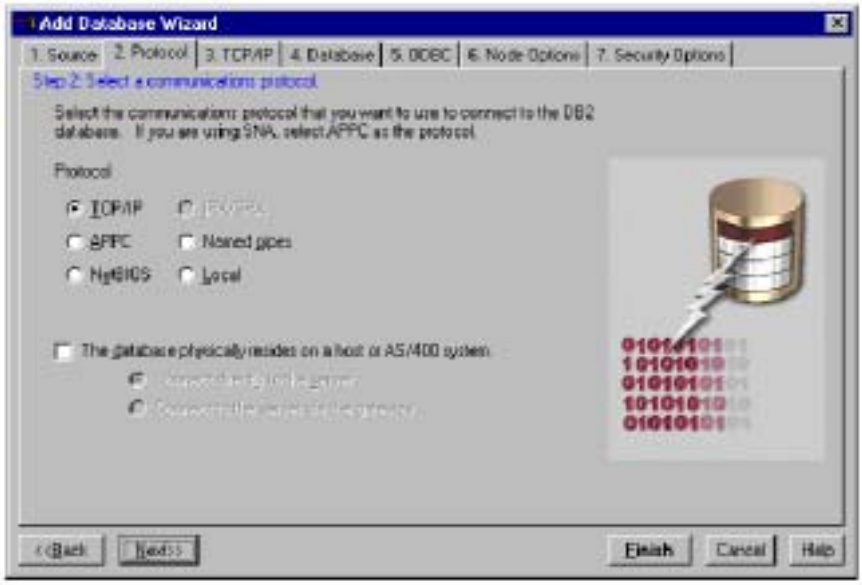

图5-61 添加数据库向导——步骤2

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。

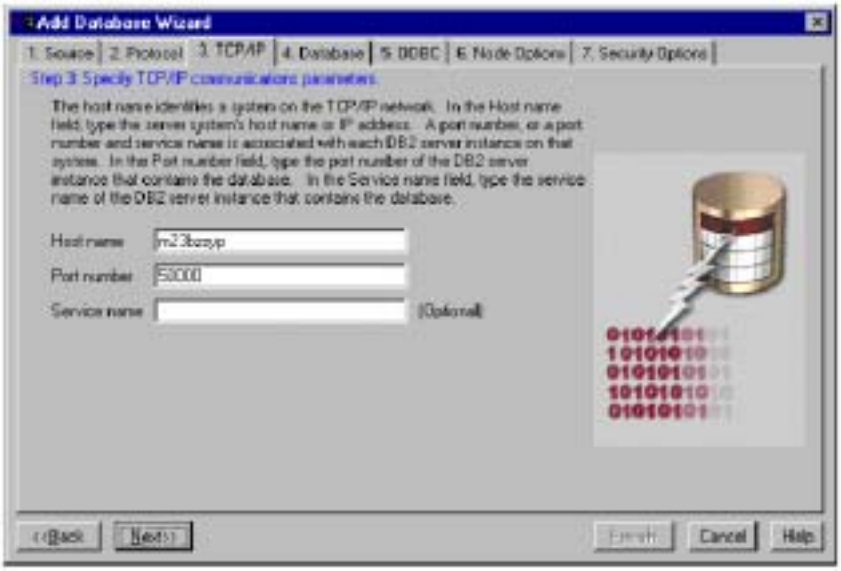

图5-62 添加数据库向导——步骤3

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。

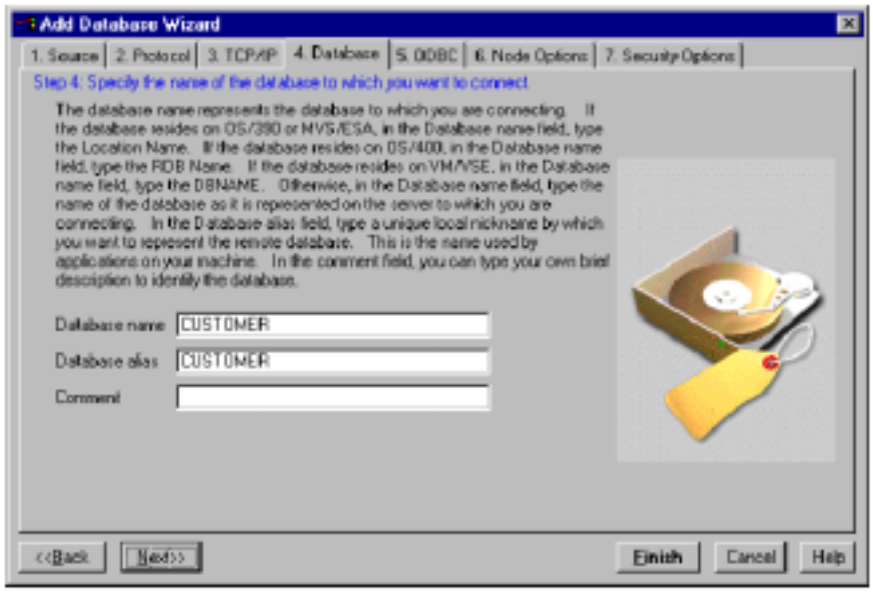

图 5-63 添加数据库向导——步骤 4

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。

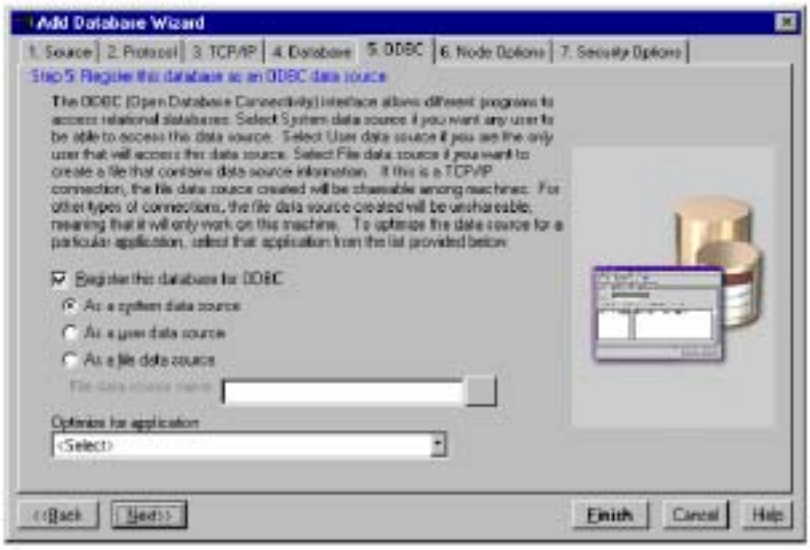

图5-64 添加数据库向导——步骤5

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》。