IBM Business Process Manager
V7.5.0

IBM Business Process Manager 概述

PDF 书籍和信息中心

为了方便打印和脱机阅读，提供了 PDF 书籍。有关最新信息，请参阅联机信息中心。

作为配套资料，这些 PDF 书籍包含的内容与信息中心的内容完全相同。PDF 书籍中的一些链接是为了在信息中心内使用而定制的，可能无法正确使用。

PDF 文档在信息中心的主要发行版（如 V7.0 或 V7.5）之后一个季度内提供。

PDF 文档的更新频率低于信息中心，但高于 Redbooks®。通常 PDF 书籍会在更改累积到足以出书的时候更新。

目录

第 1 章 IBM Business Process Manager V7.5 入门

了解 IBM® Business Process Manager 为业务流程管理提供哪些功能，以及业务流程管理的各个阶段（例如创建和部署流程应用程序）如何互相关联。

流程应用程序是 IBM Business Process Manager 中的流程及其组件的基本容器。流程设计器在编写环境中创建流程应用程序，并可以包括支持执行所需的服务、任务和工件。

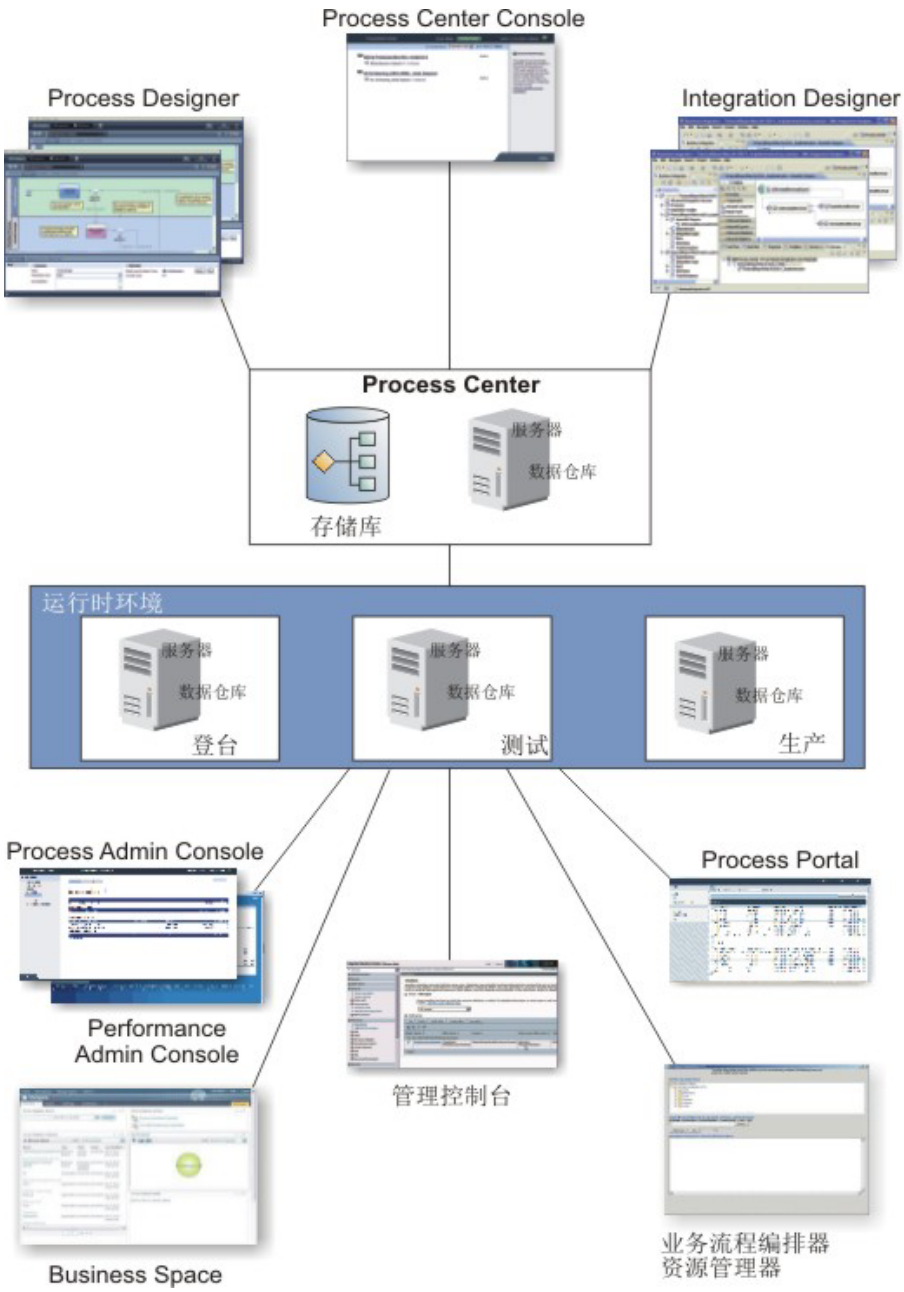
高级集成服务在 Integration Designer 中实现并与流程应用程序相关联。在 Process Center 中，流程应用程序部署至 Process Server，后者是 IBM Business Process Manager 的流程运行时环境。

同样，在 Integration Designer 中创建的自动流程可以使用在 IBM Process Designer 中开发的人员活动流。

产品概述

IBM Business Process Manager 的各个组件提供了统一的 BPM 存储库，提供了供作者、管理员和用户使用的工具，并提供了运行时平台。不同的产品配置支持不同级别的复杂性和不同程度的业务流程管理。

下图说明典型 IBM Business Process Manager 配置:



- 多名用户可以从 IBM Process Designer 和 IBM Integration Designer 编写环境连接到 Process Center。
- 在 Process Designer 和 Integration Designer 编写环境中，流程和服务设计者创建可部署的流程应用程序和可复用的工具箱。流程应用程序包含流程模型和服务实现，包括所有必需的支持文件。它们存储在 Process Center 存储库中，可以在其中共享。
- Process Center 包括 Process Center Server 和 Business Performance Data Warehouse，使在 IBM Process Designer 中工作的用户可运行其流程应用程序并存储性能数据以在开发期间用于测试和回放目的。

- 在 Process Center 控制台中，管理员将安装一些流程应用程序，这些流程应用程序已准备好在这些环境中对 Process Server 执行登台、测试或生产。
- 在 Process Center 控制台中，管理员管理所有已配置环境中的流程应用程序的运行实例。
- 在 IBM Process Portal 中，最终用户执行分配的任务。Process Center Server 和已配置的运行时环境中的 Process Server 可以运行创建所分配的任务的流程应用程序。
- 使用 Process Portal，连接参与者可以连接到 Process Center Server 或任何已配置的运行时环境中的 Process Server，这取决于流程处于开发、测试阶段还是已发布到生产环境的阶段。
- Performance Data Warehouse 定期从 Process Server 或 Process Center Server 检索已跟踪数据。用户可在 Authoring Environment 和 IBM Process Portal 创建并查看使用此数据的报告。
- 在 Process Admin Console 和 Performance Admin Console 中，管理员可管理并维护所有运行时服务器。

IBM Business Process Manager V7.5 配置

IBM Business Process Manager 的不同配置与公司的业务流程管理计划的典型入口点或阶段相关联。

表 1. IBM Business Process Manager 配置

配置	阶段
Advanced	**变换** 全套业务流程管理功能 • 对大容量流程自动化的扩展支持 • 内置 SOA 组件，面向广泛的企业范围服务集成与协调
Standard	**程序** 针对典型的业务流程管理项目进行配置 • 面向多项目改进计划，业务参与度较高 • 基本的系统集成支持 • 迅速将时间转变为价值，提高了用户生产力
Express	**项目** 针对第一个业务流程管理项目进行配置 • 迅速将时间转变为价值：提高了用户生产力 • 入门价格低廉 • 安装与配置简单方便

IBM Business Process Manager V7.5 配置功能

了解 IBM 提供的业务流程管理产品及功能，并选择适合于您的企业的产品及功能。

IBM Business Process Manager 是单一的 BPM 平台，此平台将以人员为中心的功能以及以集成为中心的功能组合成统一化的产品。此产品的不同配置供不同用户使用，并能满足企业的不同需求。产品配置可以结合成协作式编写环境和网络部署运行时环境。

表 2. IBM Business Process Manager 配置功能

功能	Advanced	Standard	Express
WebSphere Lombardi Edition 兼容执行	X	X	X
Process Designer (BPMN)	X	X	X

表 2. IBM Business Process Manager 配置功能 (续)

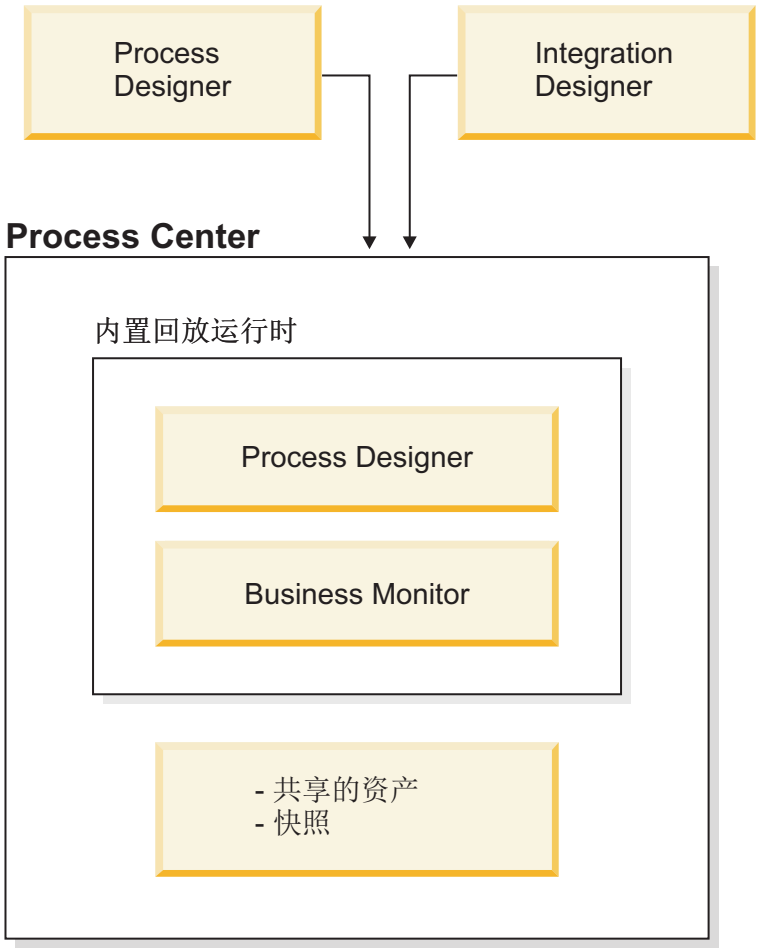
功能	Advanced	Standard	Express
协作编辑 / 立即回放	X	X	X
交互式“流程 Coach”用户界面	X	X	X
基于 ILOG 的流程规则	X	X	X
实时监视和报告	X	X	X
性能分析与优化器	X	X	X
Performance Data Warehouse	X	X	X
Process Center / 共享资产存储库	X	X	X
流程作者数和最终用户数不受限	X	X	200 个用户 / 3 个作者
高可用性: 集群与不受限的核心	X	X	• 4 核心产品 • 2 核心开发 • 无集群
WebSphere Process Server 兼容执行	X		
Integration Designer (BPEL / SOA)	X		
内置企业服务总线 (ESB)	X		
事务支持	X		
集成适配器	X		
灵活的 Business Space 用户界面	X		

Process Center 存储库

Process Center 包括用于在 IBM Business Process Manager 编写环境、Process Designer 和 Integration Designer 中创建的所有流程、服务和其他资产的存储库。

Process Center 是运行时，Process Designer 和 Integration Designer 在其中共享资产，实际上，这使它们能够以高度交互方式协作开发业务流程。这些业务流程可以使用通过 Business Monitor development toolkit 创建的监视点。这将产生一个业务流程，在运行时，您可以对其进行检查以提高其在实际工作环境中的效能。Business Monitor 提供带有标尺和记分卡的仪表板视图。可添加警报和通知以允许您不时了解业务流程正在执行的工作。您可以确定并更正运行中的业务流程中的资源分配瓶颈、效率不足之处和错误，从而提高业务流程的性能。

在下图中，您将看到多款相关的组件，这些组件合在一起使您能够构建复杂的业务流程。



Process Center 控制台提供维护此存储库所需的工具。

- 在 Process Center 控制台中，可创建流程应用程序和工具箱并授予其他用户对这些流程应用程序和工具箱的访问权。
- 通过使用编写环境，您可以在流程应用程序中创建流程模型、服务和其他资产。
- Process Center 包括 Process Center Server 和 Performance Data Warehouse，它们允许在编写环境中工作的用户运行流程并存储性能数据以用于测试和回放。
- 在 Process Center 控制台中，管理员在那些环境中的流程服务器上安装已准备好投入测试或生产的流程应用程序。
- 在 Process Center 控制台中，管理员管理已配置环境中的流程应用程序的运行实例。

Process Center Console 提供了一个方便的位置来创建和维护高级别容器，例如流程应用程序和工具箱。不经常在 Designer 视图中工作的管理员可以使用 Process Center 控制台来提供一个框架，在此框架内，BPM 分析员和开发者可以构建他们的流程和底层实现。管理员的另一项主要任务是，通过为用户和组设置适当的权限来管理对 Process Center 存储库的访问权。

具有适当权限的用户可以直接在 Process Designer 和 Integration Designer 中执行某些管理任务。例如，对流程应用程序具有写访问权的开发者如果想捕获特定里程碑处所有项目资产的状态，那么可以在 Designer 视图中工作时创建快照。

Process Server 和运行时环境

Process Server 提供了单一的 BPM 运行时环境，此环境能够支持各种业务流程、服务编排和集成功能。

在编写环境中，Process Center 中的集成流程服务器使您可以在构建流程时运行流程。您准备就绪时，可以在运行时环境中的流程服务器上安装和运行这些流程。Business Performance Data Warehouse 组件从在流程服务器上运行的流程收集和聚集流程数据。您可以使用此数据来改进业务流程。

Process Admin Console 使您能够管理运行时环境（例如登台环境、测试环境和生产环境）中的流程服务器以及作为 Process Center 组成部分的流程服务器。

编写环境

IBM Business Process Manager Advanced 提供了两个编写环境。使用 IBM Process Designer 以对涉及人员任务的业务流程高效建模。使用 IBM Integration Designer 可构建自包含服务或调用其他现有服务（如 Web Service）的服务、企业资源应用程序或在 CICS 和 IMS 中运行的应用程序。

产品的所有版本均提供了 Process Designer。IBM Business Process Manager Advanced 还提供了 Integration Designer 及其相关编辑器和适配器。

Process Designer

流程是 IBM Business Process Manager 中的主要逻辑单元。它是流程定义的所有组件（包括服务、活动和网关; 计时器、消息和异常事件; 顺序线、规则和变量）的容器。对流程建模时，您将创建可重复使用的业务流程定义 (BPD)。请使用 IBM Process Designer 来创建可以包含人员任务的流程模型。

Process Designer 可帮助您开发业务流程。通过易于使用的面向图形的工具，可创建组成业务流程的操作序列，并可在环境变化时随时间推移重新绘制该流程。如果一个或多个活动要求访问用于为业务流程提供数据的大型后端系统或服务，例如，为获取有关客户的信息，那么可以使用 Integration Designer 来满足该要求。通过使用简单的接口，Process Designer 中的活动可以调用在 Integration Designer 中创建的服务。该服务可以使用调解流来变换、路由和增强数据及适配器，以便以标准方式访问许多后端系统。简而言之，Process Designer 侧重于业务流程，而 Integration Designer 侧重于用于补充业务流程的自动化服务。

所有 Process Designer 项目都包含在流程应用程序中。您将这些流程应用程序及相关工件存储在 Process Center 存储库中。流程应用程序可以共享放置在工具箱中的资产。

IBM Business Process Manager 提供了多个用户界面，使您能够对业务流程进行建模、实现、模拟和检查。您可以在 Process Center 控制台中创建和管理流程应用程序、工具箱、跟踪和快照。您可以在 Process Designer 中创建流程模型、报告和简单服务。可以在 Inspector 中运行和调试流程。并且，还可以在 Optimizer 中运行模拟。

在 Process Designer 中开发的流程应用程序可以随时在 Process Center 服务器上运行，或保存到快照并部署在 Process Server 上。在 Integration Designer 中开发并与流程应用程序关联的服务也是如此。

Integration Designer

Integration Designer 提供了编辑器和辅助功能来帮助开发者创建复杂的自动化流程和服务。它作为 IBM Business Process Manager Advanced 中的组件提供，或者作为独立的工具集提供以用于其他用途。

IBM Integration Designer 已设计成完整的集成开发环境，供构建集成式应用程序的用户使用。集成式应用程序并不简单。它们可以调用企业信息系统 (EIS) 上的应用程序，可以涉及跨部门或企业的业务流程，并可以采用本地或远程方式调用以各种语言编写并在各种操作系统上运行的应用程序。组件通过可视编辑器创建并汇编到

其他集成应用程序（即从一组组件创建的应用程序）中。可视编辑器在组件与其实现之间提供了一个抽象层。使用工具的开发者可以组装集成式应用程序，而不必详细了解各个组件的底层实现。

Integration Designer 工具基于面向服务的体系结构。组件是服务，包含许多组件的集成应用程序也是服务。创建的服务遵循业界一流的标准。BPEL 流程也成了组件，它们是使用符合业界标准的业务流程执行语言的易用可视工具以类似方式创建的。

在 Integration Designer 范式中，在模块中组装组件。使用导入和导出在模块之间共享数据。放到库中的工件可以在模块之间共享。

模块和库可以与流程应用程序相关联以便与 Process Center 配合使用，并可由 Process Designer 中创建的流程用作服务。在此类情况下，它们也可以随流程应用程序一起部署。

另外，模块和库也可以直接部署到测试环境或 Process Server。您可以使用调解模块创建调解流，并可将调解流部署到 WebSphere Enterprise Service Bus 或 Process Server。

IBM Integration Designer 还可用于创建能够部署到 WebSphere DataPower Appliance 上的数据类型和 XML 映射。另外，您还可以与 WebSphere DataPower 往来传输文件。

V7.5 业务流程管理产品的新增功能

在每种产品或组件各自的新增功能页面中了解 IBM 业务流程管理的新增功能。

新功能和增强功能着重于这些高层次业务流程管理需求:

- 改善业务流程的可视性
- 最大程度地提高复用率
- 管理更改
- 能够扩大项目的范围和规模

以下页面描述了 IBM Business Process Manager V7.5 和其他业务流程管理产品及组件的新功能。

- IBM Business Process Manager V7.5 的新增功能
- IBM Integration Designer V7.5 的新增功能
- WebSphere 支持的 Business Space V7.5 的新增功能
- IBM Business Monitor V7.5 的新增功能

IBM Business Process Manager V7.5 的新增功能

IBM Business Process Manager 将 WebSphere Process Server、WebSphere Lombardi Edition 和 IBM Integration Designer（原名 WebSphere Integration Developer）中的关键功能整合到了统一的用户环境中。它集成了这三种产品中的最新功能，并增强了先前产品版本中的功能。

IBM Business Process Manager 是综合性、可消耗的业务流程管理平台，提供业务流程的可视性和管理功能。它包括用于流程设计、执行、监视和优化的工具和运行时。它的设计目的还包括帮助流程所有者和业务用户直接参与业务流程的改进。

以下列表中的一些新功能和增强功能在某一产品（WebSphere Process Server、WebSphere Lombardi Edition 或 WebSphere Integration Developer）的先前版本中已经存在。但这些功能对于先前仅使用过一两种产品而未使用过全部三种产品的用户来说是新增的。

在所集成产品的早期版本中曾存在过的一些功能已经被 IBM Business Process Manager 中的新功能替换。要获取不推荐的功能列表，请参阅不推荐的功能。

IBM Business Process Manager V7.5 中的新功能和增强功能

IBM Business Process Manager V7.5 提供以下新功能或增强功能:

- 统一的编写环境，用于创建以人员为中心、基于标准的业务流程建模表示法 (BPMN) 流程（使用 Process Designer），以及将它们与以系统为中心的业务流程执行语言 (BPEL) 流程集成（使用 Integration Designer）。
- 用于 BPMN 流程以及 BPEL 流程的共享库和公共执行环境。统一的编写环境、共享库以及公共执行环境合在一起，提供了以下增强功能:
  - 对流程应用程序、服务、用户界面和规则的图形实现与测试。
  - 使用 BPMN 的基于标准的流程设计。
  - 用于构造服务、数据转换、BPEL 协调以及对应用程序和后端系统进行集成的 IBM Integration Designer 工具。
  - 业务规则编写，用可访问的方式表达业务逻辑，提供与 WebSphere ILOG JRules 相同的规则编写体验。可以将规则内容轻松导出到 WebSphere ILOG JRules，提供业务决策的起点。
  - 已添加到 IBM Business Process Manager V7.5 的 IBM ILOG JViews Chart 制图引擎。对于现有的定制图表，提供了从 CDL 格式转换为级联样式表 (CSS) 格式的工具。
  - 让用户能配置实现详细信息而不必为其编码的属性表，从而使技术水平较低的用户也能参与设计。
  - 全面的生命周期集成，使解决方案从模型设计到部署始终保持同步。
  - 迭代回放，使用户能随时验证流程需求。
  - 所有流程资产的共享库，用于拖放复用和协作实现。
- 通过这些功能可以实时控制进行中的流程:
  - IBM Process Portal 中的收件箱提供了所有未完成任务的合并视图。
  - 流程状态的图形视图帮助流程参与者了解流程中的剩余步骤。
  - 显式事件建模定义了工作流程和异常处理。
  - 公开的流程参数和阈值在进行处理期间提供实时控制。
  - 支持特别操作，指定的用户可以执行自发任务和修改工作流程。
  - 用户在流程执行期间可以就特定任务进行沟通和协作。
  - 实时记分板提供对进行中的工作的可视性以及必要时采取纠正行动的能力。
  - Business Space 用户界面的框架包含在 Business Process Manager 中。
  - 联合的任务视图使用户可以实时地执行任务、管理工作项、跟踪业绩以及响应事件。
  - 用户可以选择直接从 Microsoft Office 和 Microsoft SharePoint 执行流程任务。Microsoft 集成是以可选的附加组件形式提供的。
- 增强的监视功能，具有对人员和系统交互的流程可视性。IBM Business Process Manager 使用以下这些功能支持显著的动态处理可视性:
  - 业绩测量结果以关键业绩指标 (KPI) 和服务级别协议 (SLA) 形式表达。
  - Business Performance Data Warehouse 自动收集业绩事件和业务数据，并使其与建模的流程度量关联，以获得始终符合最新情况的流程可视性。
  - 流程优化器允许直接在流程模型图上对业绩瓶颈和其他问题进行可视化。
- WebSphere Process Server 中的功能现在已整合到 IBM Process Server 中，提供从第一批项目到企业范围内解决方案的事务完整性和增强的可调整性:
  - 通过嵌入式 WebSphere Application Server 提供了高度的可调节性和可用性。

  - 一台 BPM 运行时服务器 (Process Server) 支持各类业务流程、服务协调和集成。
  - 内置的面向服务的体系结构 (SOA) 组件包括灵活的企业服务总线 (ESB) 连接基础结构，用于集成应用程序和服务。
  - 受支持的标准包括 BPMN 和 BPEL 执行，以及许多数据和服务标准。
  - 集成适配器将应用程序和信息资产连接到 ESB，通过基于最佳实践、可快速部署并适合用于企业的连接加速业务集成项目。包含一套全面的功能以支持对资产的服务，包括打包、定制和旧应用程序、技术协议以及数据库。
  - 提供了丰富的维修和可恢复性功能，包括自动重试、手动维修、补偿和存储转发。
- 使用以下这些功能支持高水平的程序可管理性和监管:
  - Process Center 提供可调节的中央存储库和控制中心，用于组织和管理所有作为 BPM 程序的一部分创建的流程工件、应用程序和服务。
  - 工具箱管理流程工件，以便跨多个流程应用程序复用。
  - 通过快照，只需单击一次鼠标，就能捕获流程应用程序或工具箱中所有工件（包括业务流程图、规则、数据、表单、服务以及模拟场景）在特定时间点的状态。
  - 版本控制功能可恢复流程应用程序或工具箱的任何历史快照。
  - Process Server 注册表提供了集中的工具，用于跨各种运行时环境安装和跟踪多个流程的已部署版本。
  - 生命周期管理提供集中控制功能，用于管理流程和服务向生产运行时的部署。
- IBM Business Process Manager 提供三种产品配置（Advanced、Standard 和 Express）以配合公司对业务流程管理的切入点。有关详细信息，请参阅本信息中心中的“IBM Business Process Manager 7.5 配置”。
- IBM Business Process Manager 提供增强的编写工具和可选附加组件，用于和流行的最终用户生产力工具集成:
  - IBM Process Designer 基于标准的流程设计工具，是用于全部三种 IBM Business Process Manager 配置（Advanced、Standard 和 Express）的基本编写环境的一部分。该流程设计工具实现了快速组合和连续流程更改的功能。
  - IBM Integration Designer 高级配置编写环境，用于可视化构造服务、数据转换、BPEL 流程以及对应用程序及后端系统的集成。该编写环境与流程设计工具协同使用，可创建强大且可伸缩的流程解决方案。它包含一套全面的支持对资产服务的适配器，包括打包、定制和旧应用程序、技术协议以及数据库。该编写环境与 WebSphere Integration Developer 的最新版本完全兼容。
  - IBM Business Process Manager for Microsoft Office 附加组件是用于全部三种配置（Advanced、Standard 和 Express）的可选附加组件。Microsoft Office 用户可以使用 IBM Business Process Manager 直接从其 Office 桌面查看和运行 IBM BPM 任务。Office 附加组件会将 IBM 插件安装到最终用户 Office 桌面客户机上。
  - IBM Business Process Manager for Microsoft SharePoint 附加组件是用于全部三种配置（Advanced、Standard 和 Express）的可选附加组件。Microsoft SharePoint 用户可以使用 IBM Business Process Manager 直接从嵌入 SharePoint 门户网站页面中的 Web 部件查看和运行 IBM BPM 任务。SharePoint 附加组件可安装在任何将用于通过 IBM BPM Web 部件托管与流程的交互的 SharePoint 服务器上。
- IBM Business Process Manager V7.5 支持以下迁移和数据导入:
  - 将流程应用程序从 WebSphere Process Server V6.1、V6.1.2、V6.2 以及 V7.0 迁移，包括设计时流程定义资产和进行中的流程实例。
  - 将流程应用程序从 Lombardi Teamworks 6.1、Teamworks 6.2、Teamworks 7.0 以及 WebSphere Lombardi Edition V7.1 和 V7.2 迁移，包括设计时流程定义资产和进行中的流程实例。
  - 从 WebSphere Business Modeler V7.0、WebSphere Business Compass V7.0 和其他流行的建模工具导入 BPMN 2.0 格式的设计时流程模型。

– 用于 Business Space 的 IBM BPM Advanced 窗口小部件可以与 IBM WebSphere Portal 配合使用。

WebSphere 支持的 Business Space 的新增功能

Business Space 是 Business Process Manager 和其他 IBM 产品的公共组件，已经过增强:

- WebSphere 支持的 Business Space V7.5 的新功能
- Business Space V7.5 for WebSphere Portal V7.0 的新增功能

IBM Business Process Manager 中的辅助功能选项

辅助功能可以帮助残疾用户（如行动不便或弱视用户）成功使用信息技术产品。

IBM 力求为不同年龄或不同能力的用户提供可操作的产品。通过辅助技术（例如，屏幕朗读器软件和数字发声合成器）可以使用屏幕上显示的内容。查阅辅助技术的产品文档以了解有关将这些技术与本产品配合使用的详细信息。

可使用键盘（而不是鼠标）来操作功能。

可以定制显示属性，例如颜色、对比度和字体大小。

可放大图形视图中显示的信息以获取更多详细信息。

可以在 IBM Web 站点 (http://www.ibm.com/able/product_accessibility/index.html)上申请符合美国联邦政府康复法案第 508 节规定条款的“志愿产品辅助功能模板”(VPAT)。

信息中心记录包括以下附加功能以帮助您使用辅助功能:

- 此文档以 HTML 格式提供，以帮助用户应用屏幕朗读器软件技术。
- 此文档中的图像提供时附带替换文本，以便有弱视用户可使用图像的内容。

IBM Business Process Manager 中本地语言的可用性

IBM Business Process Manager 支持以下语言。文档可能未进行全面翻译。

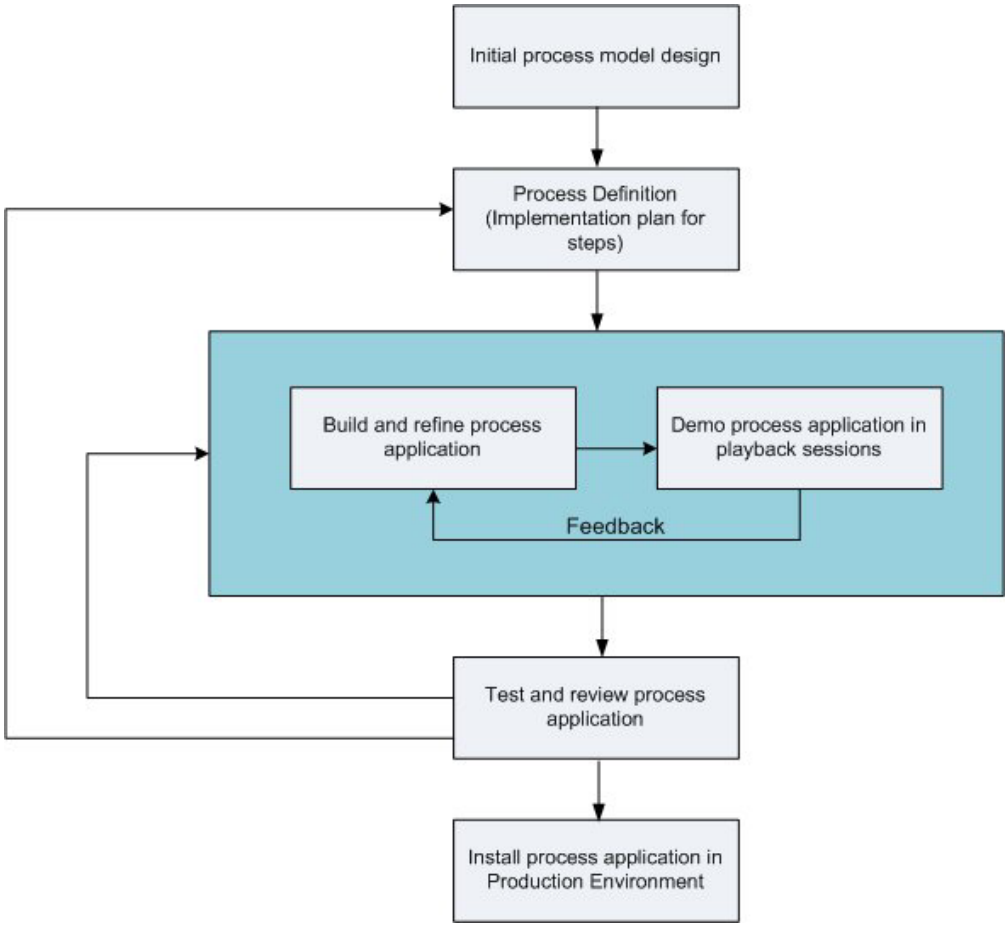
- 简体中文
- 繁体中文
- 捷克语
- 美国英语
- 法语
- 德语
- 匈牙利语
- 意大利语
- 日语
- 韩国语
- 波兰语
- 巴西葡萄牙语
- 罗马尼亚语
- 俄语
- 西班牙语

业务流程管理概述

在 Process Designer 中开发流程时，您需要在测试和生产环境的服务器上规划流程应用程序的最终安装。

下图说明典型流程开发工作的生命周期。这包括用于构建和优化安装服务的步骤，以使您能够在生产环境中安装流程应用程序。

如图所示，您可以只在开发环境中工作。但是，您需要针对测试环境和生产环境配置 Process Server。



流程建模概述

流程是 IBM Business Process Manager 中的主要逻辑单元。它是流程定义的所有组件（包括服务、活动和网关；计时器、消息和异常事件；顺序线、规则和变量）的容器。对流程建模时，您将创建可重复使用的业务流程定义 (BPD)。

流程组件允许您为最终用户定义工作流程，在流程内创建逻辑并与其他应用程序和数据源集成。要了解运行时流程内发生了什么，应了解设计时构成流程的组件。

在 IBM BPM 中构建流程

使用 IBM BPM 开发流程时通常涉及到各个组织的许多不同的人员。最大挑战是确保您要构建的最佳可行解决方案符合项目的声明目标。为了确保获得成功的结果，团队成员应该协作以捕获流程需求并反复开发模型及其实现。

重复使用 Process Designer 中的项

Process Designer 使流程开发者能够在流程应用程序内部和流程应用程序之间重复使用现有项。例如，如果知道已存在一些包括 Coach 以及您和其他开发者需要的其他共享项的服务，那么可通过将这些项添加到工具箱来访问和复用这些项。然后，在流程应用程序中，可以向这些共享项所在的工具箱添加依赖关系。这允许您在选择活动的实现时选择现有服务之一。工具箱中的项也可供在不同流程应用程序中工作的其他开发者使用。

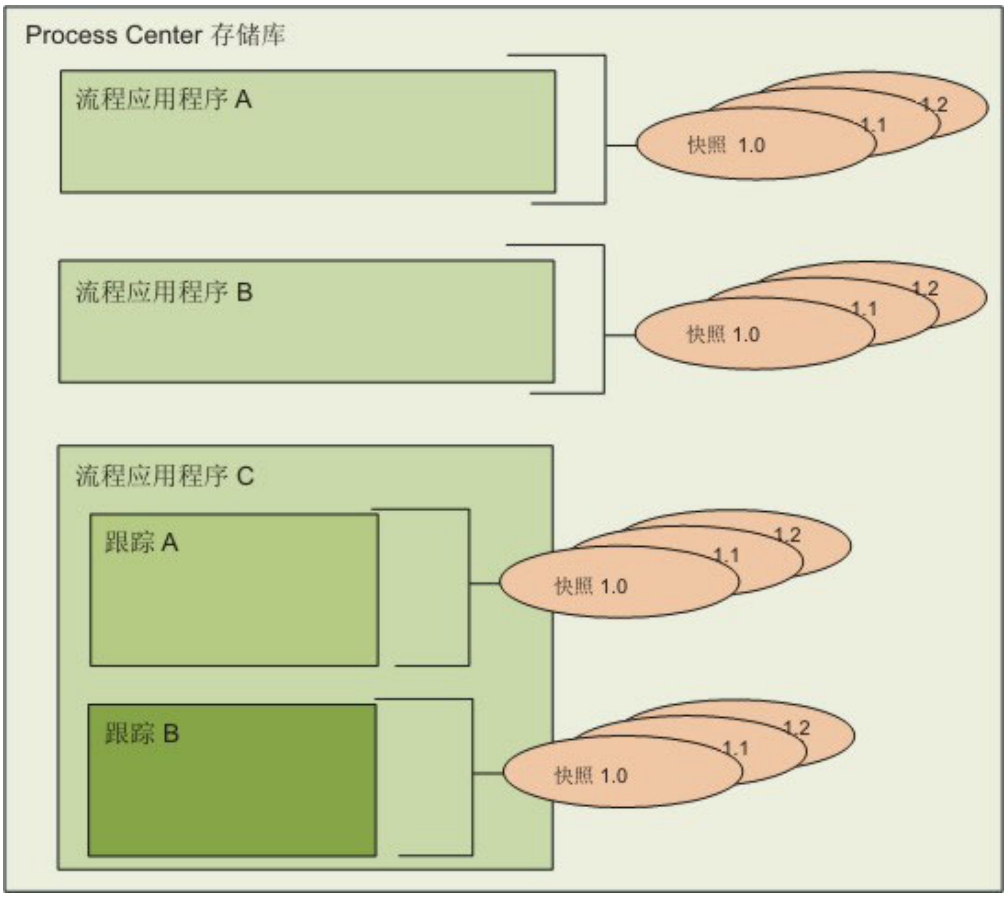
在 IBM Process Designer 中使用 Designer

Designer 界面提供在 IBM BPM 中对流程建模所需要的工具。

使用 Process Center 进行的流程开发

IBM Process Center 充当在 Process Designer 中创建的所有项目资产的中央存储库。当多个 Process Designer 客户机连接至 Process Center 时，用户可共享某些项（例如，流程和服务），还可以即时查看其他用户所作的更改。Process Center 也可用作在 IBM Integration Designer 中创建的资产的存储库。

当您在 Process Designer 中开发流程时，Process Center 存储库中提供了层次结构，此层次结构旨在帮助帮助您管理项目。下图提供了此存储库层次结构的概念性概览:



如您在上图中所见，Process Center 存储库包括下列工件:

流程应用程序	容器，用于保存 BPM 分析人员和开发者在 IBM Process Designer 的 Designer 中创建的流程模型和支持实现。

跟踪	流程应用程序中基于团队任务或流程应用程序版本的可选子部分。启用时，跟踪允许进行并行开发。管理员会确保是否需要对每个流程应用程序启用其他跟踪。
快照	在特定时间点记录流程应用程序或跟踪中的各项的状态。通常快照表示里程碑或者用于回放或安装。可比较快照并还原至先前快照。如果管理员对流程应用程序启用跟踪，那么会将快照用作新跟踪的基础。

流程应用程序: 概述

流程应用程序是流程模型及其支持实现的容器；它存储在存储库中。在编写工件或者以其他方式创建工件之后，它们将组装成流程应用程序，并且将创建快照。您可以测试、安装和管理流程应用程序快照。

流程应用程序包含下列某些或全部工件:

- 一个或多个流程模型（也称为业务流程定义 (BPD)）
- 实现活动或者与其他系统集成所需的服务
- 服务组件体系结构 (SCA) 模块
- 工具箱
- IBM Integration Designer 库
- 一个或多个工作空间
- 用于监视业务性能的 IBM Business Monitor 模型
- 运行流程所需的任何其他项

流程应用程序和业务级应用程序

每个流程应用程序都有一个业务级应用程序 (BLA)，后者充当流程应用程序及其资产的容器（资产包括监控模型、SCA 模块、工具箱和库之类的内容）。另外，流程应用程序的每个快照都有自己的 BLA。快照的许多管理任务（例如在生产服务器上停止或启动快照）都在 BLA 级别完成，从而使您能够更快更方便地管理快照及其所有资产。

使用 Inspector 运行和调试流程

IBM Process Designer 中的 Inspector 是迭代式流程开发方法的关键。使用 Inspector，各个开发人员可以在 Process Center Server 或远程运行时 Process Server 上运行流程和服务。另外，整个开发团队可以使用 Inspector 在回放会话中演示当前流程设计和实现。回放会话有助于从流程中不同的干系人（例如管理者、最终用户和业务分析人员）处捕获重要信息。采用迭代式流程开发方法确保了您的流程应用程序满足所有干系人的目标和需求。

IBM Process Designer 中的 Inspector 包含多个工具，这些工具使您能够在每个已配置的环境中完成各项任务，例如:

任务	描述
管理流程实例	运行流程时，可以查看环境中 IBM Business Process Manager 服务器上先前已执行和当前正在运行的所有实例。您可以管理正在运行的实例，例如，将它们暂停然后继续。还可以通过过滤或删除特定的记录来管理先前执行的实例。
分步执行和调试流程	对于选定的实例，请查看当前正在执行的步骤，然后执行流程中的后续步骤，从而分步执行流程。流程的树形显示与流程图中称为标记的指示符相组合，使您轻松地了解您处在流程中的什么位置。另一个优势是您可以查看每个步骤中使用的变量以及对应的值（如果适用。）

请参阅以下主题以了解关于使用 Inspector 界面的更多信息:

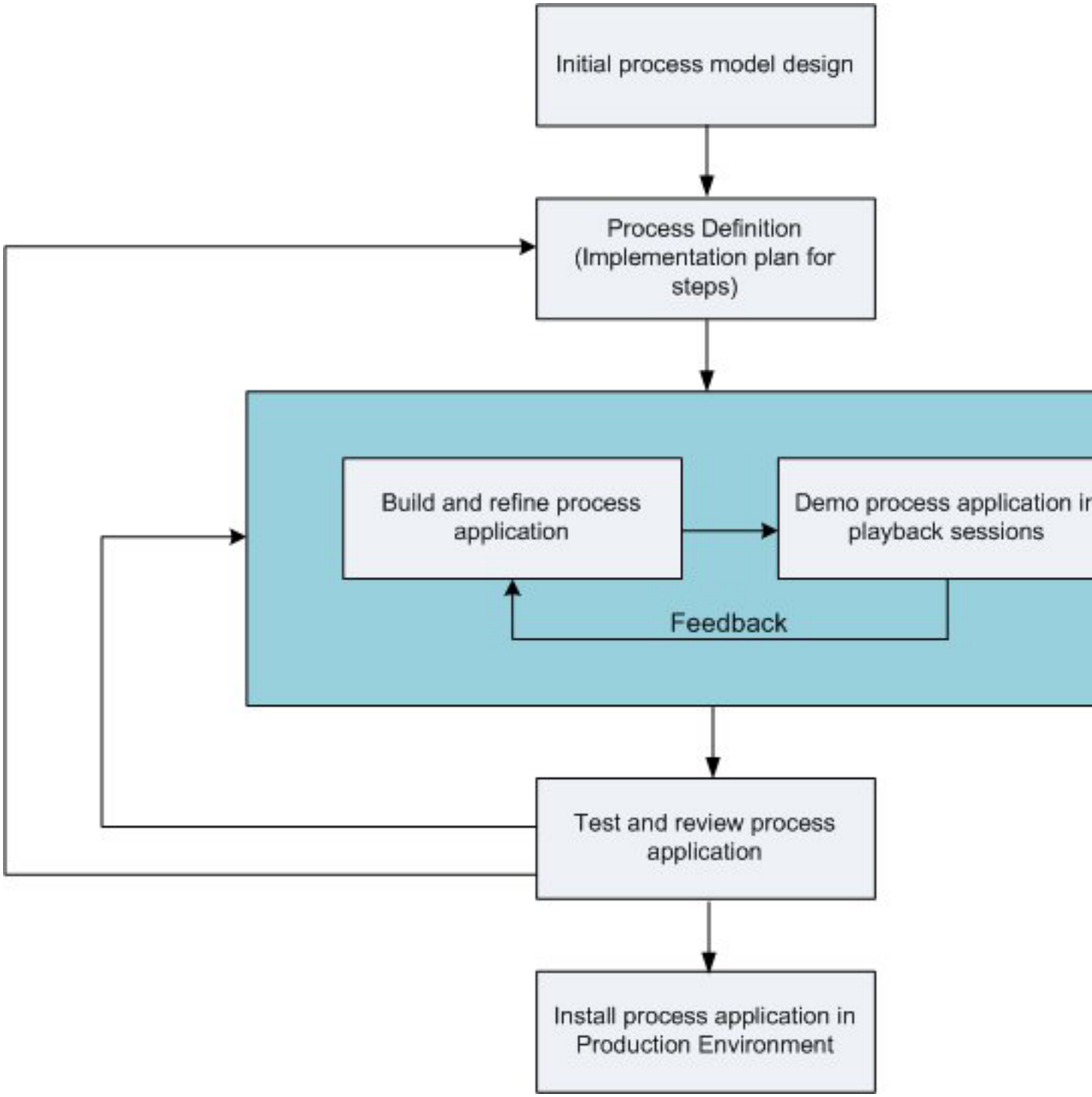
要了解如何...	请参阅...
管理您选择的服务器上当前和先前正在运行的流程实例。	管理流程实例
分步执行流程以确保您的 BPD 如预期的那样运作。	分步执行流程
检验流程执行中每个步骤内的每个底层流程或服务，从而执行比分步执行流程更详细的检查。	调试流程
轻松地找到运行 BPD 时产生错误的根源并将其解决。	解决错误
访问和使用 Inspector 提供的各个功能部件。	Inspector 引用

部署和管理流程应用程序

流程应用程序生命周期包括部署和取消部署快照以及管理快照状态。版本控制注意事项也是生命周期的组成部分。

开发流程时，您可以充分利用 Process Designer 中的工具所支持的迭代方法。流程将随时间推移不断演进，从最初的开发状态到测试状态，然后再到生产状态。即使在生产环境中，流程也可能由于需求不断变化而继续演进。为持续的流程生命周期做好准备非常重要，这有助于您从一开始就有效地进行设计。

下图演示迭代式流程开发方法:



典型的 Business Process Manager 配置包括三个环境以支持开发并最终部署流程。

环境	描述
开发	在 IBM Process Designer 中构建并优化流程应用程序。创建流程模型并使用 Designer 在这些模型中实现各个步骤。通过使用 Inspector，在回放会话中演示开发进度，以便您能够快速评估和优化原型。通过使用 Process Center Console，将流程应用程序部署在测试和生产环境中。

环境	描述
测试	通过使用 Process Center Console，将流程应用程序部署在测试环境中的 Process Server 上以实现正式的质量保证测试。您可以使用 Inspector 以帮助您验证和解决问题。
生产	解决正式测试所报告的所有问题后，使用 Process Center Console 将流程应用程序部署在生产环境中的 Process Server 上。您可以使用 Inspector 以调查和解决在生产环境中报告的任何问题。

发布和安装策略

为了确保您实现并部署的流程应用程序符合贵公司的质量标准，请考虑定义发布和安装策略。当您确定了新的以及已更新的流程应用程序的发布和安装目标及需求后，可以使审核和启动计划的流程自动化。

例如，您可能需要跨贵公司不同的汇报结构将流程发送给几位不同的经理。只有当每位经理都在新流程或已更新的流程上签名后，才能将此流程部署在生产环境中并展示给最终用户。您可以在 IBM Business Process Manager Advanced 中创建并实现此类复审所涉及的步骤，以确保满足所有企业准则并且获得所需的签名。复审的最后一步可能是通知 IT 团队经审核的流程应用程序已可以部署。

下面几节为您提供关于部署流程和在部署后管理它们的信息。

创建、访问和合并服务

访问应用程序的外部服务

本场景讨论用于访问应用程序的外部服务的不同方法，并提供用于访问这些外部服务的高级别任务。

注: 本场景适用于 WebSphere® Enterprise Service Bus 和 IBM Business Process Manager。调解模块可以部署到 WebSphere Enterprise Service Bus 和 IBM Business Process Manager。模块可以部署到 IBM Business Process Manager。

在集成式业务应用程序中，*业务服务*相互交互以提供所需的功能。业务服务执行有助于实现业务目标的可复用功能或任务。但是，找到服务并连接到此服务的工作与业务功能无关。将业务功能与用于管理服务连接的任务分隔开来为解决方案提供了灵活性。

当*服务请求者*向*服务提供者*发送执行某个业务功能的请求时，服务交互就开始。此请求以*消息*的形式发送，该消息定义要执行的功能。服务提供者执行所请求的功能并将结果包含在消息中发送给服务请求者。通常，需要对消息进行处理，以便允许服务交换数据并实现其他与业务功能和数据无关的低级别 IT 功能。例如，路由、协议转换、变换、重试失败的调用和动态服务调用。这种处理称为*调解*。

IBM Integration Designer 中有两种类型的模块: 专门设计用来包含业务逻辑（例如业务流程、业务规则和业务状态机）的模块（或业务集成模块）以及用于实现调解流的调解模块。虽然这两种类型的模块在功能方面有些重复，但是，通常建议在业务模块中隔离业务逻辑并且由调解模块执行调解逻辑。

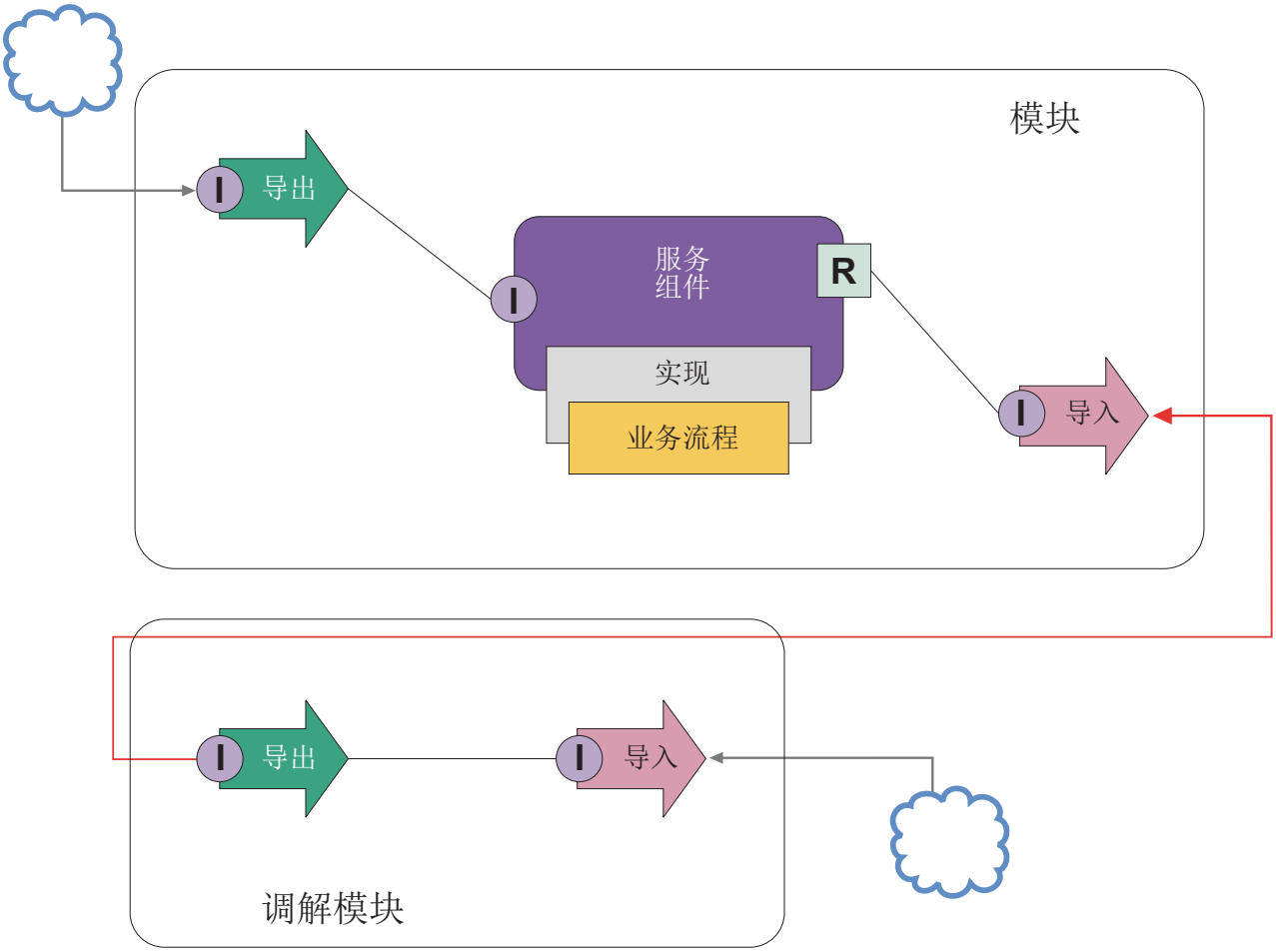
但是，业务逻辑和调解逻辑之间并非总有清晰的界限。在没有清晰界限的情况下，请考虑需要在服务调用之间处理的状态数量或变量中的数据数量。通常，如果只需进行很少状态处理或者不需要进行状态处理，请考虑使用调解流组件。如果需要存储服务调用之间的状态，或者如果有需要存储在变量中并进行处理的数据，请考虑使用业务流程组件。例如，如果您正在调用多个服务并且正在记录每个服务所返回的信息，以便在调用所有服务后，您可以对所返回的数据执行一些进一步处理，请使用可以很容易将所返回的信息指定给变量的业务流程。也就是说，如果有太多状态，那么就从调解逻辑转变为业务逻辑。

没有一个集成方案，并且技术上没有错误的答案。此处所讨论的准则是一些支持灵活性和复用的较好实践，提供给您供您考虑。像平常一样，您应该仔细考虑对您的业务集成应用程序实现这些模式的优点和缺点。让我们考虑一些情况。

访问 SCA 组件

导入不需要进行任何数据变换就访问另一个 SCA 组件是一个基本的访问服务示例。即使在这种情况下，您也可以从调解模块访问外部服务，而不是直接从业务模块进行访问。这就为将来提供了灵活性，使得可以更改服务端点、服务质量或管理（例如添加日志记录），而不会影响使用该服务的业务组件。这种体系结构模式称为“关注点分离”。

在您决定实现此模式之前，请针对另一个模块增加的开销所带来的潜在影响权衡这种模式的优势。如果您的主要需要是灵活性，并且要频繁更改所访问的服务，请考虑按此处所示的方式使用另一个模块。如果性能最为重要，并且您愿意更新并重新部署业务逻辑，请考虑使用单一模块。



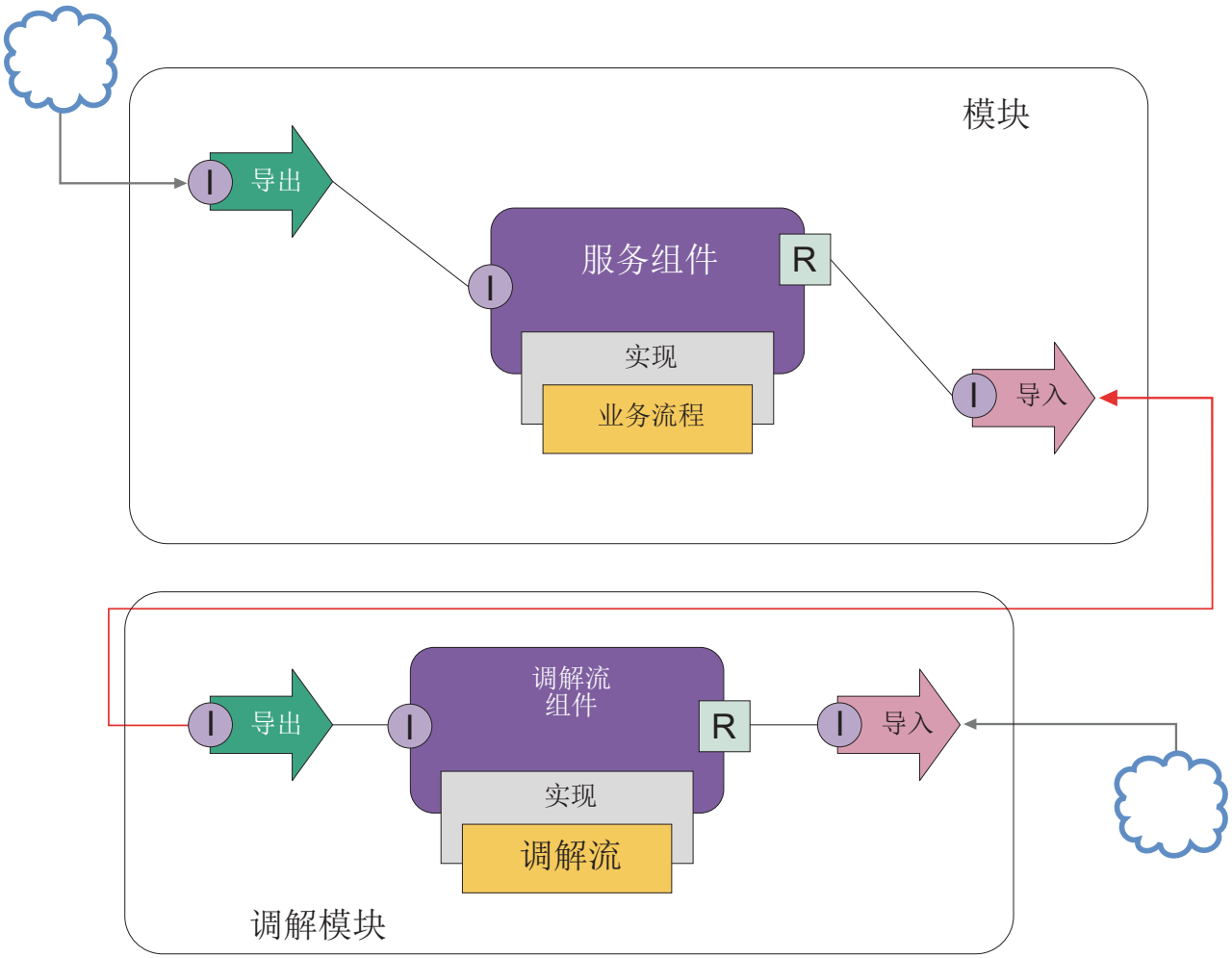
以下是实现此示例所需执行的高级别任务。

1. 创建一个调解模块。有关逐步骤指示信息，请参阅创建调解模块。
2. 在调解模块中，为要访问的外部服务创建一个具有合适绑定的导入。有关逐步骤指示信息，请参阅创建导入。有关绑定的更多信息，请参阅绑定。
3. 创建一个导出并为其提供与导入相同的接口。有关逐步骤指示信息，请参阅创建导出。
4. 为该导出生成 SCA 绑定。有关逐步骤指示信息，请参阅生成 SCA 绑定。
5. 在该调解模块的组合件中，将导出连接到导入。保存该调解模块。
6. 创建一个模块。有关逐步骤指示信息，请参阅为业务服务创建模块。
7. 添加导出和组件。
8. 在“业务集成”视图中，将步骤 4 中在调解模块中创建的导出拖到模块组合件中。这将创建一个与该导出具有相同绑定的导入。
9. 将导出连接到组件，然后将组件连接到导入。
10. 添加组件的实现。有关实现类型的信息，请参阅实现。

以后，您可以向该调解模块添加日志记录或路由之类的调解逻辑，而不会影响业务模块。

添加调解

有时仅调用外部服务还不够。有时，您需要先在服务请求者与提供者之间添加一个调解模块作为中介来进行处理。



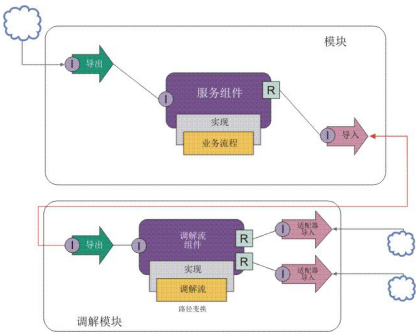
以下是中介调解流将执行的一些功能:

- 设置协议头。有关更多信息，请参阅 WebSphere Enterprise Service Bus 信息中心中的协议转换主题。
- 通过使用“业务对象映射”或“XSL 变换”原语进行接口或参数变换。变换消息。
- 通过使用“消息过滤器”原语从静态列表中选择特定服务。消息过滤器。
- 通过使用“扇出”和“扇入”原语调用多个服务以聚集结果。聚集和广播消息。
- 通过使用“服务调用”原语重试同一服务或调用另一个服务来处理服务调用失败。重试失败的服务调用。
- 通过在运行时而不是集成时选择要使用的服务来进行动态路由选择，这使得服务可以更松散地耦合并且业务能够更快速地对更改作出反应。可以添加新服务，而无需使用已部署到运行时环境的模块。动态路由选择在与注册表配合使用时功能最强大（注册表要求使用“端点查找”调解原语）。以动态方式选择端点。

访问企业信息系统

外部系统中的服务和工件可以导入到 Integration Designer 中。向导将发现企业信息系统 (EIS) 中的应用程序和数据并允许您根据所发现的应用程序和数据来生成服务。所生成的工件是接口和业务对象，这些接口和业务对象可由模块中的组件使用。

在模块与主机系统之间使用中介调解模块增强了可复用性。在下面的示例中，调解流用于路由至正确的主机系统，并用于将数据转换成主机系统所需的格式。



以下是此示例的高级别任务:

1. 使用外部服务向导连接至主机系统。通过使用外部服务向导来访问外部服务遵循类似的模式，这与所使用的适配器无关。有关如何使用外部服务向导的信息，请参阅使用适配器访问外部服务的模式。
2. 创建一个模块。有关逐步骤指示信息，请参阅为业务服务创建模块。
3. 添加一个导出、一个组件和一个具有 SCA 绑定的导入。有关更多信息，请参阅调用服务。
4. 向该导出添加接口，然后将该导出连接到组件。
5. 添加组件的实现。在实现中，设置一个指示将访问的主机服务的属性。有关实现类型的信息，请参阅实现。
6. 创建一个这样的调解模块: 该调解模块的导出具有 SCA 绑定，并且该导出的接口是步骤 2 中创建的模块的导入所具有的接口。
7. 将该导出连接到调解流组件。
8. 通过使用组合件编辑器选用板中的合适出站适配器为要访问的每个主机系统创建一个导入。
9. 将调解流组件连接至这些导入。
10. 实现调解流组件。使用“消息过滤器”原语来根据业务逻辑中设置的属性选择导入，并对每个适配器导入使用 XSL 变换原语。消息过滤器。
11. 在模块中，选择调解模块的导出作为要导入到模块中的服务。有关分步信息，请参阅调用另一个模块中的服务。

以后，您就可以进行添加适配器或更改适配器以指向另一个主机系统之类的更改，并同时确保对业务逻辑的影响降至最低。

访问消息传递系统

为了使服务组件体系结构 (SCA) 模块能够与现有 JMS、MQ 或 MQ JMS 消息传递客户机通信，您需要为导入和导出创建接口、业务对象和绑定。请参阅将消息映射至 SCA 接口。

调解流使用不仅提供对业务对象的访问、还提供对上下文和头信息的访问的消息。如果要访问 JMS 头信息或定制 JMS 属性，请使用调解流。如果正在与 MQ 系统集成并且要访问 MQ 头信息，请使用调解流。

创建或调用 Web Service

Web Service 是自包含应用程序，用于执行从简单查询到复杂业务流程交互的业务功能。您可以调用现有 Web Service，也可以开发满足自己需求的新 Web Service。本场景将描述步骤并为您提供更多信息。

虽然您可能未使用 IBM Integration Designer 从头开始创建所有服务，但确实有一些服务需要用这种方式创建。使用组合件编辑器和业务流程编辑器将服务组装成业务流程时，您可能会发现缺少某些服务。因此，使用 IBM Integration Designer 工具创建这些缺少的服务可能非常有用。反之亦然，在创建新流程后，您可能会认为将所有或一部分流程操作作为服务公开以供其他人使用非常有用。

注: 本场景适用于 IBM Integration Designer for IBM Process Server 和 WebSphere Enterprise Service Bus 用户。

使用 IBM Integration Designer 开发 Web Service 是基于以下几点原因:

- 在 IBM Integration Designer 中创建服务允许您使用业务规则实现服务。
- 在 IBM Integration Designer 在进行开发允许您开发 Java 服务并将此服务同时作为 Web Service 和 SCA 服务公开。
- 不必对接口映射进行编码是一项优势。您可以从 Java 代码中剔除所有数据映射，为 Java 开发者留下一个简单的黑匣 Java 程序。
- IBM Integration Designer 将所有服务和关系显示在一个位置。
- 重构功能也有助于使用 IBM Integration Designer 开发 Web Service。

请记住，不应该将 Web Service 视为所有集成问题的解决方案。但是，正像使用任何其他技术或体系结构方法一样，在正确的位置和正确的时间使用 Web Service 具有先天优势。

导出、导入和绑定

IBM Integration Designer 允许您导入标准 Web Service 并在组合应用程序中使用这些服务。

在 IBM Integration Designer 中，您可以使用组合件编辑器来开发服务。请遵循创建模块、调解模块、库和组件的标准过程。然后，可以使用导出、导入和绑定来共享和访问这些服务。下面列示了这些基本任务的步骤以及指向有关每个任务的更详细信息的链接。

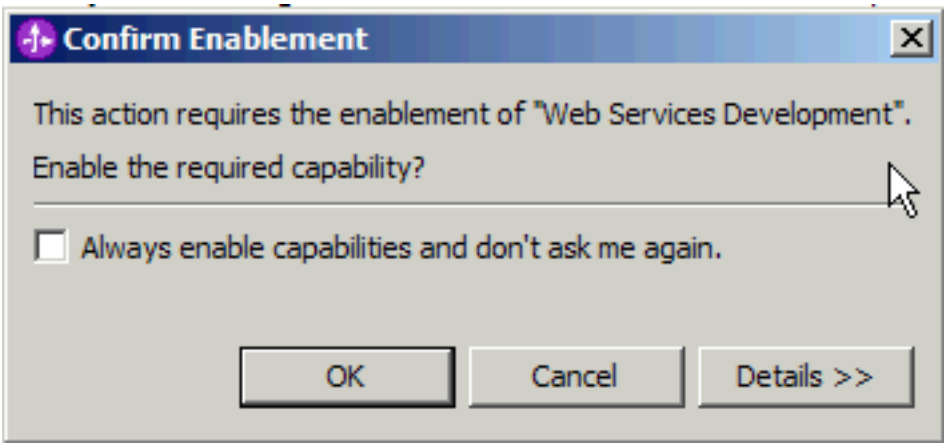
您可以对 Web Service 使用以下两个绑定中的任一绑定: Web Service 绑定或 HTTP 绑定。Web Service 绑定提供了用于将消息传输到 Web Service 以及传输来自 Web Service 的消息的规范。工具将帮助您自动生成 Web Service 绑定。HTTP 绑定客户机与服务器之间的标准请求-响应协议，正如万维网联盟 (W3C) 发布的 HTTP 协议所定义的那样。如果使用 HTTP 绑定，那么您需要提供一些最初的绑定配置信息。

1. 创建导出以发布模块的服务以供其他模块使用。
2. 为该导出生成绑定。
   - 为导出生成 Web Service 绑定。
   - 生成 HTTP 导出绑定。
3. 创建导入以调用不属于正在组装的模块的现有服务。
   - 为导入生成 Web Service 绑定。
   - 生成 HTTP 导入绑定。

如果您想要从 JavaServer Pages 调用 Web Service，请阅读链接的主题。

Web Service 开发能力

打开一个与 Web Service 创建过程相关联的编辑器时，您可能会遇到以下错误:



IBM Integration Designer 提供了一项称为*能力*的过滤功能。在“首选项”设置中，功能和工具归类到能力中，并且您可以启用和禁用能力类别或者任何类别的功能子集。有关更多信息，请参阅能力。

第 2 章 了解有关 IBM Business Process Manager 的更多信息

使用本部分作为考察 IBM Business Process Manager 所用技术的起点

版本控制

流程应用程序的生命周期从流程应用程序的创建开始，历经对该流程应用程序进行更新、部署、联合部署、取消部署和归档的整个过程。*版本控制*是一种机制，此机制通过唯一地标识流程应用程序的各个版本来管理流程应用程序的生命周期。

版本控制功能在 IBM Business Process Manager 中的工作方式取决于您正在部署的内容 - 从 IBM Process Center 中的存储库部署的流程应用程序或者直接从 IBM Integration Designer 部署的企业应用程序。

在缺省情况下，从 Process Center 部署到运行时环境的流程应用程序和工具箱已版本化。对于企业应用程序，您可以在 IBM Integration Designer 中选择对模块和库进行版本控制。

另外，还可以创建人员任务或状态机的版本，以便该任务或状态机的多个版本可以在运行时环境中共存。

对流程应用程序进行版本控制

版本控制功能使运行时环境能够在流程应用程序的生命周期内标识快照并能够以并发方式同时运行多个快照。

请将流程应用程序想像成容器。所有快照、部署和版本控制都在容器级别进行处理，而不是在容器内的工件级别进行处理。请从 Process Center 控制台中管理快照。

更改将在工作跟踪的 TIP 处动态保存到 Process Center 存储库。流程应用程序将保持处于 TIP 级别，直到您决定创建快照 (sn1) 为止。您可以将流程应用程序快照部署到流程中心服务器或流程服务器以进行测试、登台或生产。

如果您进行了更改并想部署新版本，那么需要创建新快照 (sn2)。您可以除去 sn1，也可以使其在您部署 sn2 时继续在服务器上运行。

版本上下文

版本上下文是用于标识版本的元数据。此标识由您指定，但 IBM 建议使用格式为 <major>.<minor>.<service> 的三位数版本系统。有关此版本控制方案的更详细描述，请参阅关于命名约定的主题。

IBM Business Process Manager 将对每个流程应用程序指定全局名称空间。明确而言，全局名称空间或者是流程应用程序的 TIP，或者是特定的流程应用程序快照。服务器使用的版本名称限长 7 个字符，因此，指定的名称将是使用了您指定的快照名称中的字符的首字母缩写词。如果快照名称遵循建议的 IBM VRM 样式并且长度不超过 7 个字符，那么快照首字母缩写词与其快照名称完全相同。例如，快照名称 1.0.0 的首字母缩写词为 1.0.0，而快照名称 10.3.0 的首字母缩写词为 10.3.0。快照首字母缩写词可以保证在 Process Center 服务器的作用域中的流程应用程序上下文中唯一。因此，您无法对快照首字母缩写词进行编辑。

控制 Process Designer 流程应用程序和工具箱的版本

要对 Process Center 存储库中存储的流程应用程序和工具箱进行版本控制，可保存并命名快照。这样做允许您比较两个快照以找出不同点。例如，如果某一开发者修正了一个服务问题并且在该时间点生成了其所包含的流

程应用程序或工具箱的快照，然后另一开发者对同一服务又进行了一些更改并且生成了新的快照，那么项目经理可比较这两个快照来确定更改的内容、时间和执行人员。如果项目经理决定对服务进行的其他更改没有价值，那么他可还原至原始修订的快照。

通常您应在每次准备或可能准备部署到生产或测试集成时生成流程应用程序快照。要部署到独立流程服务器，您必须生成流程应用程序的快照。对于工具箱则略有不同：您应在准备让工具箱供流程应用程序使用时生成该工具箱的快照。此后，如果要更新工具箱，您必须在准备更新时生成另一个“提示”快照，然后流程应用程序和工具箱的所有者就可确定是否要该用新快照。提示是一种特殊的快照，也是唯一可让您更改其中内容的快照，但您只能在 Process Center Server 上运行它。您不能将提示部署到 Process Server。

多个集群中的流程应用程序

您可以将同一版本的流程应用程序部署到同一单元内的多个集群。要区分同一版本流程应用程序的多个部署，请为每个部署创建一个快照，并在快照名称中包含在单元中唯一的标识（例如，v1.0_cell1_1 和 v1.0_cell1_2）。严格说来，每个快照都是该流程应用程序的一个新版本（从纯粹的生命周期管理角度而言），但其内容和功能都是相同的。

当您将流程应用程序部署到集群时，就会执行节点的自动同步。

对模块和库进行版本控制

如果模块或库位于流程应用程序或工具箱中，那么它将具有该流程应用程序或工具箱的生命周期（版本、快照和跟踪等等）。模块名和库名必须在流程应用程序或工具箱的作用域内唯一。

本主题描述与流程应用程序配合使用的模块和库的版本控制。但是，请注意，如果将模块直接从 IBM Integration Designer 部署到 Process Server，那么可以继续遵循『创建版本化模块和库』中描述的在部署期间对模块指定版本号的过程。

与 IBM Process Center 相关联的模块或库所依赖的库必须在同一流程应用程序或所依赖的工具箱中。

下表列示当一个库与流程应用程序或工具箱相关联时，您可以在 IBM Integration Designer 中的依赖关系编辑器中进行的选择:

表 3. 模块、流程应用程序或工具箱以及全局库的依赖关系

库作用域	描述	可以依赖于 . . .
模块	对于每个使用这个库的模块，服务器上都存在这个库的一个副本。	具有模块作用域的库可以依赖于所有类型的库。
流程应用程序或工具箱	这个库在流程应用程序或工具箱的作用域中的所有模块之间共享。如果通过 IBM Process Center 完成部署，那么此设置将生效。如果在 IBM Process Center 外部进行部署，那么这个库将复制到每个模块中。 **注**: 缺省情况下，在 IBM Integration Designer V7.5 中创建的库的共享级别为**流程应用程序或工具箱**。	此类型的库只能依赖于全局库。
全局	这个库在所有正在运行的模块之间共享。	全局库只能依赖于其他全局库。 **注**: 为了部署全局库，必须配置 WebSphere 共享库。有关更多信息，请参阅“模块和库依赖关系”。

命名约定

命名约定用于在流程应用程序的生命周期（更新、部署、联合部署、取消部署和归档）内对其各个版本进行区分。

此部分为您提供用于唯一地标识流程应用程序的版本的约定。

*版本上下文*是唯一地描述流程应用程序或工具箱的首字母缩写词的组合。每种类型的首字母缩写词都有命名约定。首字母缩写词限长 7 个字符，并且只能包含 [A-Z0-9_] 字符集中的字符（快照首字母缩写词除外，它还可以包含句点）。

- 流程应用程序首字母缩写词是在创建流程应用程序时创建的。其长度最多为 7 个字符。
- 快照首字母缩写词是在创建快照时自动创建的。其长度最多为 7 个字符。

  如果快照名称符合有效快照首字母缩写词的条件，那么快照名称和首字母缩写词将相同。

  注: 使用调解流组件版本感知路由功能时，请对快照进行命名，以使其遵从 *<version>.<release>.<modification>* 方案（例如 **1.0.0**）。由于快照首字母缩写词限长 7 个字符，所以数字值限制为总位数最多为 5 位（五个数字加两个句点）。因此，当数字字段递增时，请务必小心，因为前 7 个字符以外的任何内容都将被截断。

  例如，如果快照名称为 **11.22.33**，那么快照首字母缩写词将为 **11.22.3**。
- 跟踪首字母缩写词将根据跟踪名称中每个词的第一个字符自动生成。例如，对于使用名称 **My New Track** 创建的新跟踪，首字母缩写词的值将为 **MNT**。

  缺省跟踪名称和首字母缩写词为 **Main**。如果跟踪首字母缩写词不是 **Main**，那么以 IBM Process Center 服务器为目标的部署将在版本控制上下文中包括跟踪首字母缩写词。

流程应用程序中的业务流程定义通常由流程应用程序名称首字母缩写词、快照首字母缩写词和业务流程定义的名称标识。请尽可能为业务流程定义选择唯一名称。如果名称重复，那么您可能会遇到下列问题:

- 如果不进行某种形式的调解，您可能无法将业务流程定义作为 Web Service 公开。
- 您可能无法从在 IBM Integration Designer 中创建的 BPEL 流程调用在 IBM Process Designer 中创建的业务流程定义。

版本上下文随流程应用程序的部署方式不同而异。

Process Center 服务器部署命名约定

在 IBM Process Center 服务器上，可以部署流程应用程序的快照以及工具箱的快照。此外，还可以部署流程应用程序的提示或工具箱的提示。（*TIP*是流程应用程序或工具箱的当前工作版本。）版本上下文随部署类型不同而异。

对于流程应用程序，流程应用程序 TIP 或特定流程应用程序快照用于唯一地标识版本。

工具箱可以随一个或多个流程应用程序一起部署，但每个工具箱的生命周期限定为流程应用程序的生命周期。每个流程应用程序都有其自己的依赖工具箱或部署到服务器的工具箱的副本。已部署的工具箱不在各流程应用程序之间共享。

如果与流程应用程序相关联的跟踪命名为除缺省值 **Main** 外的其他名称，那么跟踪首字母缩写词也是版本上下文的组成部分。

流程应用程序快照

对于流程应用程序快照部署，版本上下文是下列各项的组合:

- 流程应用程序名称首字母缩写词
- 流程应用程序跟踪首字母缩写词（如果使用除 **Main** 外的跟踪）
- 流程应用程序快照首字母缩写词

独立工具箱

对于工具箱快照部署，版本上下文是下列各项的组合:

- 工具箱名称首字母缩写词
- 工具箱跟踪首字母缩写词（如果使用除 **Main** 外的跟踪）
- 工具箱快照首字母缩写词

TIP

在 Process Designer 中进行迭代测试期间将使用流程应用程序 TIP。它们只能部署到 Process Center 服务器。

对于流程应用程序 TIP 部署，版本上下文是下列各项的组合:

- 流程应用程序名称首字母缩写词
- 流程应用程序跟踪首字母缩写词（如果使用除 **Main** 外的跟踪）
- “Tip”

在 Process Designer 中进行迭代测试期间还将使用工具箱 TIP。它们将不会部署到生产服务器。

对于工具箱 TIP 部署，版本上下文是下列各项的组合:

- 工具箱名称首字母缩写词
- 工具箱跟踪首字母缩写词（如果使用除 **Main** 外的跟踪）
- “Tip”

示例

应该使用版本上下文唯一地命名资源并在外部标识这些资源。

- 下表列示唯一标识的名称的示例。在此示例中，流程应用程序 TIP 使用了缺省跟踪名 (**Main**):

表 4. 具有缺省跟踪名的流程应用程序 TIP

流程应用程序名称	**Process Application 1**
流程应用程序名称首字母缩写词	**PA1**
流程应用程序跟踪	**main**
流程应用程序跟踪首字母缩写词	“”（当跟踪为 **Main** 时）
流程应用程序快照	
流程应用程序快照首字母缩写词	**Tip**

任何与此流程应用程序 TIP 相关联的 SCA 模块都包含版本上下文，如下表所示:

表 5. SCA 模块以及具有版本感知能力的 EAR 文件

SCA 模块名称	**具有版本感知能力的名称**	**具有版本感知能力的 EAR/应用程序名称**
M1	PA1-Tip-M1	PA1-Tip-M1.ear
M2	PA1-Tip-M2	PA1-Tip-M2.ear

- 下表列示使用了非缺省跟踪名称的流程应用程序 TIP 的示例:

表 6. 具有非缺省跟踪名称的流程应用程序 TIP

流程应用程序名称	**Process Application 1**
流程应用程序名称首字母缩写词	**PA1**
流程应用程序跟踪	**Track1**
流程应用程序跟踪首字母缩写词	**T1**
流程应用程序快照	
流程应用程序快照首字母缩写词	**Tip**

任何与此流程应用程序 TIP 相关联的 SCA 模块都包含版本上下文，如下表所示:

表 7. SCA 模块以及具有版本感知能力的 EAR 文件

SCA 模块名称	**具有版本感知能力的名称**	**具有版本感知能力的 EAR/应用程序名称**
M1	PA1-T1-Tip-M1	PA1-T1-Tip-M1.ear
M2	PA1-T1-Tip-M2	PA1-T1-Tip-M2.ear

Process Server 部署命名约定

在 Process Server 上，您可以部署流程应用程序的快照。流程应用程序快照首字母缩写词用于唯一地标识版本。

对于流程应用程序快照部署，版本上下文是下列各项的组合:

- 流程应用程序名称首字母缩写词
- 流程应用程序快照首字母缩写词

应该使用版本上下文唯一地命名资源并在外部标识这些资源。下表列示唯一标识的名称的示例:

表 8. 名称和首字母缩写词的示例

流程应用程序名称	**Process Application 1**
流程应用程序名称首字母缩写词	**PA1**
流程应用程序快照	**1.0.0**
流程应用程序快照首字母缩写词	**1.0.0**

资源（例如模块或库）将版本上下文作为它的身份的组成部分。

下表列示了两个模块的示例以及相关联的 EAR 文件如何包含版本上下文:

表 9. SCA 模块以及具有版本感知能力的 EAR 文件

SCA 模块名称	**具有版本感知能力的名称**	**具有版本感知能力的 EAR/应用程序名称**
M1	PA1-1.0.0-M1	PA1-1.0.0-M1.ear
M2	PA1-1.0.0-M2	PA1-1.0.0-M2.ear

下表列示了两个具有流程应用程序作用域的库的示例以及相关联的 JAR 文件如何包含版本上下文:

表 10. 具有流程应用程序作用域的库以及具有版本感知能力的 JAR 文件

具有 SCA 流程应用程序作用域的库名	具有版本感知能力的名称	具有版本感知能力的 JAR 名称
Lib1	PA1-1.0.0-Lib1	PA1-1.0.0-Lib1.jar
Lib2	PA1-1.0.0-Lib2	PA1-1.0.0-Lib2.jar

具有版本感知能力的绑定

流程应用程序可以包含包括导入和导出绑定的 SCA 模块。联合部署应用程序时，每个应用程序版本的绑定必须唯一。某些绑定在部署期间自动更新以确保其在各个版本之间的唯一性。在其他情况下，您必须在部署后更新绑定以确保其唯一性。

*具有版本感知能力*的绑定的作用域限定为特定的流程应用程序版本，这将保证该绑定在各个流程应用程序之间的唯一性。下列各节描述自动更新为具有版本感知能力的绑定，并描述了在运行时当绑定不具有版本感知能力时您需要执行的任何操作。有关创建模块时需要考虑的事项的信息，请参阅『使用绑定时的注意事项』。

SCA

如果模块的导入和导出绑定是在同一流程应用程序作用域内定义的，那么部署期间 SCA 绑定的目标将自动重新命名以便具有版本感知能力。

如果这些绑定不是在同一流程应用程序作用域内定义的，那么系统将会发出一条参考消息。完成部署后，必须修改导入绑定以更改端点目标地址。您可以使用管理控制台来更改端点目标地址。

Web Service（JAX-WS 或 JAX-RPC）

如果满足下列所有条件，那么部署期间 Web Service 绑定的端点目标地址将自动重命名以便具有版本感知能力:

- 您已遵循该地址的缺省命名约定:

  http://*ip*:*port*/*ModuleName*Web/sca/*ExportName*
- 端点地址为 SOAP/HTTP。
- 模块的导入和导出绑定是在同一流程应用程序作用域内定义的。

如果未满足这些条件，那么系统将会发出一条参考消息。您接下来将执行的操作取决于部署流程应用程序的方式:

- 如果您联合部署流程应用程序，那么必须手动重命名 SOAP/HTTP 端点 URL 或 SOAP/JMS 目标队列以使其在该流程应用程序的各个版本之间唯一。可以在部署后使用管理控制台来更改端点目标地址。
- 如果您只部署该流程应用程序的单一版本，那么可以忽略此消息。

对于 SOAP/ JMS Web Service 绑定快照联合部署而言，您执行的操作取决于部署流程应用程序的方式:

- 如果导入与目标导出位于同一个流程应用程序中，请先执行下列步骤，然后再将该流程应用程序发布到 Process Center 并创建快照:
  1. 更改导出的端点 URL。请确保目标和连接工厂唯一。
  2. 更改导入的端点 URL，以使其与您在上一步中对导出指定的端点 URL 相同。
- 如果导入与目标导出位于不同的流程应用程序中，请执行下列步骤:
  1. 更改导出的端点 URL。请确保目标和连接工厂唯一。

2. 将流程应用程序发布到 Process Center。
3. 创建快照。
4. 将流程应用程序部署到 Process Server。
5. 使用 WebSphere 管理控制台来更改相应导入的端点 URL，以使其与您对导出指定的端点 URL 相同。

HTTP

如果满足下列所有条件，那么部署期间 HTTP 绑定的端点 URL 地址将自动重命名以便具有版本感知能力:

- 您已遵循该地址的缺省命名约定:

  http(s)://*ip*:*port*/*ModuleName*Web/*contextPathinExport*

- 模块的导入和导出绑定是在同一流程应用程序作用域内定义的。

如果未满足这些条件，那么系统将会发出一条参考消息。您接下来将执行的操作取决于部署流程应用程序的方式:

- 如果您联合部署流程应用程序，那么必须手动重命名端点 URL 以使其在该流程应用程序的各个版本之间唯一。可以在部署后使用管理控制台来更改端点目标地址。
- 如果您只部署该流程应用程序的单一版本，那么可以忽略此消息。

JMS 和通用 JMS

系统生成的 JMS 和通用 JMS 绑定自动具有版本感知能力。

注: 对于用户定义的 JMS 和通用 JMS 绑定，部署期间不会自动进行重命名以使这些绑定具有版本感知能力。如果绑定是用户定义的，那么您必须重命名下列属性以使它们在流程应用程序的各个版本之间唯一:

- 端点配置
- 接收目标队列
- 侦听器端口名称（如果已定义）

如果您更改目标模块端点，请设置匹配的发送目标。

MQ/JMS 和 MQ

部署期间不会自动进行重命名以使类型为 MQ/JMS 或 MQ 的绑定具有版本感知能力。

您必须重命名下列属性以使它们在流程应用程序的各个版本之间唯一:

- 端点配置
- 接收目标队列

如果您更改目标模块端点，请设置匹配的发送目标。

EJB

部署期间不会自动进行重命名以使类型为 EJB 的绑定具有版本感知能力。

您必须重命名 JNDI 名称属性以使其在流程应用程序的各个版本之间唯一。

请注意，客户机应用程序也需要更新才能使用新的 JNDI 名称。

EIS

只要您未修改缺省资源名称 (***ModuleName*App:Adapter Description**)，那么在部署期间，系统会自动地将资源适配器重命名为具有版本感知能力的名称。

如果已对缺省资源名称进行修改，那么资源适配器名称必须在各个流程应用程序版本之间唯一。

如果资源适配器名称不唯一，那么部署期间将记录一条参考消息以提醒您存在此情况。完成部署后，您可以使用管理控制台以手动方式将资源适配器重命名。

具有版本感知能力的动态调用

您可以配置调解流组件，以便将消息路由到在运行时动态确定的端点。在创建调解模块时，可以将端点查找功能配置为使用具有版本感知能力的路由。

如果对快照使用 IBM_VRM 样式 (<*version*>.<*release*>.<*modification*)，那么可以将流程应用程序 EAR 文件导出到 WebSphere Service Registry and Repository (WSRR)。然后，在创建调解模块时，将端点查找功能配置为使用具有版本感知能力的路由。例如，从**匹配策略**字段中选择**返回与基于 SCA 模块的服务的最新兼容版本匹配的端点**，并对**绑定类型**选择 **SCA**。

此流程应用程序的将来版本将部署到服务器并发布到 WSRR，并且，调解模块端点查找功能将动态地调用服务端点的最新兼容版本。

注意，作为替代方法，可以在 SMOHeader 中设置目标，并且值可以包含在请求消息中。

部署包含 Java 模块和项目的流程应用程序

流程应用程序可以包含定制 Java EE 模块和 Java 项目。联合部署应用程序时，每个应用程序版本的定制 Java EE 模块必须唯一。

注意，如果定制 Java EE 模块和 Java 项目随已声明依赖于它们的 SCA 模块一起部署，那么它们将部署到服务器。如果声明依赖关系时未选中**随模块一起部署**（缺省选项），那么必须以手动方式部署该模块或项目。

部署包含业务规则和选择器的流程应用程序

如果您要部署包括业务规则或选择器组件的流程应用程序的多个版本，请注意这些版本使用相关联的元数据的方式。

业务规则或选择器组件的动态元数据是由组件名称、组件目标名称空间和组件类型在运行时定义的。如果将包含业务规则或选择器的流程应用程序的两个或两个以上版本部署到同一运行时环境，那么这些版本将共享相同规则逻辑（业务规则）或路由（选择器）元数据。

为了使流程应用程序的业务规则或选择器组件的每个版本都可以使用其自己的动态元数据（规则逻辑或路由），请重构目标名称空间以使其对流程应用程序的每个版本唯一。

绑定

面向服务的体系结构的核心是*服务*这一概念，此概念是指由计算设备之间的交互完成的功能单元。*导出*用于定义模块的外部接口（即访问点），以使此模块内的服务组件体系结构 (SCA) 组件能够向外部客户机提供服务。导入用于定义与模块外部的服务的接口，以便能够从模块中调用这些服务。您将特定于协议的*绑定*与导入和导出配合使用，以指定从外部向模块内或者从模块内向外部传输数据的方法。

导出

外部客户机可以使用各种格式（例如 XML、CSV、COBOL 和 JavaBean）的数据通过各种协议（例如 HTTP、JMS、MQ 和 RMI/IIOP）调用集成模块中的 SCA 组件。导出是一些组件，它们接收这些来自外部源的请求，然后使用 SCA 编程模型来调用 IBM Business Process Manager 组件。

例如，在下图中，导出通过 HTTP 协议接收来自客户机应用程序的请求。数据将变换为业务对象，即 SCA 组件所使用的格式。然后，使用该数据对象来调用此组件。

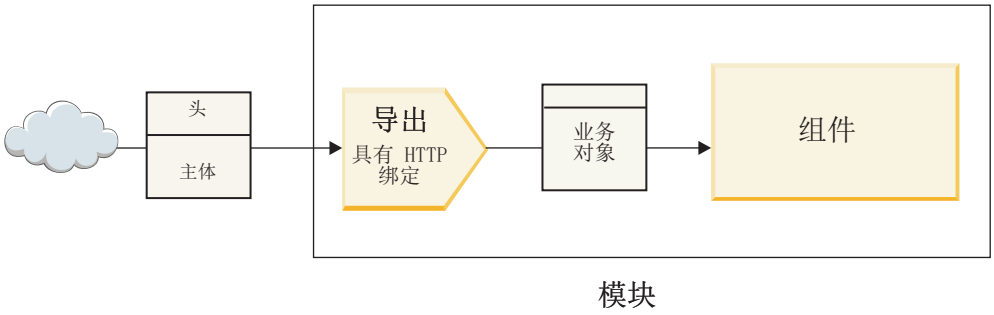


图 1. 具有 HTTP 绑定的导出

导入

SCA 组件可能想调用一个期望数据采用另一格式的非 SCA 外部服务。SCA 组件使用导入来通过 SCA 编程模型调用外部服务。然后，此导入以外部服务所期望的方式调用该服务。

例如，在下图中，导入将来自某个 SCA 组件的请求发送到外部服务。将业务对象（SCA 组件所使用的格式）变换为该服务所期望的格式，然后调用该服务。

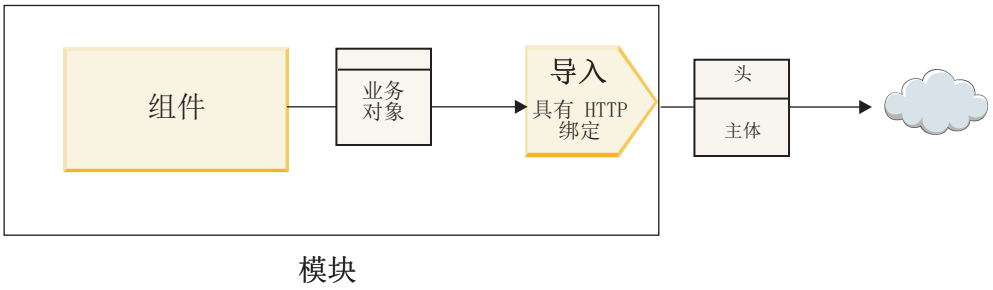


图 2. 具有 HTTP 绑定的导入

绑定列表

您可以使用 WebSphere Integration Developer 为导入或导出生成绑定以及配置绑定。以下列表对可用的绑定类型作了描述。

- SCA

  SCA 绑定（这是缺省绑定）允许服务与其他 SCA 模块中的服务进行通信。您可以使用具有 SCA 绑定的导入来访问另一 SCA 模块中的服务。并且，可以使用具有 SCA 绑定的导出向其他 SCA 模块提供服务。
- Web Service

Web Service 绑定允许您使用可互操作 SOAP 消息和服务质量来访问外部服务。另外，还可以使用 Web Service 绑定来包括附件作为 SOAP 消息的组成部分。

Web Service 绑定可以使用传输协议 SOAP/HTTP (SOAP over HTTP) 或 SOAP/JMS (SOAP over JMS)。无论使用哪种传输（HTTP 或 JMS）来传递 SOAP 消息，Web Service 绑定都始终以同步方式处理请求-响应交互。

- HTTP

  HTTP 绑定允许您在使用非 SOAP 消息或需要进行直接 HTTP 访问时使用 HTTP 协议来访问外部服务。如果您使用基于 HTTP 模型的 Web Service（即，使用 GET、PUT 和 DELETE 之类的熟知 HTTP 接口操作的服务），请使用此绑定。

- Enterprise JavaBeans (EJB)

  EJB 绑定允许 SCA 组件与 Java EE 服务器上运行的 Java EE 业务逻辑所提供的服务进行交互。

- EIS

  EIS（企业信息系统）绑定在与 JCA 资源适配器配合使用时，允许您访问企业信息系统中的服务或者使您的服务可供 EIS 使用。

- JMS 绑定

  Java 消息服务 (JMS)、通用 JMS 和 WebSphere MQ JMS (MQ JMS) 绑定用于与消息传递系统进行交互，在这方面，通过消息队列进行的异步通信对于可靠性而言至关重要。

  具有其中一个 JMS 绑定的导出将监测队列中的消息到达情况，并以异步方式将响应（如果有）发送到应答队列。具有其中一个 JMS 绑定的导入将构建消息并将其发送到 JMS 队列，而且监测队列中的响应（如果有）到达情况。

  – JMS

    JMS 绑定允许您访问 WebSphere 嵌入式 JMS 提供程序。

  – 通用 JMS

    通用 JMS 绑定允许您访问非 IBM 供应商消息传递系统。

  – MQ JMS

    MQ JMS 绑定允许您访问 WebSphere MQ 消息传递系统的 JMS 子集。在 JMS 功能子集足以满足应用程序的需要时，请使用此绑定。

- MQ

  WebSphere MQ 绑定允许您与 MQ 本机应用程序进行通信，从而将其引入面向服务的体系结构框架以及提供对特定于 MQ 的头信息的访问。当您需要使用 MQ 本机功能时，请使用此绑定。

导出和导入绑定概述

导出允许您使集成模块中的服务可供外部客户机使用，导入使集成模块中的 SCA 组件能够调用外部服务。与导出或导入相关联的绑定指定了协议消息与业务对象之间的关系。另外，它还指定选择操作和故障的方式。

通过导出的信息流

导出接收请求，该请求通过相关联绑定所确定的特定传输（例如 HTTP）针对该导出所连接的组件发出。

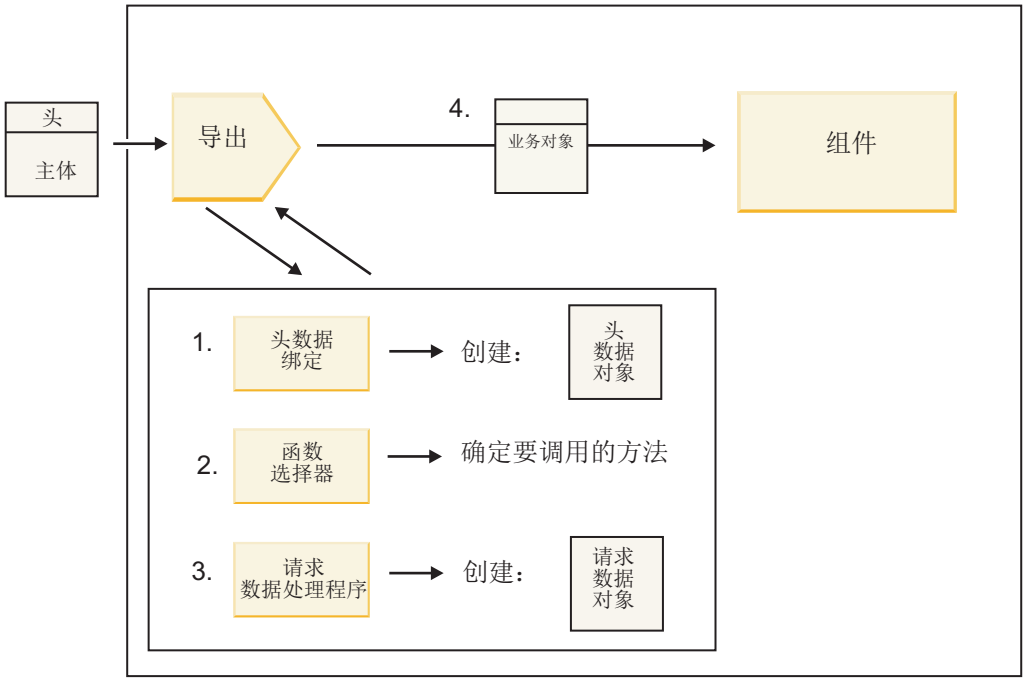


图 3. 通过导出的组件请求流

此导出接收到请求时，将发生以下事件顺序:

1. （仅限于 WebSphere MQ 绑定）头数据绑定将协议头变换为头数据对象。
2. 函数选择器根据协议消息确定本机方法名称。导出配置将本机方法名称映射到此导出的接口上的操作名称。
3. 方法的请求数据处理程序或数据绑定将此请求变换为请求业务对象。
4. 此导出使用该请求业务对象来调用组件方法。
   - HTTP 导出绑定和 Web Service 导出绑定以及 EJB 导出绑定以同步方式调用 SCA 组件。
   - JMS、通用 JMS、MQ JMS 和 WebSphere MQ 导出绑定以异步方式调用 SCA 组件。

注意，如果启用了上下文传播功能，那么导出可以传播它通过协议接收到的头和用户属性。于是，连接到此导出的组件就可以访问这些头和用户属性。有关更多信息，请参阅 WebSphere Integration Developer 信息中心中的“传播”主题。

如果这是双向操作，那么该组件将返回响应。

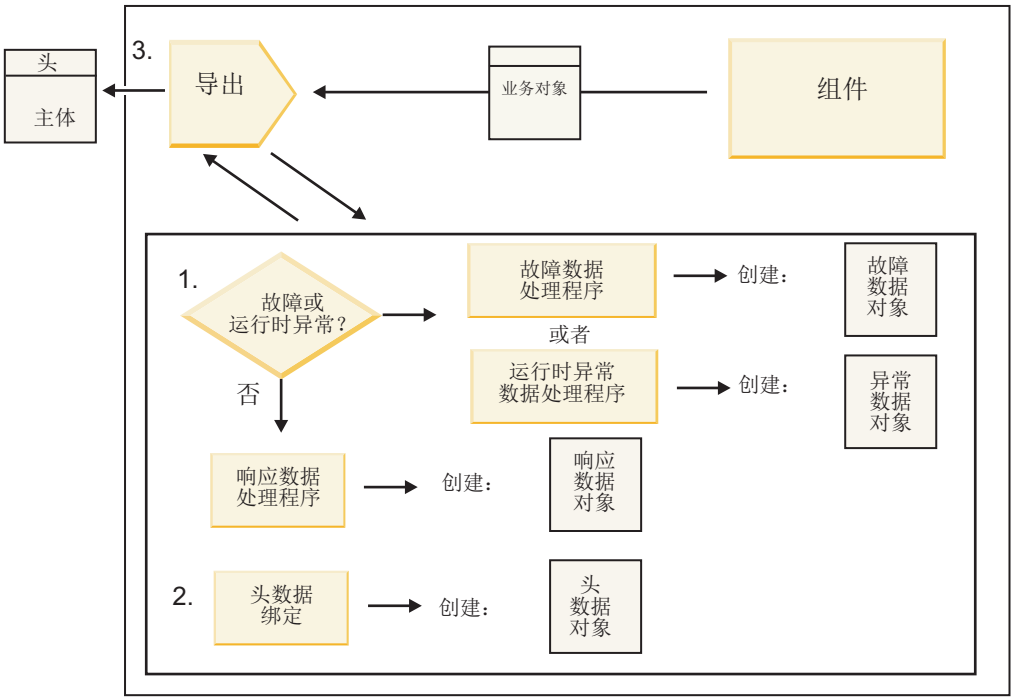


图 4. 通过导出返回的响应流

这将发生以下步骤序列:

1. 如果导出绑定接收到正常的响应消息，那么方法的响应数据处理程序或数据绑定将业务对象变换为响应。

   如果此响应是故障，那么方法的故障数据处理程序或数据绑定将该故障变换为故障响应。

   （仅限于 HTTP 导出绑定）如果此响应是运行时异常，那么将调用运行时异常数据处理程序（如果已配置此处理程序）。
2. （仅限于 WebSphere MQ 绑定）头数据绑定将头数据对象变换为协议头。
3. 此导出通过传输发送服务响应。

通过导入的信息流

组件使用导入向模块外部的服务发送请求。这将通过相关联绑定所确定的特定传输来发送此请求。

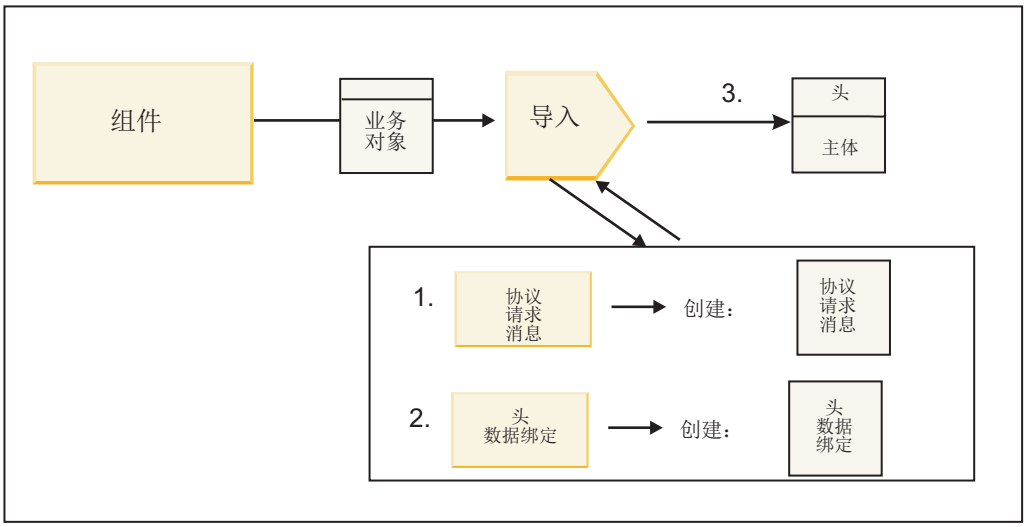


图 5. 从组件通过导入到服务的流

此组件使用请求业务对象来调用导入。

注:

- 执行调用的组件应该以同步方式调用 HTTP 导入绑定、Web Service 导入绑定和 EJB 导入绑定。
- 应该以异步方式调用 JMS、通用 JMS、MQ JMS 和 WebSphere MQ 导入绑定。

在组件调用此导入之后，将发生以下事件序列:

1. 方法的请求数据处理程序或数据绑定将请求业务对象变换为协议请求消息。
2. （仅限于 WebSphere MQ 绑定）方法的头数据绑定在协议头中设置头业务对象。
3. 此导入使用服务请求通过传输来调用该服务。

如果这是双向操作，那么该服务将返回响应，并且将发生以下步骤序列:

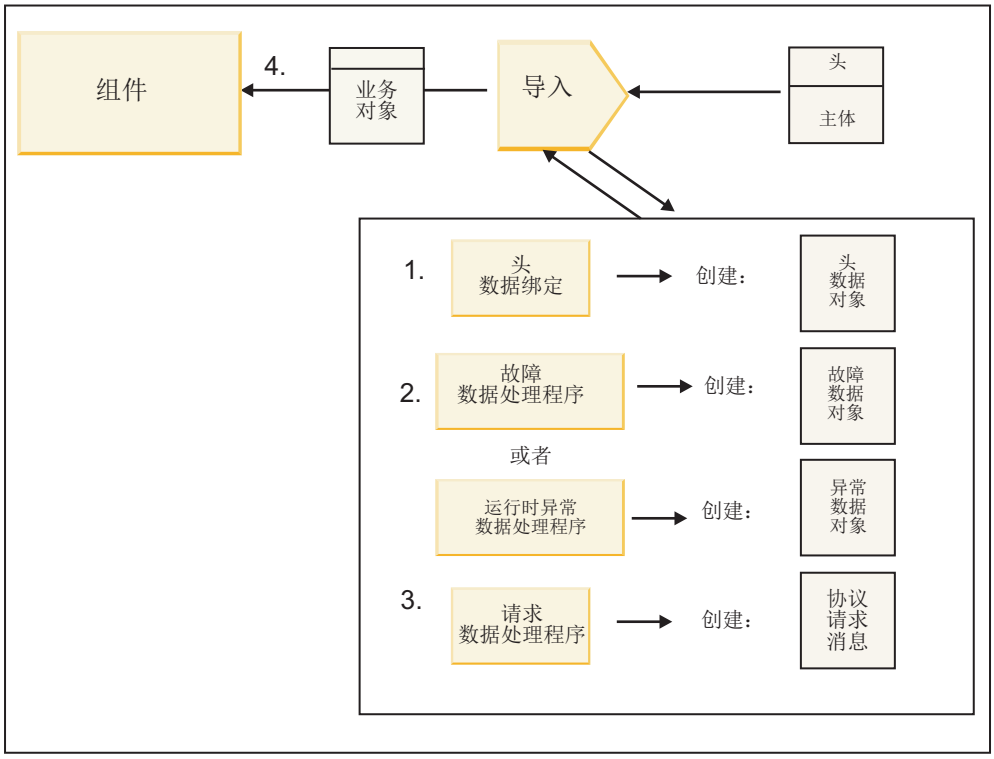


图 6. 通过导入返回的响应流

1. （仅限于 WebSphere MQ 绑定）头数据绑定将协议头变换为头数据对象。
2. 确定此响应是否为故障。
   - 如果此响应是故障，那么故障选择器将检查此故障以确定它所映射到的 WSDL 故障。然后，方法的故障数据处理程序将此故障变换为故障响应。
   - 如果此响应是运行时异常，那么将调用运行时异常数据处理程序（如果已配置此处理程序）。
3. 方法的响应数据处理程序或绑定将此响应变换为响应业务对象。
4. 此导入将该响应业务对象返回给组件。

导出和导入绑定配置

导出和导入绑定的其中一个重要方面是数据格式变换，它指示数据如何从本机有线格式映射（反序列化）为业务对象或者如何从业务对象映射（序列化）为本机有线格式。对于与导出相关联的绑定，您还可以指定函数选择器来指示应该对数据执行的操作。对于与导出或导入相关联的绑定，您可以指示应该如何处理在处理期间发生的故障。

另外，您还可以指定有关绑定的特定于输出的信息。例如，对于 HTTP 绑定，指定端点 URL。对于 HTTP 绑定，有关特定于输出的信息在“生成 HTTP 导入绑定”和“生成 HTTP 导出绑定”主题中作了阐述。您也可以在信息中心中找到有关其他绑定的信息。

导入和导出中的数据格式变换

在 WebSphere Integration Developer 中配置导出或导入绑定时，您指定的其中一个配置属性是绑定使用的数据格式。

- 对于导出绑定（在这种绑定中，客户机应用程序向 SCA 组件发送请求并接收来自该 SCA 组件的响应），您指示本机数据的格式。系统将根据该格式选择合适的数据处理程序或数据绑定以将本机数据变换成业务对象（此业务对象由 SCA 组件使用），并且相反地将业务对象变换成本机数据（此本机数据是对客户机应用程序的响应）。
- 对于导入绑定（在这种绑定中，SCA 组件向模块外的服务发送请求并接收来自该服务的响应），您指示本机数据的数据格式。系统将根据该格式选择适合的数据处理程序或数据绑定以将业务对象变换成本机数据以及将本机数据变换成业务对象。

IBM Business Process Manager 提供了一组预定义的数据格式以及支持这些格式的相应数据处理程序或数据绑定。您也可以创建自己的定制数据处理程序并为这些数据处理程序注册数据格式。有关更多信息，请参阅 WebSphere Integration Developer 信息中心中的“开发数据处理程序”主题。

- *数据处理程序*与协议无关，用于将数据从一种格式变换成另一种格式。在 IBM Business Process Manager 中，数据处理程序通常将本机数据（例如 XML、CSV 和 COBOL）变换成业务对象以及将业务对象变换成本机数据。由于数据处理程序与协议无关，因此，您可以将同一数据处理程序与各种导出和导入绑定重复配合使用。例如，可以将同一 XML 数据处理程序与 HTTP 导出或导入绑定或者与 JMS 导出或导入绑定配合使用。
- *数据绑定*也将本机数据变换成业务对象（反之亦然），但它们是特定于协议的。例如，HTTP 数据绑定只能与 HTTP 导出或导入绑定配合使用。与数据处理程序不同，HTTP 数据绑定不能与 MQ 导出或导入绑定重复配合使用。

  注: 从 IBM Business Process Manager V7.0 开始，不推荐使用以下三个 HTTP 数据绑定: HTTPStreamDataBindingSOAP、HTTPStreamDataBindingXML 和 HTTPServiceGatewayDataBinding。请尽可能使用数据处理程序。

如前面所述，您可以根据需要创建定制数据处理程序。另外，您也可以创建定制数据绑定；但是，建议您创建定制数据处理程序，这是因为数据处理程序可以跨多个绑定使用。

数据处理程序:

数据处理程序是针对导出和导入绑定配置的，用于以与协议无关的方式将数据从一种格式变换成另一种格式。随产品提供了一些数据处理程序，但您也可以根据需要创建自己的数据处理程序。可以在两个级别中的任一级别使数据处理程序与导出或导入绑定相关联: 可以使数据处理程序与导出或导入的接口中的所有操作相关联，也可以使数据处理程序与用于请求或响应的特定操作相关联。

预定义的数据处理程序

您可以使用 IBM Integration Designer 来指定要使用的数据处理程序。

下表列示已预定义以供您使用的数据处理程序，并且还描述每个数据处理程序变换入站和出站数据的方式。

注: 除非特别说明，否则这些数据处理程序可以与 JMS、通用 JMS、MQ JMS、WebSphere MQ 和 HTTP 绑定配合使用。
有关更详细的信息，请参阅 Integration Designer 信息中心中的“数据处理程序”主题。

表 11. 预定义的数据处理程序

数据处理程序	本机数据到业务对象	业务对象到本机数据
ATOM	将 ATOM 订阅源解析成 ATOM 订阅源业务对象。	将 ATOM 订阅源业务对象序列化为 ATOM 订阅源。
定界	将定界数据解析成业务对象。	将业务对象序列化为定界数据（包括 CSV）。

表 11. 预定义的数据处理程序 (续)

数据处理程序	本机数据到业务对象	业务对象到本机数据
定宽	将定宽数据解析成业务对象。	将业务对象序列化为定宽数据。
由 WTX 处理	将数据格式变换委派给 WebSphere Transformation Extender (WTX)。WTX 映射名由数据处理程序派生。	将数据格式变换委派给 WebSphere Transformation Extender (WTX)。WTX 映射名由数据处理程序派生。
由 WTX 调用者处理	将数据格式变换委派给 WebSphere Transformation Extender (WTX)。WTX 映射名由用户提供。	将数据格式变换委派给 WebSphere Transformation Extender (WTX)。WTX 映射名由用户提供。
JAXB	使用 Java XML 绑定体系结构 (JAXB) 规范所定义的映射规则将 Java Bean 序列化为业务对象。	使用 JAXB 规范所定义的映射规则将业务对象反序列化为 Java Bean。
JAXWS **注**: JAXWS 数据处理程序只能与 EJB 绑定配合使用。	此数据处理程序由 EJB 绑定通过使用“针对 XML Web Service 的 Java API” (JAX-WS) 规范所定义的映射规则将响应 Java 对象或异常 Java 对象变换成响应业务对象。	此数据处理程序由 EJB 绑定通过使用 JAX-WS 规范所定义的映射规则将业务对象变换成出局 Java 方法参数。
JSON	将 JSON 数据解析成业务对象。	将业务对象序列化为 JSON 数据。
本机主体	将本机字节、文本、映射、流或对象解析成五种基本业务对象（文本、字节、映射、流或对象）中的一种。	将五种基本业务对象变换成字节、文本、映射、流或对象。
SOAP	将 SOAP 消息（和头）解析成业务对象。	将业务对象序列化为 SOAP 消息。
XML	将 XML 数据解析成业务对象。	将业务对象序列化为 XML 数据。
UTF8XMLDataHandler	将 UTF-8 编码的 XML 数据解析成业务对象。	发送消息时将业务对象序列化为 UTF-8 编码的 XML 数据。

创建数据处理程序

您可以在 Integration Designer 信息中心的“开发数据处理程序”主题中找到有关创建数据处理程序的详细信息。

数据绑定:

数据绑定是针对导出和导入绑定配置的，用于将数据从一种格式变换成另一种格式。数据绑定特定于协议。随产品提供了一些数据绑定，但您也可以根据需要创建自己的数据绑定。可以在两个级别中的任一级别使数据绑定与导出或导入绑定相关联: 可以使数据绑定与导出或导入的接口中的所有操作相关联，也可以使数据绑定与用于请求或响应的特定操作相关联。

您可以使用 WebSphere Integration Developer 来指定要使用的数据绑定或创建自己的数据绑定。您可以在 WebSphere Integration Developer 信息中心中的“JMS、MQ JMS 和通用 JMS 绑定概述”部分中找到关于创建数据绑定的讨论。

JMS 绑定

下表列示可以与下列绑定配合使用的数据绑定:

- JMS 绑定
- 通用 JMS 绑定
- WebSphere MQ JMS 绑定

此表还包括这些数据绑定可以执行的任务的描述。

表 12. 适用于 JMS 绑定的预定义的数据绑定

数据绑定	本机数据到业务对象	业务对象到本机数据
序列化 Java 对象	将 Java 序列化对象变换成业务对象（此业务对象在 WSDL 中将映射为输入或输出类型）。	将业务对象序列化为 JMS 对象消息中的 Java 序列化对象。
包装字节	从入局的 JMS 字节消息中抽取字节，然后将这些字节包装成 JMSBytesBody 业务对象。	从 JMSBytesBody 业务对象中抽取字节，然后将这些字节包装成出局 JMS 字节消息。
包装映射条目	从入局的 JMS 映射消息中抽取每个条目的名称、值和类型信息并创建 MapEntry 业务对象的列表。然后，将此列表包装成 JMSMapBody 业务对象。	从 JMSMapBody 业务对象中的 MapEntry 列表中抽取名称、值和类型信息，然后在出局 JMS 映射消息中创建相应的条目。
包装对象	从入局的 JMS 对象消息中抽取对象，然后将此对象包装成 JMSObjectBody 业务对象。	从 JMSObjectBody 业务对象中抽取对象，然后将此对象包装成出局 JMS 业务对象。
包装文本	从入局的 JMS 文本消息中抽取文本，然后将此文本包装成 JMSTextBody 业务对象。	从 JMSTextBody 业务对象中抽取文本，然后将此文本包装成出局 JMS 文本对象。

WebSphere MQ 绑定

下表列示可以与 WebSphere MQ 绑定配合使用的数据绑定，并且描述这些数据绑定所执行的任务。

表 13. 适用于 WebSphere MQ 绑定的预定义的数据绑定

数据绑定	本机数据到业务对象	业务对象到本机数据
序列化 Java 对象	将入局消息中的 Java 序列化对象变换成业务对象（此业务对象在 WSDL 中将映射为输入或输出类型）。	将业务对象变换成出局消息中的 Java 序列化对象。
包装字节	从非结构化的 MQ 字节消息中抽取字节，然后将这些字节包装成 JMSBytesBody 业务对象。	从 JMSBytesBody 业务对象中抽取字节，然后将这些字节包装成出局非结构化 MQ 字节消息。
包装文本	从非结构化 MQ 文本消息中抽取文本，然后将此文本包装成 JMSTextBody 业务对象。	从 JMSTextBody 业务对象中抽取文本，然后将此文本包装成非结构化 MQ 文本消息。
包装流条目	从入局的 JMS 流消息中抽取每个条目的名称和类型信息并创建 StreamEntry 业务对象的列表。然后，将此列表包装成 JMSStreamBody 业务对象。	从 JMSStreamBody 业务对象中的 StreamEntry 列表中抽取名称和类型信息，然后在出局 JMSStreamMessage 中创建相应的条目。

除了 表 13 中列示的数据绑定外，WebSphere MQ 还使用头数据绑定。有关详细信息，请参阅 WebSphere Integration Developer 信息中心。

HTTP 绑定

下表列示可以与 HTTP 绑定配合使用的数据绑定，并且描述这些数据绑定所执行的任务。

表 14. 适用于 HTTP 绑定的预定义的数据绑定

数据绑定	本机数据到业务对象	业务对象到本机数据
包装字节	从入局的 HTTP 消息的主体中抽取字节，然后将这些字节包装成 HTTPBytes 业务对象。	从 HTTPBytes 业务对象中抽取字节，然后将这些字节添加至出局 HTTP 消息的主体。
包装文本	从入局的 HTTP 消息的主体中抽取文本，然后将此文本包装成 HTTPText 业务对象。	从 HTTPText 业务对象中抽取文本，然后将此文本添加至出局 HTTP 消息的主体。

导出绑定中的函数选择器

函数选择器用于指示应该对请求消息中的数据执行的操作。函数选择器配置为导出绑定的组成部分。

请考虑公开某个接口的 SCA 导出。该接口包含两项操作 - Create 和 Update。该导出具有从队列中读取内容的 JMS 绑定。

消息到达队列后，将向导出传递相关联的数据，但应该对所连接的组件调用该导出的接口中的哪项操作呢？该操作由函数选择器和导出绑定配置确定。

函数选择器将返回本机函数名（发送该消息的客户机系统中的函数名）。然后，该本机函数名映射至与该导出相关联的接口上的操作名或函数名。例如，在下图中，函数选择器返回入局消息中的本机函数名 (CRT)，该本机函数名将映射至 Create 操作，并且业务对象将发送至具有 Create 操作的 SCA 组件。

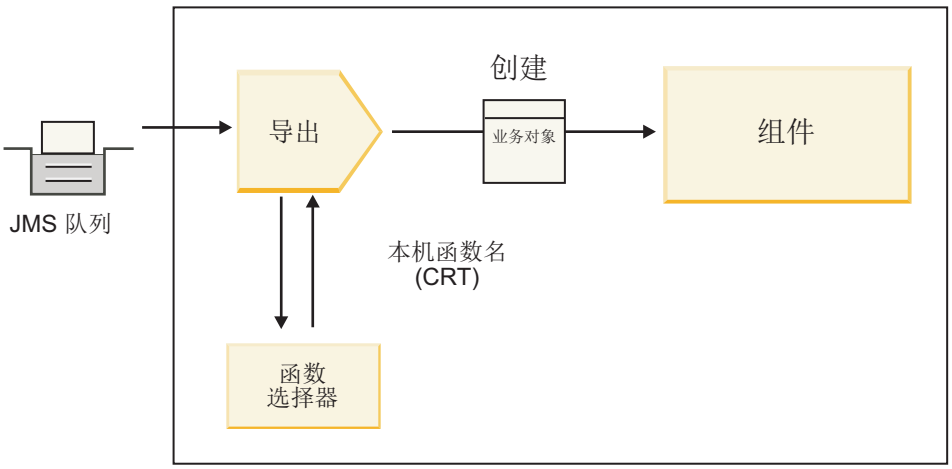


图 7. 函数选择器

如果该接口只有一项操作，那么无需指定函数选择器。

多个预先打包的函数选择器可用并且列示在下面各部分中。

JMS 绑定

下表列示可以与下列绑定配合使用的函数选择器:

- JMS 绑定

- 通用 JMS 绑定
- WebSphere MQ JMS 绑定

表 15. 适用于 JMS 绑定的预定义的函数选择器

函数选择器	描述
适用于简单 JMS 数据绑定的 JMS 函数选择器	使用消息的 JMSType 属性来选择操作名称。
JMS 头属性函数选择器	返回头中的 JMS 字符串属性 TargetFunctionName 的值。
JMS 服务网关函数选择器	通过检查客户机所设置的 JMSReplyTo 属性确定请求是单向还是双向操作。

WebSphere MQ 绑定

下表列示可以与 WebSphere MQ 绑定配合使用的函数选择器。

表 16. 适用于 WebSphere MQ 绑定的预定义的函数选择器

函数选择器	描述
MQ handleMessage 函数选择器	将 handleMessage 以值的形式返回，然后通过使用导出方法绑定将 handleMessage 映射至接口上的操作名称。
MQ 使用 JMS 缺省函数选择器	从 MQRFH2 头的文件夹的 TargetFunctionName 属性中读取本机操作。
MQ 使用消息主体格式作为本机函数	查找最后一个头的 Format 字段并将该字段以字符串形式返回。
MQ 类型函数选择器	通过检索包含在 MQRFH2 头中找到的 Msd、Set、Type 和 Format 属性的 URL 在导出绑定中创建方法。
MQ 服务网关函数选择器	使用 MQMD 头中的 MsgType 属性来确定操作名称。

HTTP 绑定

下表列示可以与 HTTP 绑定配合使用的函数选择器。

表 17. 适用于 HTTP 绑定的预定义的函数选择器

函数选择器	描述
基于 TargetFunctionName 头的 HTTP 函数选择器	使用来自客户机的 HTTP 头属性 TargetFunctionName 确定在运行时从导出调用哪项操作。
基于 URL 和 HTTP 方法的 HTTP 函数选择器	使用来自客户机且后面追加了 HTTP 方法的 URL 中的相对路径来确定对导出定义的本机操作。
基于带有操作名的 URL 的 HTTP 服务网关函数选择器	如果“operationMode = oneWay”已追加至请求 URL，那么根据该 URL 确定要调用的方法。

注: 您也可以使用 IBM Integration Designer 创建自己的函数选择器。可以在 IBM Integration Designer 信息中心中找到有关创建函数选择器的信息。例如，可以在“MQ 函数选择器概述”中找到关于为 WebSphere MQ 绑定创建函数选择器的描述。

故障处理

您可以通过指定故障数据处理程序来配置导入和导出绑定以处理在处理期间发生的故障（例如业务异常）。可以在三个级别设置故障数据处理程序: 可以使故障数据处理程序与某个故障相关联、与某项操作相关联或者与绑定相关联（针对所有操作）。

故障数据处理程序用于处理故障数据并将这些数据变换成要由导出或导入绑定发送的正确格式。

- 对于导出绑定，故障数据处理程序将组件所发送的异常业务对象变换成可由客户机应用程序使用的响应消息。
- 对于导入绑定，故障数据处理程序将服务所发送的故障数据或响应消息变换成可由 SCA 组件使用的异常业务对象。

导入绑定将调用用于确定响应消息为正常响应、业务故障还是运行时异常的故障选择器。

您可以对特定故障、某项操作或具有绑定的所有操作指定故障数据处理程序。

- 如果在三个级别都设置了故障数据处理程序，那么将调用与特定故障相关联的数据处理程序。
- 如果在操作和绑定级别设置了故障数据处理程序，那么将调用与该操作相关联的数据处理程序。

IBM Integration Designer 中使用了两个编辑器来指定故障处理。接口编辑器用于指示操作是否发生故障。使用此接口生成绑定后，“属性”视图中的编辑器允许您配置故障处理方式。有关更多信息，请参阅 IBM Integration Designer 信息中心中的“故障选择器”。

如何处理导出绑定中的故障:

如果在处理来自客户机应用程序的请求期间发生故障，那么导出绑定可以将故障信息返回给客户机。您可以配置导出绑定以指定处理故障以及将故障返回给客户机的方式。

可以使用 IBM Integration Designer 来配置导出绑定。

在请求处理期间，客户机通过发出请求来调用导出，并且该导出调用 SCA 组件。在处理该请求期间，SCA 组件可以返回业务响应，也可以抛出服务业务异常或服务运行时异常。抛出异常时，导出绑定将该异常变换成故障消息，然后将此消息发送给客户机，如下图所示。此过程在后面各部分中作了阐述。

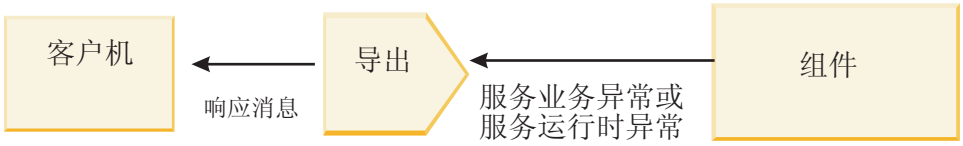


图 8. 故障信息如何从组件通过导出绑定发送到客户机

您可以创建定制数据处理程序或数据绑定来处理故障。

业务故障

业务故障是指处理期间发生的业务错误或异常。

请考虑以下接口，该接口上具有 createCustomer 操作。此操作定义了两个业务故障：CustomerAlreadyExists 和 MissingCustomerId。

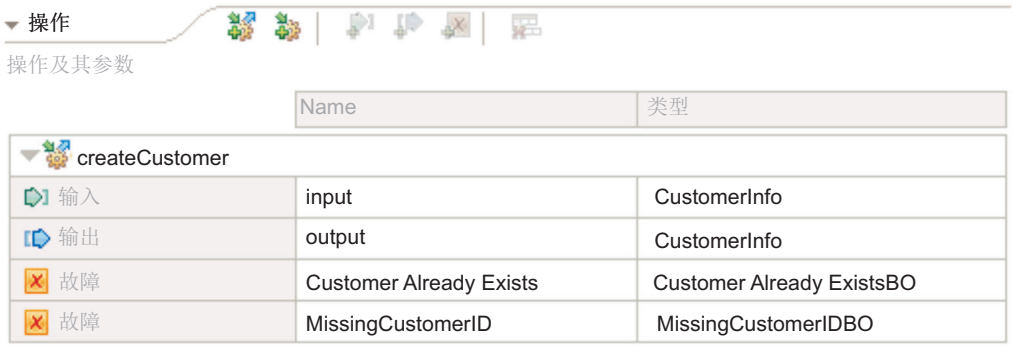


图 9. 具有两个故障的接口

在此示例中，如果客户机向此 SCA 组件发送一个创建客户的请求，但该客户已存在，那么该组件将向导出抛出 CustomerAlreadyExists 故障。导出需要将此业务故障传播回给调用客户机。为此，导出使用对导出绑定设置的故障数据处理程序。

当导出绑定接收到业务故障时，将执行下列处理:

1. 该绑定确定调用哪个故障数据处理程序来处理故障。如果服务业务异常包含故障名称，那么将调用对该故障设置的数据处理程序。如果服务业务异常未包含故障名称，那么将通过匹配的故障类型派生故障名称。
2. 该绑定使用服务业务异常中的数据对象调用故障数据处理程序。
3. 故障数据处理程序将故障数据对象变换成响应消息，然后将此消息返回给导出绑定。
4. 导出将响应消息返回给客户机。

如果服务业务异常包含故障名称，那么将调用对该故障设置的数据处理程序。如果服务业务异常未包含故障名称，那么将通过匹配的故障类型派生故障名称。

运行时异常

运行时异常是指在处理请求期间在 SCA 应用程序中发生的与业务故障不相对应的异常。与业务故障不同，运行时异常不是对接口定义的。

在某些情况下，您可能需要将这些运行时异常传播到客户机应用程序，以便客户机应用程序可以执行适当的操作。

例如，如果客户机向 SCA 组件发送一个创建客户的请求，并且在处理该请求期间发生授权错误，那么该组件将抛出运行时异常。此运行时异常必须传播回调用客户机，这样该客户机才能执行与授权相关的适当操作。这是由对导出绑定配置的运行时异常数据处理程序实现的。

注: 您只能对 HTTP 绑定配置运行时异常数据处理程序。

运行时异常的处理方式类似于业务故障的处理方式。如果设置了运行时异常数据处理程序，那么将执行下列处理:

1. 导出绑定使用服务运行时异常调用适当的数据处理程序。
2. 数据处理程序将故障数据对象变换成响应消息，然后将此消息返回给导出绑定。
3. 导出将响应消息返回给客户机。

故障处理和运行时异常处理是可选的。如果您不想将故障或运行时异常传播到调用客户机，请不要配置故障数据处理程序或运行时异常数据处理程序。

如何处理导入绑定中的故障:

组件使用导入向模块外的服务发送请求。如果在处理该请求期间发生故障，那么该服务将故障返回给导入绑定。您可以配置导入绑定以指定处理故障以及将故障返回给组件的方式。

可以使用 IBM Integration Designer 配置导入绑定。您可以指定故障数据处理程序（或数据绑定），并且还可以指定故障选择器。

故障数据处理程序

处理请求的服务将故障信息以异常的形式或者将包含故障数据的响应消息发送至导入绑定。

导入绑定将该服务异常或响应消息变换成服务业务异常或服务运行时异常，如下图所示。此过程在后面各部分中作了阐述。

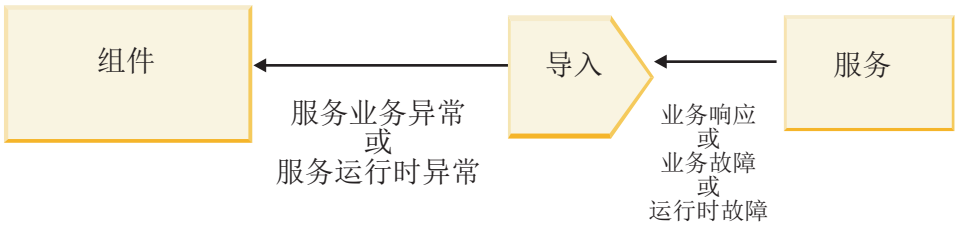


图 10. 故障信息如何从服务通过导入发送到组件

您可以创建定制数据处理程序或数据绑定来处理故障。

故障选择器

当您配置导入绑定时，可以指定故障选择器。故障选择器确定导入响应为实际响应、业务异常还是运行时故障。另外，它还根据响应主体或头确定本机故障名称，该故障名称将通过绑定配置映射至关联接口中的故障名称。

两种类型的预先打包的故障选择器可供与 JMS、MQ JMS、通用 JMS、WebSphere MQ 和 HTTP 导入配合使用:

表 18. 预先打包的故障选择器

故障选择器类型	描述
基于头	根据入局响应消息中的头确定响应消息是业务故障、运行时异常还是正常消息。
SOAP	确定响应 SOAP 消息是正常响应、业务故障还是运行时异常。

以下显示基于头的故障选择器和 SOAP 故障选择器的示例。

- 基于头的故障选择器

  如果应用程序想要指示入局消息是业务故障，那么入局消息中必须有两个头表示业务故障，如下所示:

  ```
  Header name = FaultType, Header value = Business
  Header name = FaultName, Header value = <user defined native fault name>
  ```

  如果应用程序想要指示入局响应消息是运行时异常，那么入局消息中必须有一个头，如下所示:

  ```
  Header name = FaultType, Header value = Runtime
  ```

- SOAP 故障选择器

  可以将业务故障作为包含定制 SOAP 头的 SOAP 消息的组成部分发送。在以下示例中，“CustomerAlreadyExists”是故障的名称。

```
<ibmSoap:BusinessFaultName
xmlns:ibmSoap="http://www.ibm.com/soap">CustomerAlreadyExists
<ibmSoap:BusinessFaultName>
```

故障选择器是可选的。如果您未指定故障选择器，那么导入绑定将无法确定响应的类型。因此，该绑定将响应视为业务响应，并且调用响应数据处理程序或数据绑定。

您可以创建定制故障选择器。在 IBM Integration Designer 信息中心的“开发定制故障选择器”主题中提供了有关创建定制故障选择器的步骤。

业务故障

如果处理请求期间出现错误，那么可能产生业务故障。例如，如果您发送一个创建客户的请求，但该客户已存在，那么服务将向导入绑定发送业务异常。

绑定接收到业务异常后，处理步骤将取决于是否为该绑定设置了故障选择器。

- 如果未设置故障选择器，那么该绑定将调用响应数据处理程序或数据绑定。
- 如果设置了故障选择器，那么将执行下列处理:
  1. 导入绑定调用故障选择器以确定响应是是业务故障、业务响应还是运行时故障。
  2. 如果响应是业务故障，那么导入绑定将调用故障选择器以提供本机故障名称。
  3. 导入绑定确定与故障选择器所返回的本机故障名称相对应的 WSDL 故障。
  4. 导入绑定确定为此 WSDL 故障配置的故障数据处理程序。
  5. 导入绑定使用故障数据调用此故障数据处理程序。
  6. 故障数据处理程序将故障数据变换成数据对象，然后将此数据对象返回给导入绑定。
  7. 导入绑定使用数据对象和故障名称构造一个服务业务异常对象。
  8. 导入将此服务业务异常对象返回给组件。

运行时异常

如果在与服务通信期间出现问题，那么可能产生运行时异常。运行时异常的处理方式类似于业务异常的处理方式。如果设置了故障选择器，那么将执行下列处理:

1. 导入绑定使用异常数据调用适当的运行时异常数据处理程序。
2. 运行时异常数据处理程序将异常数据变换成服务运行时异常对象，然后将此对象返回给导入绑定。
3. 导入将此服务运行时异常对象返回给组件。

SCA 模块与开放式 SCA 服务之间的互操作性

IBM WebSphere Application Server V7.0 Feature Pack for Service Component Architecture (SCA) 提供了简单但功能强大的编程模型来构造基于开放式 SCA 规范的应用程序。IBM Business Process Manager 的 SCA 模块使用导入和导出绑定与在 Rational® Application Developer 环境中开发并且由 WebSphere Application Server Feature Pack for Service Component Architecture 主管的开放式 SCA 服务互操作。

SCA 应用程序通过导入绑定调用开放式 SCA 应用程序。SCA 应用程序通过导出绑定接收来自开放式 SCA 应用程序的调用。第 47 页的『通过可互操作的绑定调用服务』中显示了受支持的绑定的列表。

从 SCA 模块调用开放式 SCA 服务

使用 IBM Integration Designer 开发的 SCA 应用程序可以调用在 Rational Application Developer 环境中开发的开放式 SCA 应用程序。本部分提供一个使用 SCA 导入绑定从 SCA 模块调用开放式 SCA 服务的示例。

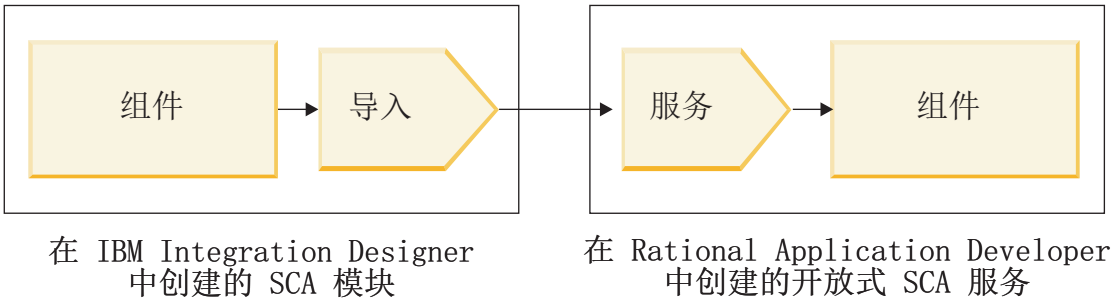


图 11. SCA 模块中调用开放式 SCA 服务的组件

调用开放式 SCA 服务无需特殊配置。

要通过 SCA 导入绑定连接到开放式 SCA 服务，请在导入绑定中提供开放式 SCA 服务的组件名称和服务名称。

1. 要从开放式 SCA 组合体中获取目标组件和服务的名称，请执行下列步骤:
   a. 通过单击**窗口** → **显示视图** → **属性**确保**属性**选项卡已打开。
   b. 通过双击包含该组件和服务的组合图来打开组合编辑器。例如，对于名为 **customer** 的组件，组合图为 **customer.composite_diagram**。
   c. 单击目标组件。
   d. 在**属性**选项卡的**名称**字段中，记录目标组件的名称。
   e. 单击与该组件相关联的服务图标。
   f. 在**属性**选项卡的**名称**字段中，记录服务的名称。
2. 要配置 IBM Business Process Manager 导入以使其与开放式 SCA 服务连接，请执行下列步骤:
   a. 在 IBM Integration Designer 中，浏览至要与开放式 SCA 服务连接的 SCA 导入的**属性**选项卡。
   b. 在**模块名称**字段中，输入步骤 1d 中的组件名称。
   c. 在**导出名称**字段中，输入步骤 1f 中的服务名称。
   d. 通过按 Ctrl+S 键保存您的工作。

从开放式 SCA 服务调用 SCA 模块

在 Rational Application Developer 环境中开发的开放式 SCA 应用程序可以调用使用 IBM Integration Designer 开发的 SCA 应用程序。本部分提供一个通过 SCA 导出绑定从开放式 SCA 服务调用 SCA 模块的示例。

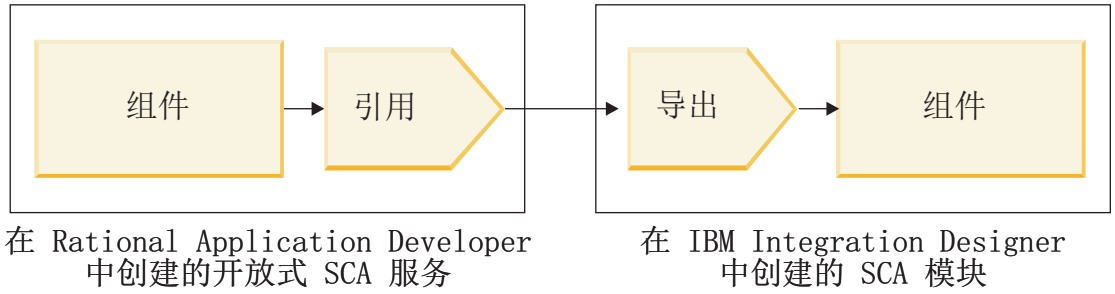


图 12. 调用 SCA 模块中的组件的开放式 SCA 服务

要通过开放式 SCA 引用绑定连接到 SCA 组件，请提供模块名称和导出名称。

1. 要获取目标模块和导出的名称，请执行下列步骤:
   a. 在 IBM Integration Designer 中，通过双击模块在组合件编辑器中打开该模块。
   b. 单击导出。
   c. 在**属性**选项卡的**名称**字段中，记录导出的名称。
2. 配置要与 IBM Business Process Manager 模块和导出连接的开放式 SCA 引用:
   a. 在 Rational Application Developer 中，通过双击包含该组件和服务的组合图来打开组合编辑器。
   b. 单击组件引用的引用图标以便在**属性**选项卡中显示引用属性。
   c. 单击此页面左侧的**绑定**选项卡。
   d. 单击**绑定**，然后单击**添加**。
   e. 选择 **SCA** 绑定。
   f. 在 **URI** 字段中，输入 IBM Business Process Manager 模块名称，后跟斜杠“/”，接着后跟导出名称（此名称是在步骤 1c 中确定的）。
   g. 单击**确定**。
   h. 通过按 Ctrl+S 键保存您的工作。

通过可互操作的绑定调用服务

支持下列绑定与开放式 SCA 服务互操作。

- SCA 绑定

  在 IBM Business Process Manager 中，当 SCA 模块通过 SCA 导入绑定调用开放式 SCA 服务时，支持下列调用样式:

  – 异步（单向）
  – 同步（请求-响应）

  SCA 导入接口和开放式 SCA 服务接口必须使用符合 Web Service 互操作性 (WS-I) 的 WSDL 接口。

  请注意，SCA 绑定支持传播事务和安全上下文。
- 使用 SOAP1.1/HTTP 或 SOAP1.2/HTTP 协议的 Web Service (JAX-WS) 绑定

  SCA 导入接口和开放式 SCA 服务接口必须使用符合 Web Service 互操作性 (WS-I) 的 WSDL 接口。

  另外，还支持下列服务质量:

  - Web Service 原子事务
  - Web Service 安全性
- EJB 绑定

  使用 EJB 绑定时，Java 接口用于定义 SCA 模块与开放式 SCA 服务之间的交互。

  请注意，EJB 绑定支持传播事务和安全上下文。
- JMS 绑定

  SCA 导入接口和开放式 SCA 服务接口必须使用符合 Web Service 互操作性 (WS-I) 的 WSDL 接口。

  支持下列 JMS 提供程序:
  - WebSphere Platform Messaging（JMS 绑定）
  - WebSphere MQ（MQ JMS 绑定）

注: 业务图通过任何 SCA 绑定都不可互操作，因此，在用于与 WebSphere Application Server Feature Pack for Service Component Architecture 交互的接口中不受支持。

绑定类型

您将特定于协议的*绑定*与导入和导出配合使用，以指定从外部向模块内或者从模块内向外部传输数据的方法。

选择适当的绑定

提供了各种绑定以满足应用程序的需要。

WebSphere Integration Developer 中提供的绑定具有一系列选项。此列表可帮助您确定某种类型的绑定何时更能满足应用程序的需要。

满足下列因素时，请考虑使用 *SCA* 绑定:
- 所有服务都包含在 WebSphere Integration Developer 模块中；即，没有外部服务。
- 您想要将功能分配到互相直接进行交互的不同 SCA 模块中。
- 各个模块紧密耦合。

满足下列因素时，请考虑使用 *Web Service* 绑定:
- 您需要通过因特网访问外部服务或者通过因特网提供服务。
- 服务之间松散耦合。
- 最好是进行同步通信；即，来自一项服务的请求可以等待另一项服务作出响应。
- 您正在访问的外部服务或者您希望提供的服务使用 SOAP/HTTP 或 SOAP/JMS 协议。

满足下列因素时，请考虑使用 *HTTP* 绑定:
- 您需要通过因特网访问外部服务或者通过因特网提供服务，并且您正在使用其他基于 HTTP 模型的 Web Service（即，使用熟知的 HTTP 接口操作，例如 GET、PUT 和 DELETE 等）。
- 服务之间松散耦合。
- 最好是进行同步通信；即，来自一项服务的请求可以等待另一项服务作出响应。

满足下列因素时，请考虑使用 *EJB* 绑定:
- 这种绑定适用于本身是 EJB 或者需要通过 EJB 客户机访问的已导入服务。
- 导入的服务松散耦合。

- 不需要进行有状态 EJB 交互。
- 最好是进行同步通信；即，来自一项服务的请求可以等待另一项服务作出响应。

满足下列因素时，请考虑使用 *EIS* 绑定:
- 您需要使用资源适配器来访问 EIS 系统上的服务。
- 优先采用同步数据传输而不是异步数据传输。

满足下列因素时，请考虑使用 *JMS* 绑定:

注: 有多种类型的 JMS 绑定。如果您希望使用 JMS 来交换 SOAP 消息，请考虑对 Web Service 绑定使用 SOAP/JMS 协议。请参阅『Web Service 绑定』。
- 您需要访问消息传递系统。
- 服务之间松散耦合。
- 优先采用异步数据传输而不是同步数据传输。

满足下列因素时，请考虑使用*通用 JMS* 绑定:
- 您需要访问非 IBM 供应商提供的消息传递系统。
- 服务之间松散耦合。
- 可靠性比性能更重要；即，优先采用异步数据传输而非同步数据传输。

满足下列因素时，请考虑使用 *MQ* 绑定:
- 您需要访问 WebSphere MQ 消息传递系统，并且需要使用 MQ 本机功能。
- 服务之间松散耦合。
- 可靠性比性能更重要；即，优先采用异步数据传输而非同步数据传输。

满足下列因素时，请考虑使用 *MQ JMS* 绑定:
- 您需要访问 WebSphere MQ 消息传递系统，但是可以在 JMS 上下文中执行此访问；即，JMS 功能子集就足以满足应用程序的需要。
- 服务之间松散耦合。
- 可靠性比性能更重要；即，优先采用异步数据传输而非同步数据传输。

SCA 绑定

服务组件体系结构 (SCA) 绑定允许服务与其他模块中的其他服务通信。具有 SCA 绑定的导入允许您访问另一个 SCA 模块中的服务。具有 SCA 绑定的导出允许您提供服务供其他模块使用。

您可以使用 WebSphere Integration Developer 为 SCA 模块中的导入和导出生成并配置 SCA 绑定。

如果模块在同一服务器上运行或者部署在同一集群中，那么 SCA 绑定是最容易使用且速度最快的绑定。

将包含 SCA 绑定的模块部署到服务器后，您可以使用管理控制台来查看关于该绑定的信息或者更改该绑定的所选属性（对于导入绑定）。

Web Service 绑定

Web Service 绑定是用于将消息从服务组件体系结构 (SCA) 组件传输到 Web Service 以及从 Web Service 传输到 SCA 组件的方法。

Web Service 绑定概述:

Web Service 导入绑定允许您从服务组件体系结构 (SCA) 组件调用外部 Web Service。Web Service 导出绑定允许您将 SCA 组件作为 Web Service 向客户机公开。

借助 Web Service 绑定，您可以使用可互操作 SOAP 消息和服务质量 (QoS) 来访问外部服务。

可以使用 Integration Designer 对 SCA 模块中的导入和导出生成并配置 Web Service 绑定。下列类型的 Web Service 绑定可用:

- SOAP1.2/HTTP 和 SOAP1.1/HTTP

  这些绑定基于“针对 XML Web Service 的 Java API”(JAX-WS)，JAX-WS 是一个用于创建 Web Service 的 Java 编程 API。

  – 如果 Web Service 符合 SOAP 1.2 规范，请使用 SOAP1.2/HTTP。
  – 如果 Web Service 符合 SOAP 1.1 规范，请使用 SOAP1.1/HTTP。

  要点: 部署具有 Web Service (JAX-WS) 绑定的应用程序时，目标服务器不能选中**根据需要启动组件**选项。有关详细信息，请参阅第 57 页的『检查服务器配置』。

  选择这些绑定中的一个绑定后，您可以随 SOAP 消息发送附件。

  Web Service 绑定处理标准 SOAP 消息。但是，通过使用其中一个 Web Service JAX-WS 绑定，您可以定制解析或编写 SOAP 消息的方式。例如，可以处理 SOAP 消息中的非标准元素或者对 SOAP 消息应用其他处理。配置绑定时，可以指定用于对 SOAP 消息执行这种处理的定制数据处理程序。

  可以将策略集与 Web Service (JAX-WS) 绑定配合使用。策略集是策略类型组成的集合，每种策略类型都提供一种服务质量 (QoS)。例如，WSAddressing 策略集提供了一种与传输无关的方法对 Web Service 和消息进行统一寻址。您可以使用 Integration Designer 为绑定选择策略集。

  注: 如果要使用安全性断言标记语言 (SAML) 策略集，那么必须按第 55 页的『导入 SAML 策略集』所述执行一些其他配置。

- SOAP1.1/HTTP

  如果要创建的 Web Service 使用基于 JAX-RPC（针对基于 XML 的 RPC 的 Java API）的 SOAP 编码消息，请使用此绑定。

- SOAP1.1/JMS

  使用此绑定来发送或接收使用 Java 消息服务 (JMS) 目标的 SOAP 消息。

无论使用哪种传输（HTTP 或 JMS）来传递 SOAP 消息，Web Service 绑定都始终以同步方式处理请求-响应交互。在接收到来自服务提供者的响应之前，调用该提供者的线程将被阻塞。有关这种调用样式的更多信息，请参阅“同步调用”。

要点: 不能对同一模块中的导出使用下列 Web Service 绑定的组合。如果需要公开使用这些导出绑定中的多个绑定的组件，那么需要另一个模块中具有每个绑定，然后将这些模块连接至使用 SCA 绑定的组件:

- 使用 JAX-RPC 的 SOAP 1.1/JMS 和 SOAP 1.1/HTTP
- 使用 JAX-RPC 的 SOAP 1.1/HTTP 和使用 JAX-WS 的 SOAP 1.1/HTTP
- 使用 JAX-RPC 的 SOAP 1.1/HTTP 和使用 JAX-WS 的 SOAP 1.2/HTTP

将包含 Web Service 绑定的 SCA 模块部署到服务器后，您可以使用管理控制台来查看关于该绑定的信息或者更改该绑定的所选属性。

注: Web Service 允许应用程序通过使用服务的标准描述以及它们所交换的消息的标准格式进行互操作。例如，Web Service 导入和导出绑定可以与使用 Web Services Enhancements (WSE) V3.5 和用于 Microsoft .NET 的 Windows Communication Foundation (WCF) V3.5 实现的 Web Service 互操作。与这种服务互操作时，您必须确保:

- 对于接口中的每项操作，用于访问 Web Service 导出的 Web 服务描述语言 (WSDL) 文件都包含一个非空的 SOAP 操作值。
- 向 Web Service 导出发送消息时，Web Service 客户机将设置 SOAPAction 头或 wsa:Action 头。

SOAP 头传播:

在处理 SOAP 消息时，可能需要访问所接收的消息中某些 SOAP 头中的信息，可能需要确保在 SOAP 头包含特定值的情况下发送消息，或者需要允许在模块中传递 SOAP 头。

在 Integration Designer 中配置 Web Service 绑定时，您可以指示要传播 SOAP 头。

- 在导出处接收到请求或者在导入处接收到响应时，可以访问 SOAP 头信息，从而允许模块中的逻辑基于头值以及允许对那些头进行修改。
- 从导出发送请求或者从导入发送响应时，可以在那些消息中包括 SOAP 头。

所传播的 SOAP 头的格式和存在性可能受您对导入或导出配置的策略集影响，如第 52 页的表 19所述。

要为导入或导出配置 SOAP 头的传播，请从 Integration Designer 的“属性”视图中选择**传播协议头**选项卡，然后选择所需的选项。

WS-Addressing 头

WS-Addressing 头可以由 Web Service (JAX-WS) 绑定传播。

在传播 WS-Addressing 头时，请注意下列信息:

- 如果对 WS-Addressing 头启用传播，那么这个头在下列情况下将传播到模块中:
  - 在导出处接收到请求时
  - 在导入处接收到响应时
- WS-Addressing 头不会从 IBM Business Process Manager 传播到出站消息（即，从导入发送请求或者从导出发送响应时，不会传播这个头）。

WS-Security 头

WS-Security 头既可以由 Web Service (JAX-WS) 绑定传播也可以由 Web Service (JAX-RPC) 绑定传播。

Web Service WS-Security 规范描述了对 SOAP 消息传递进行的增强，以便通过消息完整性、消息机密性和单一消息认证来提供保护质量。这些机制可用于适应各种安全性模型以及加密技术。

在传播 WS-Security 头时，请注意下列信息:

- 如果对 WS-Security 头启用传播，那么这个头在下列情况下将在模块中传播:
  - 在导出处接收到请求时
  - 从导入发送请求时
  - 在导入处接收到响应时
- 在缺省情况下，从导出发送响应时，不会传播这个头。但是，如果将 JVM 属性 **WSSECURITY.ECHO.ENABLED** 设置为 **true**，那么从导出发送响应时，将传播这个头。在这种情况下，如果未在请求路径中修改 WS-Security 头，那么可能会自动地将 WS-Security 头从请求回传到响应中。

- 对于从导入为请求发送的 SOAP 消息或者从导出为响应发送的 SOAP 消息而言，这些消息的准确格式可能与最初接收到的 SOAP 消息不完全匹配。因此，应该认为任何数字签名均变为无效。如果在发送的消息中需要数字签名，那么必须使用适当的安全策略集来确定此签名，并应该在模块中除去所接收的消息中与数字签名相关的 WS-Security 头。

要传播 WS-Security 头，必须将 WS-Security 模式与应用程序模块包括在一起。请参阅『将 WS-Security 模式包括在应用程序模块中』以了解包括此模式的过程。

头的传播方式

头的传播方式取决于导入或导出绑定的安全策略设置，如表 19 所示:

表 19. 安全性头的传递方式

	不具有安全策略的导出绑定	具有安全策略的导出绑定
不具有安全策略的导入绑定	安全性头在模块中按原样传递。系统不会对其进行解密。 这些头以其接收格式向站外发送。 数字签名可能会变为无效。	尽管模块具有签名验证和认证功能，但还是对安全性头进行解密并传递。 向站外发送已解密的头。 数字签名可能会变为无效。
具有安全策略的导入绑定	安全性头在模块中按原样传递。系统不会对其进行解密。 不应将这些头传播到导入。否则，将由于重复而发生错误。	尽管模块具有签名验证和认证功能，但还是对安全性头进行解密并传递。 不应将这些头传播到导入。否则，将由于重复而发生错误。

请对导出绑定和导入绑定配置适当的策略集，其原因在于，这样将使服务请求者不受服务提供者的配置更改或 QoS 需求更改影响。然后，可以使标准的 SOAP 头在模块中可视，从而影响模块中的处理（例如，进行日志记录和跟踪）。将 SOAP 头通过模块从接收到的消息传播到发送的消息并不表示模块的隔离益处受损。

如果导入或导出具有通常会导致生成标准头（例如 WS-Security 头）的相关联策略集，那么不应在发送到该导入的请求或者发送到该导出的响应中传播这些头。否则，将由于头重复而发生错误。而是，应该显式地除去这些头，或者应该将导入绑定或导出绑定配置为阻止传播协议头。

访问 SOAP 头

从 Web Service 导入或导出接收到包含 SOAP 头的消息时，这些头将被放到服务消息对象 (SMO) 的头部分中。您可以访问头信息，如“访问 SMO 中的 SOAP 头信息”所述。

将 WS-Security 模式包括在应用程序模块中

以下过程概述在应用程序模块中包括此模式的步骤:

- 如果运行 Integration Designer 的计算机有权访问因特网，请执行下列步骤:
  1. 在“业务集成”透视图中，对项目选择**依赖性**。
  2. 展开**预定义的资源**并选择 **WS-Security 1.0 模式文件**或 **WS-Security 1.1 模式文件**，以便将模式导入到模块中。
  3. 清理并重建项目。
- 如果运行 Integration Designer 的计算机无权访问因特网，那么您可以将模式下载到另一台有权访问因特网的计算机。然后，可以将模式复制到运行 Integration Designer 的计算机。
  1. 在有权访问因特网的计算机上，下载远程模式:

a. 单击**文件 → 导入 → 业务集成 → WSDL 和 XSD**。
b. 选择**远程 WSDL 或 XSD 文件**。
c. 导入下列模式:

http://www.w3.org/2003/05/soap-envelope/

http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd

http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd

2. 将这些模式复制到无权访问因特网的计算机。
3. 在无权访问因特网的计算机上，导入模式:
   a. 单击**文件 → 导入 → 业务集成 → WSDL 和 XSD**。
   b. 选择**本地 WSDL 或 XSD 文件**。
4. 更改 oasis-wss-wssecurity_secext-1.1.xsd 的模式位置:
   a. 打开 *workplace_location*/*module_name*/StandardImportFilesGen/oasis-wss-wssecurity-secext-1.1.xsd 处的模式。
   b. 将:

```
<xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
schemaLocation='http://www.w3.org/2003/05/soap-envelope/'/>
```

   更改为:

```
<xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
schemaLocation='../w3/_2003/_05/soap_envelope.xsd'/>
```

   c. 将:

```
<xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
schemaLocation='http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd'/>
```

   更改为:

```
<xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
schemaLocation='../w3/tr/_2002/rec_xmlenc_core_20021210/xenc-schema.xsd'/>
```

5. 更改 oasis-200401-wss-wssecurity-secext-1.0.xsd 的模式位置:
   a. 打开 *workplace_location*/*module_name*/StandardImportFilesGen/oasis-200401-wss-wssecurity-secext-1.0.xsd 处的模式。
   b. 将:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
```

   更改为:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="../w3/tr/_2002/rec_xmldsig_core_20020212/xmldsig-core-schema.xsd"/>
```

6. 清理并重建项目。

传输头传播:

处理 SOAP 消息时，您可能需要访问所接收到的消息中的某些传输头中的信息、确保包含传输头的消息带有特定值发送或者允许传输头通过模块传递。

在 Integration Designer 中配置 Web Service 绑定时，您可以指示想要传播传输头。

- 在导出处接收到请求或者在导入处接收到响应时，您可以访问传输头信息，这允许模块中的逻辑基于头值并允许修改这些头。

- 从导出发送响应或者从导入发送请求时，传输头可以包括在这些消息中。

指定传播头

要对导入或导出配置传输头传播，请执行下列步骤:

1. 在 Integration Designer 的“属性”视图中，选择**绑定 → 传播**。
2. 设置所需要的传输头传播选项。

注: 缺省情况下，传输头传播处于禁用状态，并且只能部署到 V7.0.0.3（或更高版本）运行时环境中。另外，还请注意，对于 V7.0.0.3，传输头传播仅限于 HTTP 传输头。

如果启用传输头传播，那么所接收到的消息中的头将通过模块传播，并且如果您未显式除去这些头，那么这些头将在同一线程中的后续调用中使用。

注: 如果您正在使用 Web Service (JAX-RPC) 绑定，那么无法传播传输头。

访问头信息

对所接收到的消息启用传输头传播后，所有传输头（包括客户定义的头）都将在服务消息对象 (SMO) 中可视。您可以将这些头设置为不同的值，也可以创建新的头。但是，请注意，不会对您设置的值进行检查或验证，而任何不恰当或不正确的头都可能导致 Web Service 运行时问题。

请考虑以下关于 HTTP 头设置的信息:

- 对于为 Web Service 引擎保留的头，出站消息中将不接受对这些头所作的任何更改。例如，HTTP 版本或方法、Content-Type、Content-Length 和 SOAPAction 头是为 Web Service 引擎保留的头。
- 如果头值是一个数字，那么应该直接设置该数字（而不是设置字符串）。例如，使用 **Max-Forwards = 5**（而不是 **Max-Forwards = Max-Forwards: 5**）和 **Age = 300**（而不是 **Age = Age: 300**）。
- 如果请求消息的大小小于 32 KB，那么 Web Service 引擎将除去 Transfer-Encoding 头并改为将 Content-Length 头设置为固定消息大小。
- Content-language 将在响应路径中由 WAS.channel.http 重置。
- 无效的 Upgrade 设置将产生 500 错误。
- 下列头将 Web Service 引擎所保留的值追加至客户设置:
  - User-Agent
  - Cache-Control
  - Pragma
  - Accept
  - Connection

您可以使用下列其中一种方式访问头信息:

- 使用调解原语来访问 SMO 结构

  请参阅“相关信息”链接以查找有关使用调解原语的信息。

- 使用上下文服务 SPI

  以下样本代码从上下文服务中读取 HTTP 传输头:

```
HeadersType headerType = ContextService.INSTANCE.getHeaders();
HTTPHeaderType httpHeaderType = headerType.getHTTPHeader();
List HTTPHeader  httpHeaders = httpHeaderType.getHeader();
if(httpHeaders!=null){
```

```
 for(HTTPHeader httpHeader: httpHeaders){
   String httpHeadername = httpHeader.getName();
   String httpHeaderValue = httpHeader.getValue();
     }
}
List PropertyType  properties = headerType.getProperties();
if(properties!=null){
  for(PropertyType property: properties){
   String propertyName = property.getName();
   String propertyValue = property.getValue().toString();
     }
}
```

故障诊断

如果您在发送已修改的头时遇到问题，那么可以通过使用 Integration Designer 中的 TCP/IP 监视器之类的工具来拦截 TCP/IP 消息。可以通过从“首选项”页面中选择**运行/调试 → TCP/IP 监视器**来访问 TCP/IP 监视器。

另外，您也可以使用 JAX-WS 引擎跟踪来查看头值：**org.apache.axis2.*=all: com.ibm.ws.websvcs.*=all:**。

使用 Web Service (JAX-WS) 绑定:

将 Web Service (JAX-WS) 绑定与应用程序配合使用时，可以对此绑定添加安全性断言标记语言 (SAML) 服务质量 (QOS)。必须先使用管理控制台来导入策略集。另外，还可以使用管理控制台来确保正确配置服务器以使用 Web Service (JAX-WS) 绑定。

导入 SAML 策略集:

安全性断言标记语言 (SAML) 是用于交换用户身份和安全性属性信息的基于 XML 的 OASIS 标准。在 Integration Designer 中配置 Web Service (JAX-WS) 绑定时，可指定 SAML 策略集。先使用 IBM Business Process Manager 的管理控制台以提供 SAML 策略集，以便可将它们导入到 Integration Designer 中。

SAML 策略集通常位于概要文件配置目录:

profile_root/config/templates/PolicySets

开始此过程之前，请验证以下目录（这些目录包含策略集）是否位于概要文件配置目录:

- SAML11 Bearer WSHTTPS default
- SAML20 Bearer WSHTTPS default
- SAML11 Bearer WSSecurity default
- SAML20 Bearer WSSecurity default
- SAML11 HoK Public WSSecurity default
- SAML20 HoK Public WSSecurity default
- SAML11 HoK Symmetric WSSecurity default
- SAML20 HoK Symmetric WSSecurity default
- Username WSHTTPS default

如果这些目录不在概要文件配置目录中，请从以下位置将它们复制到该目录:

app_server_root/profileTemplates/default/documents/config/templates/PolicySets

将策略集导入到管理控制台中，选择要向 Integration Designer 提供的策略集，然后将对应每个策略集的一个 zip 文件保存至 Integration Designer 可访问的位置。

1. 通过遵循以下步骤来导入策略集:
   a. 在管理控制台中，单击**服务 → 策略集 → 应用程序策略集**。
   b. 单击**导入 → 从缺省存储库**。
   c. 选择 SAML 缺省策略集，然后单击**确定**。
2. 导出策略集以便 Integration Designer 可使用这些策略集:
   a. 从“应用程序策略集”页面中选择要导出的 SAML 策略集，然后单击**导出**。

      注: 如果当前未显示“应用程序策略集”页面，请从管理控制台单击**服务 → 策略集 → 应用程序策略集**。
   b. 在下一页上，单击该策略集的 .zip 文件链接。
   c. 在“文件下载”窗口中，单击**保存**并指示 Integration Designer 可访问的位置。
   d. 单击**返回**。
   e. 对要导出的每个策略集完成步骤 2a 到 2d。

SAML 策略集保存在 .zip 文件中，并且已准备好导入到 Integration Designer 中。

按“策略集”主题中的描述将策略集导入到 Integration Designer 中。

调用需要进行 HTTP 基本认证的 Web Service:

HTTP 基本认证利用用户名和密码向安全断点认证服务客户机。发送或接收 Web Service 请求时，可以设置 HTTP 基本认证。

通过配置“针对 XML Web Service 的 Java API”(JAX-WS) 导出绑定为接收 Web Service 请求设置 HTTP 基本认证，如“对 Web Service 导出创建和指定安全角色”主题中所述。

可以采用下面的一种或两种方法对 JAX-WS 导入绑定所发送的 Web Service 请求启用 HTTP 基本认证:

- 在 SCA 模块中配置导入绑定时，可以选择所提供的名为 BPMHTTPBasicAuthentication 的 HTTP 认证策略集（此策略集随 Web Service (JAX-WS) 导入绑定一起提供），或者选择任何其他包括 HTTPTransport 策略的策略集。
- 构造 SCA 模块时，可以使用调解流功能来动态创建新的 HTTP 认证头，并在此头中指定用户名和密码。

注: 策略集优先于在此头中指定的值。如果您想要在运行时使用 HTTP 认证头中设置的值，请勿连接其中包括 HTTPTransport 策略的策略集。具体来说，请勿使用缺省 BPMHTTPBasicAuthentication 策略集；如果您已经定义了策略集，那么务必使它排除 HTTPTransport 策略。

可以在 WebSphere Application Server 信息中心的“Web Service 策略集”主题中找到有关 Web Service 策略集和策略绑定及其使用方式的更多信息。指向此主题的链接列示在“相关信息”下。

- 要使用所提供的策略集，请执行下列步骤:
  1. 可选: 在管理控制台中，创建客户机常规策略绑定或者编辑提供了 HTTPTransport 策略以及期望的用户标识和密码值的现有策略绑定。
  2. 在 IBM Integration Designer 中，生成 Web Service (JAX-WS) 导入绑定并连接 BPMHTTPBasicAuthentication 策略集。
  3. 请执行下列*其中一个*步骤:
     – 在 IBM Integration Designer 中，从 Web Service (JAX-WS) 导入绑定属性指定其中包括 HTTPTransport 策略的现有客户机常规策略绑定的名称。
     – 部署 SCA 模块之后，从流程服务器的管理控制台中选择现有客户机策略绑定，或者创建新的客户机策略绑定并使它与导入绑定关联。

4. 可选: 在流程服务器的管理控制台中，编辑所选择的策略集绑定以指定必需的标识和密码。

- 要在 HTTP 认证头中指定用户名和密码，请执行下列其中一组步骤:
  - 在 IBM Integration Designer 中使用调解原语“HTTP 头设置器”来创建 HTTP 认证头，并指定用户名和密码。
  - 如果需要其他逻辑，请在定制调解原语中使用 Java 代码（如以下示例中所示）来执行下列操作:
    1. 创建 HTTP 认证头。
    2. 指定用户名和密码信息。
    3. 将新的 HTTP 认证头添加至 HTTPControl。
    4. 在“上下文”服务中重新设置已更新的 HTTPControl。

```
//Get the HeaderInfoType from contextService
ContextService contextService = (ContextService) ServiceManager.INSTANCE
.locateService("com/ibm/bpm/context/ContextService");
HeaderInfoType headers = contextService.getHeaderInfo();
if(headers == null){
    headers = ContextObjectFactory.eINSTANCE.createHeaderInfoType();
}
//Get the HTTP header and HTTP Control from HeaderInfoType
HTTPHeaderType httpHeaderType = headers.getHTTPHeader();
HTTPControl cp = httpHeaderType.getControl();
HeadersFactory factory = HeadersFactory.eINSTANCE;
if(cp == null){
    cp = factory.createHTTPControl();
}
//Create new HTTPAuthentication and set the HTTPCredentials
HTTPAuthentication authorization = factory.createHTTPAuthentication();
HTTPCredentials credentials = factory.createHTTPCredentials();
authorization.setAuthenticationType(HTTPAuthenticationType.BASIC_LITERAL);
credentials.setUserId("USERNAME");
credentials.setPassword("PASSWORD");
authorization.setCredentials(credentials);
cp.setAuthentication(authorization);
httpHeaderType.setControl(cp);
// Set header info back to the current execution context.
contextService.setHeaderInfo(headers);
```

检查服务器配置:

部署具有 Web Service (JAX-WS) 绑定的应用程序时，必须确保部署该应用程序的服务器未选择**根据需要启动组件**选项。

可通过从管理控制台执行以下步骤来检查是否选择了此选项:

1. 单击**服务器 → 服务器类型 → WebSphere Application Server**。
2. 单击服务器名称。
3. 在“配置”选项卡中，确定是否选择了**根据需要启动组件**。
4. 执行下列其中一个步骤:
   - 如果选择了**根据需要启动组件**，请除去选中标记，然后单击**应用**。
   - 如果未选择**根据需要启动组件**，请单击**取消**。

***SOAP** 消息中的附件:*

您可以发送和接收包含二进制数据（例如 PDF 文件或 JPEG 图像）作为附件的 SOAP 消息。附件可以是*引用的*附件（即，在服务接口中以消息部件显式表示），也可以*未引用*的附件（在这种情况下，可以包括任意数目和任意类型的附件）。

可以使用下列其中一种方式表示引用的附件:

- 作为消息模式中的 wsi:swaRef-typed 元素

  使用 wsi:swaRef 类型定义的附件遵从 Web Services 互操作性组织 (WS-I) *Attachments Profile V1.0* (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html)，该概要文件定义了消息元素与 MIME 部件的相关方式。
- 作为使用二进制模式类型的顶级消息部件

  以顶级消息部件表示的附件遵从 *SOAP Messages with Attachments* (http://www.w3.org/TR/SOAP-attachments) 规范。

  也可以配置以顶级消息部件表示的附件，以确保绑定所生成的 WSDL 文档和消息遵从 WS-I *Attachments Profile V1.0* 和 WS-I *Basic Profile V1.1* (http://www.ws-i.org/Profiles/BasicProfile-1.1.html)。

未引用的附件在消息模式中没有任何表示就包含在 SOAP 消息中。

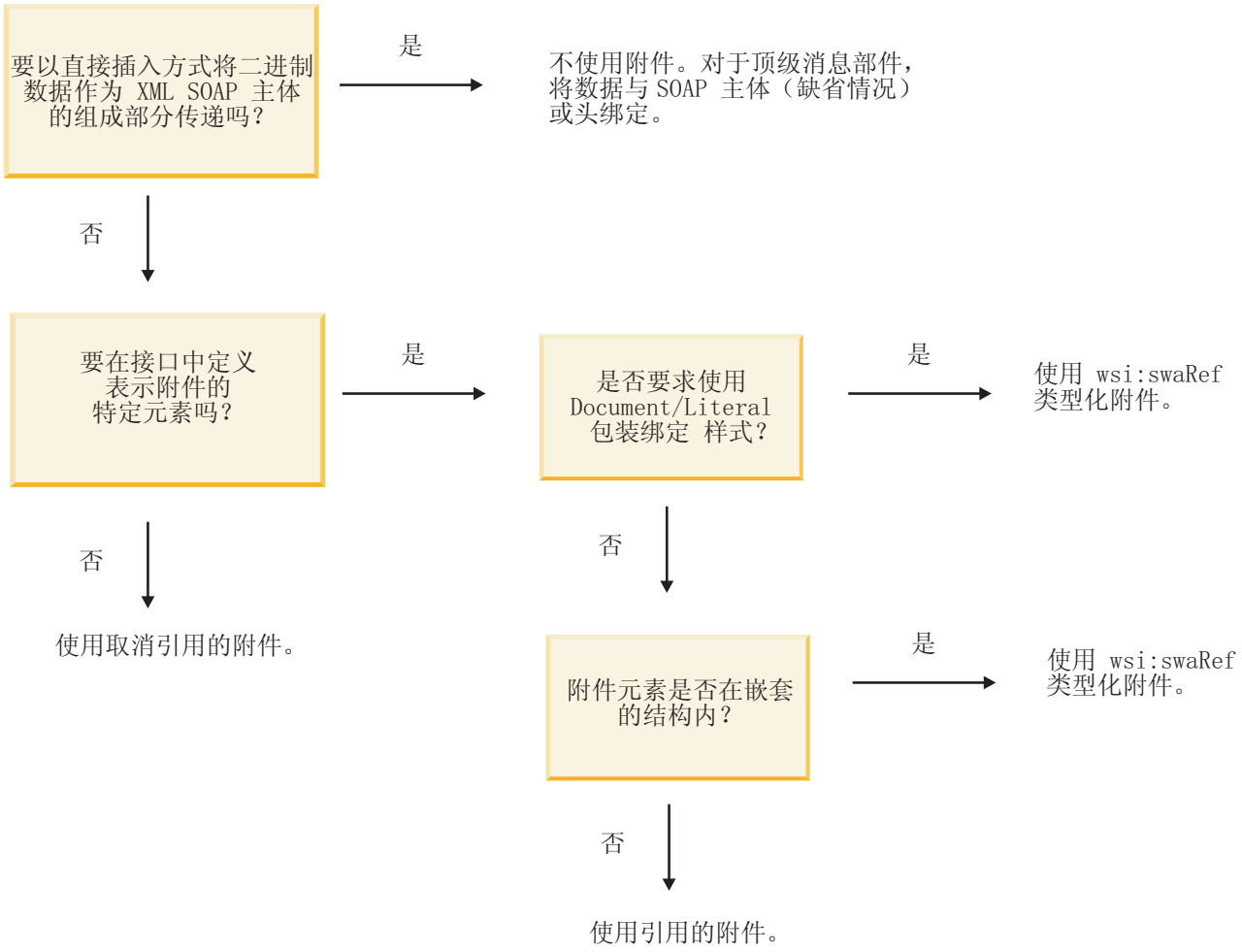
在所有情况下，WSDL SOAP 绑定都应该包括用于要使用的附件的 MIME 绑定，并且附件的最大大小应该不超过 20 MB。

注: 要发送或接收带有附件的 SOAP 消息，您必须使用其中一个基于“针对 XML Web Service 的 Java API” (JAX-WS) 的 Web Service 绑定。

如何选择合适的附件样式:

设计包含二进制数据的新服务接口时，请考虑服务所发送和接收的 SOAP 消息中如何携带二进制数据。

请使用下列一组问题来确定合适的附件样式:



引用的附件: swaRef 类型化元素:

您可以发送和接收包含那些在服务接口中表示为 swaRef 类型化元素的附件的 SOAP 消息。

swaRef 类型化元素是在 Web Services 互操作性组织 (WS-I) *Attachments Profile* V1.0 (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html) 中定义的，后者定义了消息元素与 MIME 部件的相关方式。

在 SOAP 消息中，SOAP 主体包含一个 swaRef 类型化元素，此元素用于标识附件的内容标识。

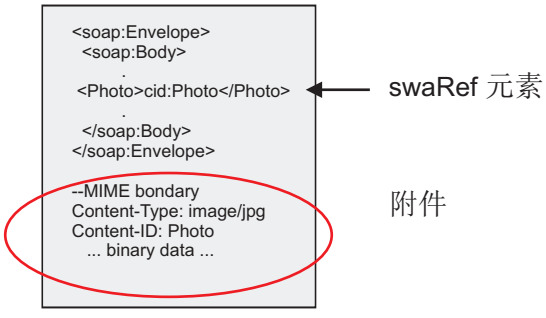


图 13. 包含 swaRef 元素的 SOAP 消息

此 SOAP 消息的 WSDL 在消息部件中包含一个 swaRef 类型化元素，该元素用于标识附件。

```
<element name="sendPhoto">
  <complexType>
     <sequence>
       <element name="Photo" type="wsi:swaRef"/>
     </sequence>
  </complexType>
 </element>
```

WSDL 也应该包含 MIME 绑定，此绑定用于指示所要使用的 MIME 多重部件消息。

注: 由于 MIME 绑定仅应用于顶级消息部件，因此 WSDL 未包含用于特定 swaRef 类型化消息元素的 MIME 绑定。

表示为 swaRef 类型化元素的附件只能跨调解流组件传播。如果某个附件必须由另一组件类型访问或传播到另一组件类型，请使用调解流组件将该附件移至可供该组件访问的位置。

附件的入站处理

请使用 Integration Designer 来配置导出绑定以接收附件。您创建模块及其相关联的接口和操作，包括类型为 swaRef 的元素。然后，创建 Web Service (JAX-WS) 绑定。

注: 有关更多详细信息，请参阅 Integration Designer 信息中心中的“使用附件”主题。

当客户机将带有 swaRef 附件的 SOAP 消息传递到服务组件体系结构 (SCA) 组件时，Web Service (JAX-WS) 导出绑定将先除去该附件。然后，它解析该消息的 SOAP 部件并创建业务对象。最后，此绑定在该业务对象中设置附件的内容标识。

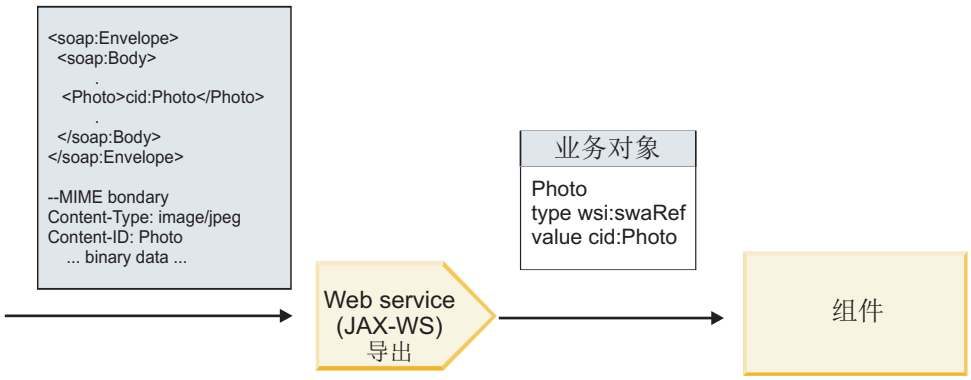


图 14. Web Service (JAX-WS) 导出绑定处理包含 swaRef 附件的 SOAP 消息的方式

在调解流组件中访问附件元数据

如第 61 页的图 15 所示，组件访问 swaRef 附件时，附件内容标识以类型为 swaRef 的元素出现。

SOAP 消息中的每个附件在 SMO 中也都有相应的 **attachments** 元素。使用 WS-I swaRef 类型时，**attachments** 元素包含附件内容类型和内容标识以及该附件的实际二进制数据。

因此，要获取 swaRef 附件的值，必须获取 swaRef 类型化元素的值，然后找到具有相应 **contentID** 值的 **attachments** 元素。注意，**contentID** 值通常具有从 swaRef 值中除去的 **cid:** 前缀。

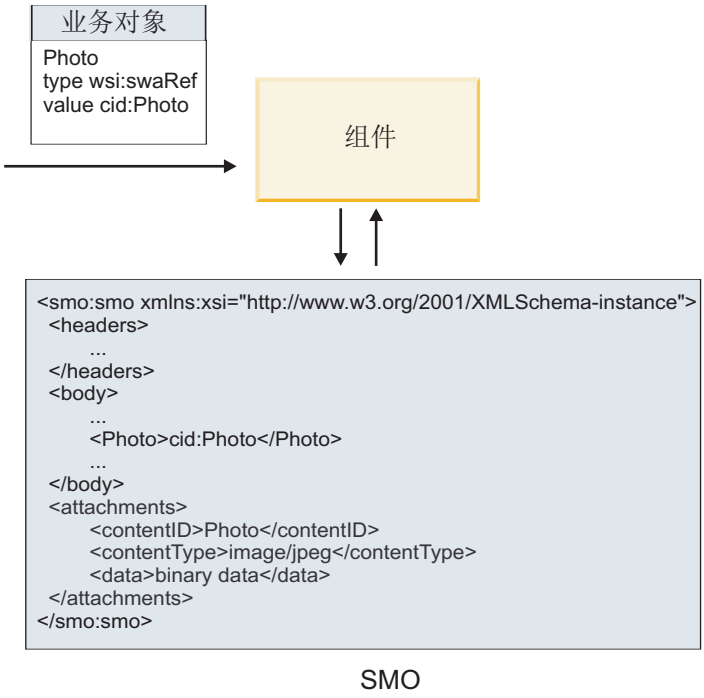


图 15. swaRef 附件在 SMO 中的表示方式

出站处理

请使用 Integration Designer 来配置 Web Service (JAX-WS) 导入绑定以调用外部 Web Service。导入绑定是使用一个 WSDL 文档配置的，该文档描述了所要调用的 Web Service 并定义了将传递到该 Web Service 的附件。

Web Service (JAX-WS) 导入绑定接收到 SCA 消息时，如果该导入连接到调解流组件，并且 swaRef 类型化元素具有相应的 **attachments** 元素，那么 swaRef 类型化元素将作为附件发送。

对于出站处理，swaRef 类型化元素始终与其内容标识值一起发送；但是，调解模块必须确保存在具有匹配 **contentID** 值的相应 **attachments** 元素。

注：为了与 WS-I Attachments Profile 保持一致，**content ID** 值应该遵循 WS-I *Attachments Profile* 1.0 的 3.8 节中描述的“content-id 部件编码”。

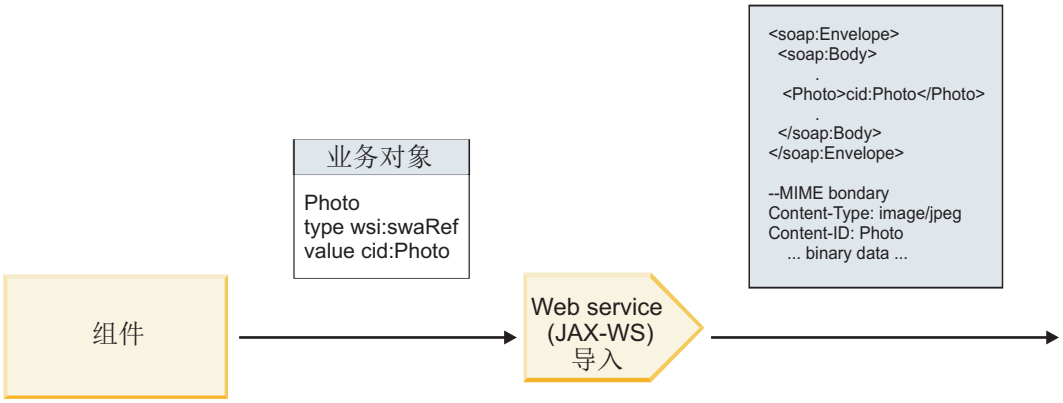


图 16. Web Service (JAX-WS) 导入绑定生成包含 swaRef 附件的 SOAP 消息的方式

在调解流组件中设置附件元数据

在 SMO 中，如果存在 swaRef 类型化元素值和 **attachments** 元素，那么此绑定将准备 SOAP 消息（包含附件）并将其发送到接收方。

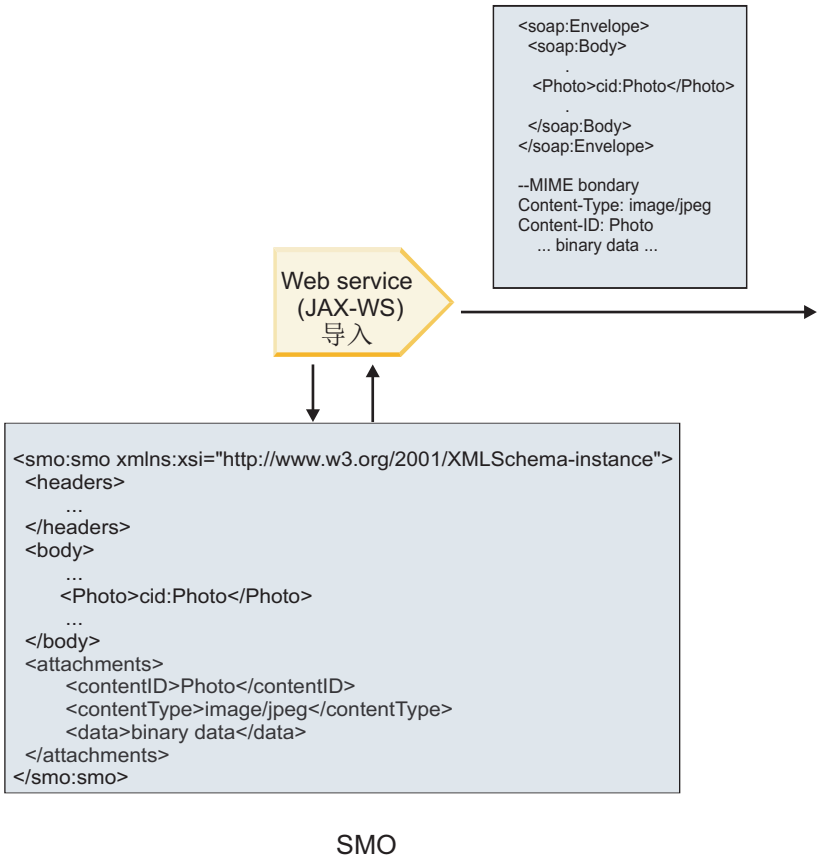


图 17. 如何访问 SMO 中的 swaRef 附件以创建 SOAP 消息

仅当调解流组件直接连接到导入或导出时，SMO 才包含 **attachments** 元素；不会跨其他组件类型传递此元素。如果在包含其他组件类型的模块中需要这些值，那么应使用调解流组件将这些值复制到一个能够从模块中对其进行访问的位置，并在通过 Web Service 导入执行出站调用之前使用另一个调解流组件设置正确的值。

要点: 如“SMO 的 XSL 表示”所述，“XSL 变换”调解原语使用 XSLT 1.0 变换来变换消息。此变换对 SMO 的 XML 序列化执行操作。“XSL 变换”调解原语允许指定序列化根，XML 文档的根元素将反映这个根。

在发送带有附件的 SOAP 消息时，您选择的根元素确定了附件的传播方式。

- 如果使用“/body”作为 XML 映射的根，那么在缺省情况下，所有附件都将通过此映射传播。
- 如果使用“/”作为映射的根，那么您可以控制附件的传播。

引用的附件: 顶级消息部件:

您可以发送和接收包含那些在服务接口中声明为部件的二进制附件的 SOAP 消息。

在 MIME 多重部件 SOAP 消息中，SOAP 主体是消息的第一个部件，而附件包含在后续部件中。

在 SOAP 消息中发送或接收引用的附件有何优点？构成附件的二进制数据（通常非常大）与 SOAP 消息体分开存放，因此不必解析为 XML。相对于在 XML 元素中存放二进制数据，这可以提高处理效率。

具有引用的附件的 SOAP 消息类型

从 IBM Business Process Manager V7.0.0.3 开始，您可以选择 SOAP 消息的生成方式:

- **与 WS-I 一致的消息**

  运行时能够生成与 WS-I *Attachments Profile V1.0* 和 WS-I *Basic Profile V1.1* 一致的 SOAP 消息。在与这些概要一致的 SOAP 消息中，只有一个消息部件与 SOAP 主体绑定；对于那些作为附件绑定的部件来说，将使用 WS-I *Attachments Profile V1.0* 中描述的 content-id 部件编码使附件与消息部件相关。

- **与 WS-I 不一致的消息**

  运行时能够生成与 WS-I 概要不一致但与 IBM Business Process Manager V7.0 或 V7.0.0.2 中生成的消息兼容的 SOAP 消息。这些 SOAP 消息使用了顶级元素（这些元素根据相应 **href** 属性包含附件 **content-id** 的消息部件命名），但不使用 WS-I *Attachments Profile V1.0* 中描述的 content-id 部件编码。

为 Web Service 导出选择 WS-I 一致性

请使用 Integration Designer 来配置导出绑定。您创建模块及其相关联的接口和操作。然后，创建 Web Service (JAX-WS) 绑定。“引用的附件”页面将显示所创建的操作中的所有二进制部件，您可以选择哪些部件将是附件。然后，在 Integration Designer 的“指定 WS-I AP 1.0 一致性 ”页面上指定下列其中一个选项:

- **使用与 WS-I AP 1.0 一致的 SOAP 消息**

  如果选择此选项，那么还需指定应该与 SOAP 主体绑定的消息部件。

  注: 仅当相应的 WSDL 文件也与 WS-I 一致时，才能使用此选项。

  由 Integration Designer V7.0.0.3 生成的 WSDL 文件与 WS-I 一致。但是，如果导入与 WS-I 不一致的 WSDL 文件，那么无法选择此选项。

- **使用与 WS-I AP 1.0 不一致的 SOAP 消息**

  如果选择此选项（这是缺省选项），那么第一个消息部件将与 SOAP 主体绑定。

注: 只有具有二进制类型（base64Binary 或 hexBinary）的顶级消息部件（即，在 WSDL portType 中定义为输入消息部件或输出消息部件的元素）才能作为引用的附件进行发送或接收。
有关更多详细信息，请参阅 Integration Designer 信息中心中的“使用附件”主题。

对于与 WS-I 一致的消息，在 SOAP 消息中生成的 content-ID 是下列元素的并置:

- **mime:content** 所引用的 **wsdl:part** 元素的 **name** 属性的值
- 字符 **=**
- 全局唯一值，例如 UUID
- 字符 **@**
- 有效的域名

引用的附件的入站处理

当客户机将带有附件的 SOAP 消息传递到服务组件体系结构 (SCA) 组件时，Web Service (JAX-WS) 导出绑定将先除去该附件。然后，它解析该消息的 SOAP 部件并创建业务对象。最后，此绑定在该业务对象中设置附件二进制内容。

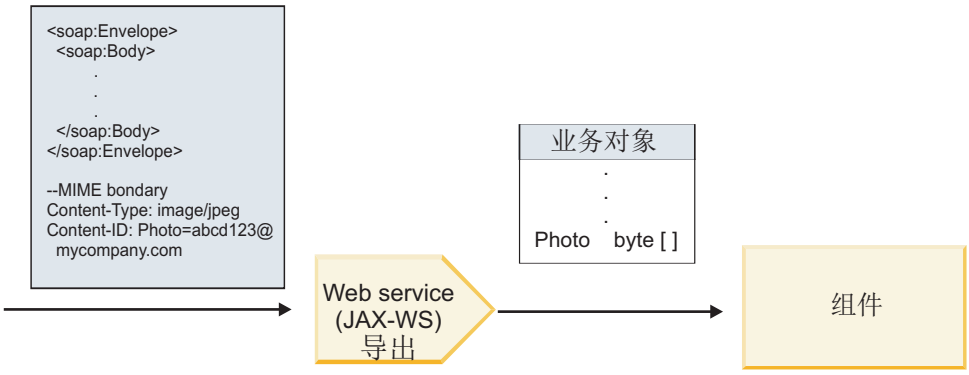


图 18. Web Service (JAX-WS) 导出绑定处理带有所引用附件并与 WS-I 一致的 SOAP 消息的方式

在调解流组件中访问附件元数据

如图 18 所示，组件访问引用的附件时，附件数据以字节数组形式出现。

SOAP 消息中的每个所引用附件在 SMO 中也都有相应的 **attachments** 元素。**attachments** 元素包含附件内容类型以及用于存放该附件的消息体元素的路径。

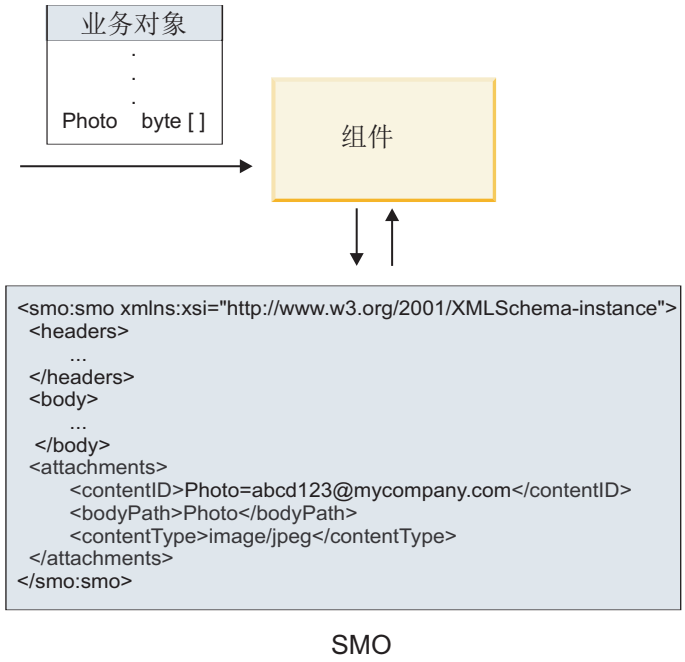


图 19. 引用的附件在 SMO 中的表示方式

要点: 如果对消息进行了变换并移动了附件，那么不会自动更新消息体元素的路径。您可以使用调解流来更新 **attachments** 元素以便对其指定新路径（例如，在变换过程中进行更新或者使用独立的消息元素设置器进行更新）。

出站 SOAP 消息的构造

请使用 Integration Designer 来配置 Web Service (JAX-WS) 导入绑定以调用外部 Web Service。导入绑定是使用一个 WSDL 文档配置的，该文档描述了所要调用的 Web Service 并定义了应该作为附件传递的消息部件。另外，还可以在 Integration Designer 的“指定 WS-I AP 1.0 一致性 ”页面上指定下列其中一个选项:

- **使用与 WS-I AP 1.0 一致的 SOAP 消息**

  如果选择此选项，那么还需指定应该与 SOAP 主体绑定的消息部件；所有其他消息部件都与附件或头绑定。此绑定所发送的消息不包含 SOAP 主体中引用了附件的元素；此关系由包含消息部件名的附件内容标识表示。

- **使用与 WS-I AP 1.0 不一致的 SOAP 消息**

  如果选择此选项（这是缺省选项），那么第一个消息部件将与 SOAP 主体绑定；所有其他消息部件都与附件或头绑定。此绑定发送的消息包含 SOAP 主体中一个或多个通过 **href** 属性引用了附件的元素。

注: 用于表示附件的部件（在 WSDL 中定义）必须是简单类型（base64Binary 或 hexBinary）。如果某个部件由 complexType 定义，那么无法将该部件绑定为附件。

引用的附件的出站处理

导入绑定使用 SMO 中的信息来确定如何将二进制顶级消息部件作为附件发送。

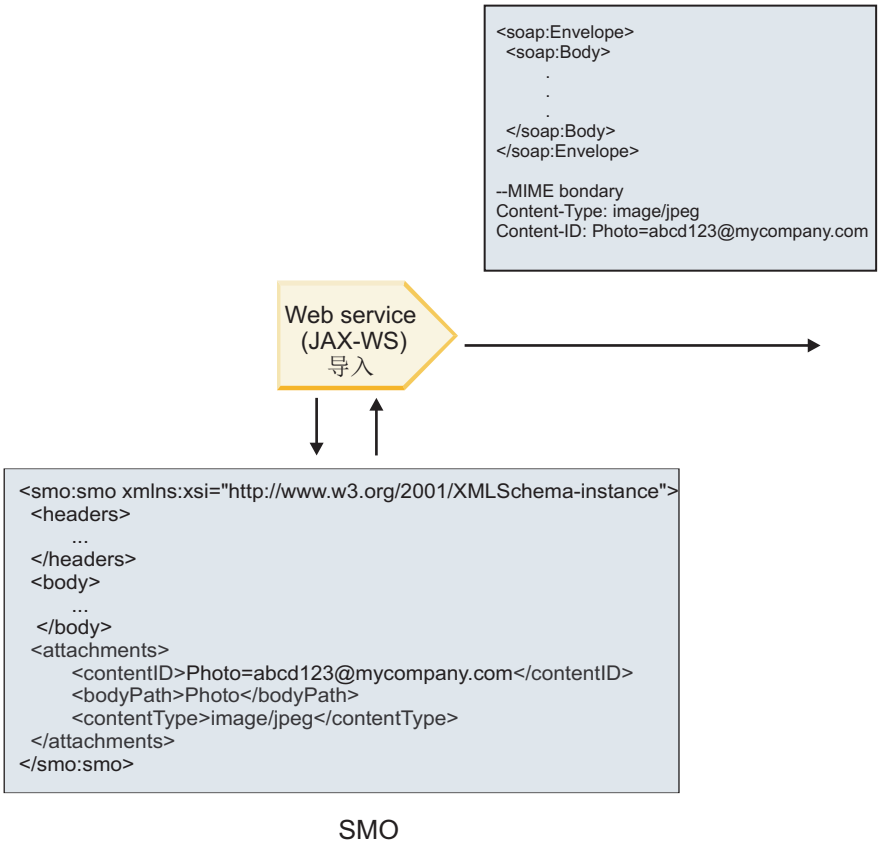


图 20. 如何访问 SMO 中的所引用附件以创建 SOAP 消息

仅当调解流组件直接连接到导入或导出时，SMO 才包含 **attachments** 元素；不会跨其他组件类型传递此元素。如果在包含其他组件类型的模块中需要这些值，那么应使用调解流组件将这些值复制到一个能够从模块中对其进行访问的位置，并在通过 Web Service 导入执行出站调用之前使用另一个调解流组件设置正确的值。

此绑定使用下列条件的组合来确定如何（或者是否）发送该消息:

- 是否存在顶级二进制消息部件的 WSDL MIME 绑定，如果存在，其内容类型如何定义
- 在 SMO 中是否存在其 **bodyPath** 值引用了顶级二进制部件的 **attachments** 元素

SMO 中存在 attachment 元素时创建附件的方式

下表列示 SMO 包含 **bodyPath** 与消息名称部件匹配的 **attachment** 元素时创建和发送附件的方式:

表 20. 附件的生成方式

顶级二进制消息部件的 WSDL MIME 绑定的状态	消息的创建和发送方式
存在，并且下列其中一个条件成立: • 没有为消息部件定义内容类型 • 定义了多种内容类型 • 定义了通配内容类型	将消息部件作为附件发送。 如果存在 attachments 元素，那么 **Content-Id** 设置为该元素中的值；否则，生成一个值。 如果存在 attachments 元素，那么 **Content-Type** 设置为该元素中的值；否则，设置为 **application/octet-stream**。

表 20. 附件的生成方式 (续)

顶级二进制消息部件的 WSDL MIME 绑定的状态	消息的创建和发送方式
存在，并且消息部件具有单一的非通配内容	将消息部件作为附件发送。 如果存在 attachments 元素，那么 **Content-Id** 设置为该元素中的值；否则，生成一个值。 如果存在 attachments 元素，那么 **Content-Type** 设置为该元素中的值；否则，设置为 WSDL MIME content 元素中定义的类型。
不存在	将消息部件作为附件发送。 如果存在 attachments 元素，那么 **Content-Id** 设置为该元素中的值；否则，生成一个值。 如果存在 attachments 元素，那么 **Content-Type** 设置为该元素中的值；否则，设置为 **application/octet-stream**。 **注**: 在 WSDL 未将消息部件定义为附件的情况下将这些部件作为附件发送可能会违反 WS-I Attachments Profile 1.0，因此应尽可能避免。

SMO 中不存在 attachment 元素时创建附件的方式

下表列示 SMO 未包含 **bodyPath** 与消息名称部件匹配的 **attachment** 元素时创建和发送附件的方式:

表 21. 附件的生成方式

顶级二进制消息部件的 WSDL MIME 绑定的状态	消息的创建和发送方式
存在，并且下列其中一个条件成立: • 没有为消息部件定义内容类型 • 定义了多种内容类型 • 定义了通配内容类型	将消息部件作为附件发送。 生成 **Content-Id**。 **Content-Type** 设置为 **application/octet-stream**。
存在，并且消息部件具有单一的非通配内容	将消息部件作为附件发送。 生成 **Content-Id**。 **Content-Type** 设置为 WSDL MIME content 元素中定义的类型。
不存在	不将消息部件作为附件发送。

要点: 如“SMO 的 XSL 表示”所述，“XSL 变换”调解原语使用 XSLT 1.0 变换来变换消息。此变换对 SMO 的 XML 序列化执行操作。“XSL 变换”调解原语允许指定序列化根，XML 文档的根元素将反映这个根。

在发送带有附件的 SOAP 消息时，您选择的根元素确定了附件的传播方式。

- 如果使用“/body”作为 XML 映射的根，那么在缺省情况下，所有附件都将通过此映射传播。
- 如果使用“/”作为映射的根，那么您可以控制附件的传播。

未引用的附件:

您可以发送和接收未在服务接口中声明的*未引用的*附件。

在 MIME 多重部件 SOAP 消息中，SOAP 主体是消息的第一个部件，而附件包含在后续部件中。SOAP 主体中不包括对附件的引用。

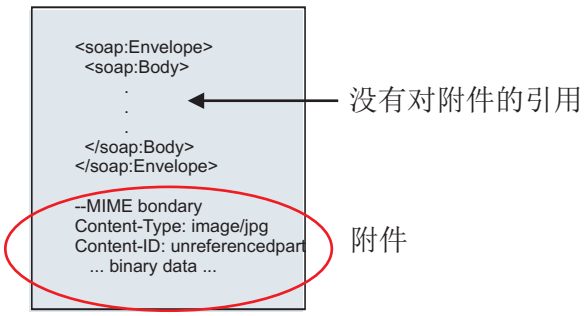


图 21. 带有未引用的附件的 SOAP 消息

您可以通过 Web Service 导出向 Web Service 导入发送带有未引用的附件的 SOAP 消息。向目标 Web Service 发送的输出消息包含附件。

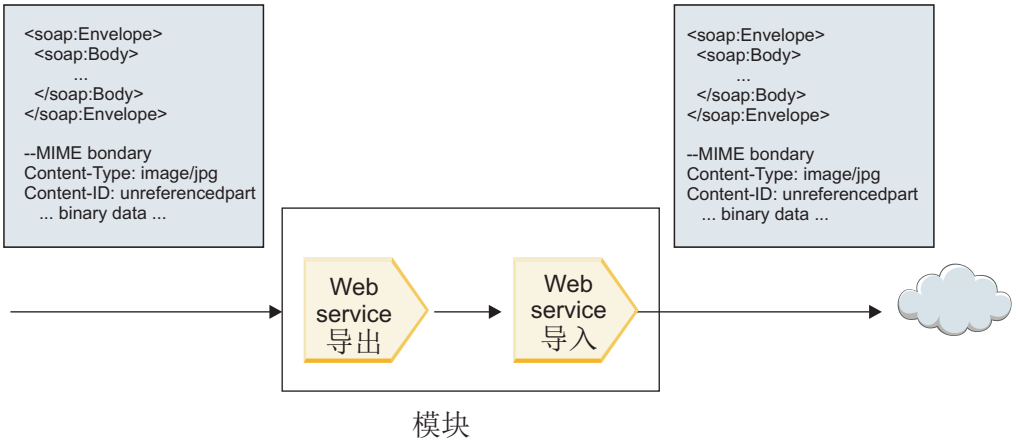


图 22. 通过 SCA 模块传递的附件

在图 22 中，带有附件的 SOAP 消息通过时未修改。

您也可以通过使用调解流组件来修改 SOAP 消息。例如，可以使用调解流组件从 SOAP 消息中抽取数据（在本示例中，这是指消息主体中的二进制数据），并创建带有附件消息的 SOAP。这些数据将作为服务消息对象 (SMO) 的附件元素的组成部分进行处理。

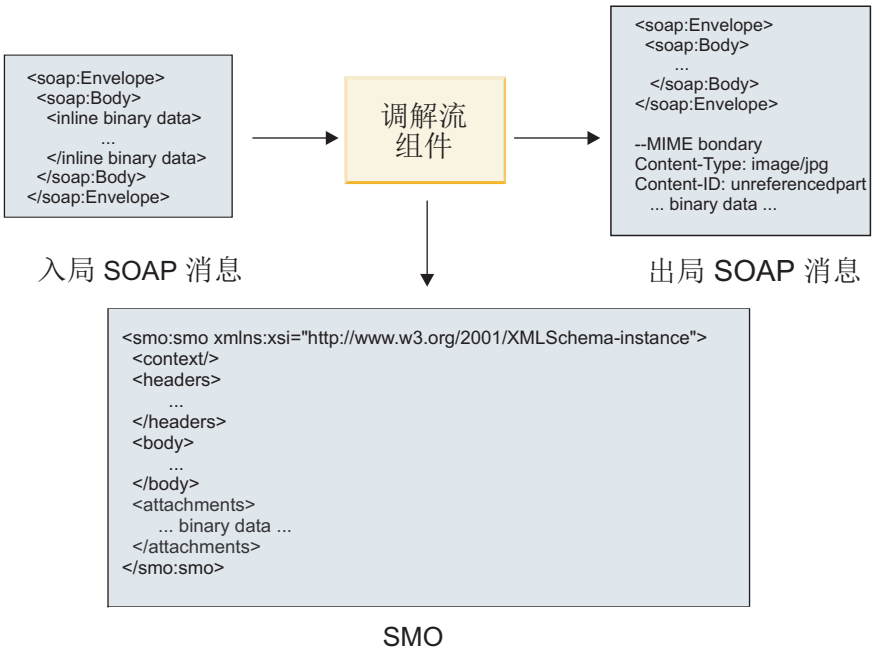


图 23. 由调解流组件处理的消息

相反，调解流组件可以通过抽取并编码附件，然后传输不带有附件的消息来变换入局消息。

您可以从数据库之类的外部源中获取附件数据，而不是从入局 SOAP 消息中抽取数据来构成带有附件消息的 SOAP。

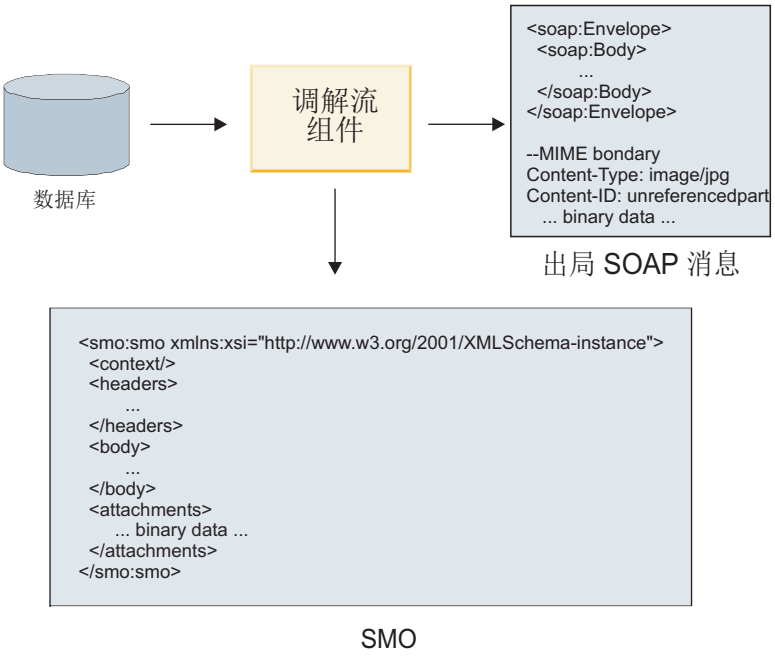


图 24. 从数据库中获取并添加至 SOAP 消息的附件

相反，调解流组件可以从入局 SOAP 消息中抽取附件并处理此消息（例如，将附件存储在数据库中）。

未引用的附件只能通过调解流组件传播。如果某个附件必须由另一组件类型访问或传播到另一组件类型，请使用调解流组件将该附件移至可供该组件访问的位置。

要点: 如“SMO 的 XSL 表示”所述，“XSL 变换”调解原语使用 XSLT 1.0 变换来变换消息。此变换对 SMO 的 XML 序列化执行操作。“XSL 变换”调解原语允许指定序列化根，XML 文档的根元素将反映这个根。

在发送带有附件的 SOAP 消息时，您选择的根元素确定了附件的传播方式。

- 如果使用“/body”作为 XML 映射的根，那么在缺省情况下，所有附件都将通过此映射传播。
- 如果使用“/”作为映射的根，那么您可以控制附件的传播。

将 WSDL 文档样式绑定与多重部件消息配合使用:

Web Services 互操作性组织 (WS-I) 组织定义了一组规则来指示应该如何通过 WSDL 描述 Web Service 以及如何构造相应 SOAP 消息以确保互操作性。

这些规则是在 WS-I *Basic Profile V1.1* (http://www.ws-i.org/Profiles/BasicProfile-1.1.html) 中指定的。尤其是，WS-I Basic Profile 1.1 R2712 指出: “Document/Literal 绑定必须序列化为带有 soap:Body 的包络，其中，soap:Body 的子元素是相应 wsdl:message 部件所引用的全局元素声明的实例”。

这意味着，将文档样式 SOAP 绑定用于带有多重部件消息（输入消息、输出消息或故障消息）的操作时，只有其中一个部件应该与 SOAP 主体绑定才能与 WS-I Basic Profile 1.1 保持一致。

另外，WS-I Attachments Profile 1.0 R2941 指出: “描述中的 wsdl:binding 应该将它引用的 wsdl:portType 中 wsdl:message 的每个 wsdl:part 与 soapbind:body、soapbind:header、soapbind:fault、soapbind:headerfault 或 mime:content 绑定”。

这意味着，将文档样式 SOAP 绑定用于带有多重部件消息（输入消息、输出消息或故障消息）的操作时，除了选择与 SOAP 主体绑定的部件以外的所有部件都必须作为附件或头进行绑定。

在这种情况下，为具有 Web Service（JAX-WS 和 JAX-RPC）绑定的导出生成 WSDL 描述时，将使用以下方法:

- 如果存在多个非二进制类型化元素，那么您可以选择要与 SOAP 主体绑定的消息部件。如果存在单一非二进制类型化元素，那么该元素将自动与 SOAP 主体绑定。
- 对于 JAX-WS 绑定而言，所有其他具有“hexBinary”或“base64Binary”类型的消息部件都将作为所引用附件绑定。请参阅第 63 页的『引用的附件: 顶级消息部件』。
- 所有其他消息部件都将作为 SOAP 头绑定。

JAX-RPC 和 JAX-WS 导入绑定采用现有 WSDL 文档中具有多重部件文档样式消息的 SOAP 绑定，即使它未将多个部件与 SOAP 主体绑定也是如此；但是，无法在 Rational Application Developer 中为此类 WSDL 文档生成 Web Service 客户机。

注: JAX-RPC 绑定不支持附件。

因此，将多重部件消息与具有文档样式 SOAP 绑定的操作配合使用时，建议的模式如下所示:

1. 使用 Document/Literal 包装样式。在这种情况下，消息始终包含单一部件；但是，在这种情况下，必须将附件取消引用（如第 67 页的『未引用的附件』所述）或者将其 swaRef 类型化（如第 59 页的『引用的附件: swaRef 类型化元素』所述）。
2. 使用 RPC/Literal 样式。在这种情况下，在与 SOAP 主体绑定的部件数方面对 WSDL 绑定没有限制；生成的 SOAP 消息始终具有用于表示所调用操作的单一子代，并且消息部件是该元素的子代。
3. 对于 JAX-WS 绑定而言，除非可以将其他非二进制部件与 SOAP 头绑定，否则最多只能有一个消息部件的类型不是“hexBinary”或“base64Binary”。
4. 所有其他情况均遵从描述的行为。

注: 在使用与 WS-I *Basic Profile Version 1.1* 不一致的 SOAP 消息时，存在其他限制。

- 第一个消息部件应该是非二进制消息部件。
- 在接收具有所引用附件的多重部件文档样式 SOAP 消息时，JAX-WS 绑定期望每个所引用附件都由 SOAP 主体子元素表示，并期望该元素的 href 属性值按附件的内容标识来标识该附件。JAX-WS 绑定以同一方式发送此类消息的所引用附件。此行为与 WS-I Basic Profile 不一致。

  要确保消息与 Basic Profile 一致，请采用以上列表中的方法 第 70 页的1 或 第 70 页的2，或者避免将所引用附件用于此类消息并改为使用取消引用的附件或 swaRef 类型化附件。

HTTP 绑定

HTTP 绑定旨在提供面向 HTTP 的服务组件体系结构 (SCA) 连通性。因此，现有或新开发的 HTTP 应用程序可以参与面向服务的体系结构 (SOA) 环境。

超文本传输协议 (HTTP) 是一个广泛使用的协议，用于在 Web 上传输信息。处理使用 HTTP 协议的外部应用程序时，将需要 HTTP 绑定。HTTP 绑定用于将以本机格式的消息形式传入的数据变换成 SCA 应用程序中的业务对象。另外，HTTP 绑定还可以将以业务对象形式出局数据变换成外部应用程序对入局消息所期望的本机格式。

注: 如果您想要与使用 Web Service SOAP/HTTP 协议的客户机和服务交互，请考虑使用其中一个 Web Service 绑定，这些绑定提供关于处理 Web Service 标准服务质量的附加功能。

以下列表描述了一些使用 HTTP 绑定的常见场景:

- 主管 SCA 的服务可以使用 HTTP 导入调用 HTTP 应用程序。
- 主管 SCA 的服务可以将本身作为支持 HTTP 的应用程序公开，以便 HTTP 客户机可以使用 HTTP 导出来使用这些服务。
- IBM Business Process Manager 和 Process Server 可以跨 HTTP 基础结构相互通信，因此，用户可以按照企业标准管理其通信。
- IBM Business Process Manager 和 Process Server 可以充当 HTTP 通信的介体并变换和路由消息，这提高了使用 HTTP 网络的应用程序的集成度。
- IBM Business Process Manager 和 Process Server 可用于桥接 HTTP 与其他协议，例如 SOAP/HTTP Web Service、基于 Java 连接器体系结构 (JCA) 的资源适配器、JMS 等等。

您可以在 Integration Designer 信息中心中找到有关创建 HTTP 导入和导出绑定的详细信息。请参阅**开发集成应用程序 → 使用 HTTP 访问外部服务**主题。

HTTP 绑定概述:

HTTP 绑定提供面向主管 HTTP 的应用程序的连通性。HTTP 绑定调解 HTTP 应用程序之间的通信并允许从模块调用现有基于 HTTP 的应用程序。

HTTP 导入绑定

HTTP 导入绑定提供从服务组件体系结构 (SCA) 应用程序到 HTTP 服务器或应用程序的出站连接。

该导入调用一个 HTTP 端点 URL。可以使用下列三种方式中的一种指定该 URL:

- 可以通过使用动态覆盖 URL 在 HTTP 头中动态设置该 URL。
- 可以在 SMO 目标地址元素中动态设置该 URL。
- 可以将该 URL 指定为导入中的配置属性。

此调用本质上始终是同步调用。

虽然 HTTP 调用始终是请求-应答模式，但 HTTP 导入不仅支持单向操作，还支持双向操作，并且在单向操作的情况下将忽略响应。

HTTP 导出绑定

HTTP 导出绑定提供从 HTTP 应用程序到 SCA 应用程序的入站连接。

在 HTTP 导出中定义了一个 URL。想要向该导出发送请求消息的 HTTP 应用程序使用此 URL 来调用该导出。

HTTP 导出还支持 Ping 操作。

运行时的 HTTP 绑定

在运行时具有 HTTP 绑定的导入向外部 Web Service 发送来自 SCA 应用程序的请求（该请求的消息主体中可能包含数据，也可能不包含数据）。该请求是 SCA 应用程序向外部 Web Service 发出的，如图 25 所示。

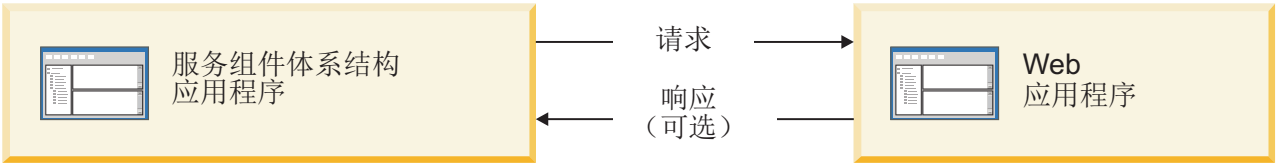


图 25. 从 SCA 应用程序到 Web 应用程序的请求流。

（可选）具有 HTTP 绑定的导入可以接收到 Web 应用程序为响应该请求而发送回的数据。

对于导出，该请求是客户机应用程序向 Web Service 发出的，如图 26 所示。

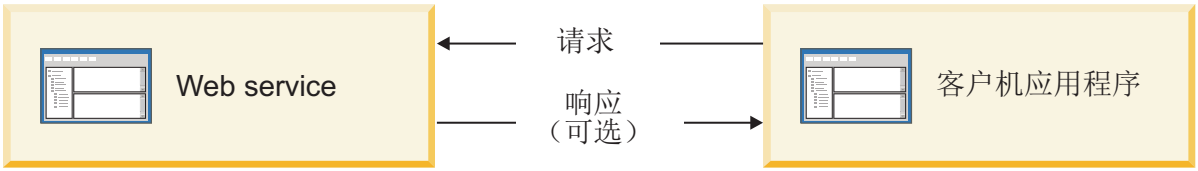


图 26. 从 Web Service 到客户机应用程序的请求流。

Web Service 是正在服务器上运行的 Web 应用程序。导出在该 Web 应用程序中实现为 Servlet，因此，客户机向一个 URL 地址发送其请求。该 Servlet 在运行时将该请求传递给 SCA 应用程序。

（可选）导出可能会向客户机应用程序发送数据以响应该请求。

HTTP 头:

HTTP 导入和导出绑定允许对要用于出站消息的 HTTP 头及其值进行配置。HTTP 导入将这些头用于请求，而 HTTP 导出将这些头用于响应。

以静态方式配置的头和控制信息优先于在运行时动态设置的值。但是，动态覆盖 URL、版本和方法控制值将覆盖静态值，否则这些静态值将被视为缺省值。

通过在运行时确定 HTTP 目标 URL、版本和方法的值，绑定支持 HTTP 导入 URL 的动态性质。这些值是通过抽取端点引用、动态覆盖 URL、版本和方法的值来确定的。

- 对于端点引用，请使用 com.ibm.websphere.sca.addressing.EndpointReference API 或者设置 SMO 头中的 /headers/SMOHeader/Target/address 字段。
- 对于动态覆盖 URL、版本和方法，请使用服务组件体系结构 (SCA) 消息的 HTTP 控制参数部分。请注意，动态覆盖 URL 优先于目标端点引用；但是，端点引用可跨绑定应用，因此，它是首选方法并且应尽可能使用。

将按以下顺序处理 HTTP 导出和导入绑定下的出站消息的控制和头信息:

1. 头和控制信息（不包括 SCA 消息中的 HTTP 动态覆盖 URL、版本和方法）（最低优先级）
2. 管理控制台中对导出/导入级别的更改
3. 管理控制台中对导出或导入的方法级别的更改
4. 通过使用端点引用或 SMO 头指定的目标地址
5. SCA 消息中的动态覆盖 URL、版本和方法
6. 数据处理程序或数据绑定中的头和控制信息（最高优先级）

仅当协议头传播设置为 **True** 时，HTTP 导出和导入才使用入局消息中的数据填充入站方向头和控制参数（HTTPExportRequest 和 HTTPImportResponse）。相反，仅当协议头传播设置为 **True** 时，HTTP 导出和导入才读取并处理出站头和控制参数（HTTPExportResponse 和 HTTPImportRequest）。

注: 数据处理程序或数据绑定对导入响应或导出请求中的头或控制参数所作的更改将不会改变导入或导出绑定内部的消息处理指令，并且应该仅用于将已修改的值传播到下游 SCA 组件。

上下文服务负责沿 SCA 调用路径传播上下文（包括 HTTP 头之类的协议头以及帐户标识之类的用户上下文）。在 IBM Integration Designer 中进行开发期间，您可以通过导入和导出属性来控制上下文的传播。有关更多详细信息，请参阅 IBM Integration Designer 信息中心中的导入和导出绑定信息。

提供的 HTTP 头结构和支持

表 22 详细列出了 HTTP 导入请求-响应和 HTTP 导出请求-响应的请求-响应参数。

表 22. 提供的 HTTP 头信息

控制名称	HTTP 导入请求	HTTP 导入响应	HTTP 导出请求	HTTP 导出响应
URL	忽略	未设置	从请求消息中读取。 **注:** 查询字符串也是 URL 控制参数的组成部分。	忽略
版本（可能的值: 1.0 和 1.1; 缺省值为 1.1）	忽略	未设置	从请求消息中读取	忽略
方法	忽略	未设置	从请求消息中读取	忽略
动态覆盖 URL	如果在数据处理程序或数据绑定中设置了此参数，那么它将覆盖 HTTP 导入 URL。此参数将写入消息中的请求行中。 **注:** 查询字符串也是 URL 控制参数的组成部分。	未设置	未设置	忽略

表 22. 提供的 HTTP 头信息 (续)

控制名称	HTTP 导入请求	HTTP 导入响应	HTTP 导出请求	HTTP 导出响应
动态覆盖版本	如果设置了此参数，那么它将覆盖 HTTP 导入版本。此参数将写入消息中的请求行中。	未设置	未设置	忽略
动态覆盖方法	如果设置了此参数，那么它将覆盖 HTTP 导入方法。此参数将写入消息中的请求行中。	未设置	未设置	忽略
介质类型（此控制参数携带 HTTP 头 Content-Type 的部分值。）	如果此参数存在，那么它将作为 Content-Type 头的组成部分写入消息中。 **注**: 此控制元素值应该由数据处理程序或数据绑定提供。	从响应消息的 Content-Type 头中读取	从请求消息的 Content-Type 头中读取。	如果此参数存在，那么它将作为 Content-Type 头的组成部分写入消息中。 **注**: 此控制元素值应该由数据处理程序或数据绑定提供。
字符集（缺省值: UTF-8）	如果此参数存在，那么它将作为 Content-Type 头的组成部分写入消息中。 **注**: 此控件元素值应该由数据绑定提供。	从响应消息的 Content-Type 头中读取	从请求消息的 Content-Type 头中读取。	如果此参数受支持，那么它将作为 Content-Type 头的组成部分写入消息中。 **注**: 此控件元素值应该由数据绑定提供。
传输编码（可能的值: chunked 和 identity; 缺省值为 identity）	如果此参数存在，那么它将作为头写入消息中并且控制编码消息变换的方式。	从响应消息中读取	从请求消息中读取	如果此参数存在，那么它将作为头写入消息中并且控制编码消息变换的方式。
内容编码（可能的值: gzip、x-gzip、deflate 和 identity; 缺省值为 identity）	如果此参数存在，那么它将作为头写入消息中并且控制编码有效内容的方式。	从响应消息中读取	从请求消息中读取	如果此参数存在，那么它将作为头写入消息中并且控制编码有效内容的方式。
内容长度	忽略	从响应消息中读取	从请求消息中读取	忽略
状态码（缺省值: 200）	不受支持	从响应消息中读取	不受支持	如果此参数存在，那么它将写入消息中的响应行中
原因短语（缺省值: OK）	不受支持	从响应消息中读取	不受支持	忽略控制值。消息的响应行值将根据状态码生成。
认证（包含多个属性）	如果此参数存在，那么它将用于构造 Basic Authentication 头。 **注**: 仅对 HTTP 协议编码此头的值。在 SCA 中，此头的值将解码为明文并以明文形式传递。	不适用	从请求消息的 Basic Authentication 头中读取。存在此头并非表示用户已认证。认证应该由 Servlet 配置控制。 **注**: 仅对 HTTP 协议编码此头的值。在 SCA 中，此头的值将解码为明文并以明文形式传递。	不适用

表 22. 提供的 HTTP 头信息 (续)

控制名称	HTTP 导入请求	HTTP 导入响应	HTTP 导出请求	HTTP 导出响应
代理（包含多个属性: Host、Port 和 Authentication）	如果此参数存在，那么它将用于通过代理建立连接。	不适用	不适用	不适用
SSL（包含多个属性: Keystore、Keystore Password、Trustore、Trustore Password 和 ClientAuth）	如果填充了此参数并且目标 URL 为 HTTPS，那么此参数将用于通过 SSL 建立连接。	不适用	不适用	不适用

HTTP 数据绑定:

对于服务组件体系结构 (SCA) 消息与 HTTP 协议消息之间的每种不同数据映射，必须配置数据处理程序或 HTTP 数据绑定。数据处理程序提供了独立于绑定的接口（此接口允许跨传输绑定进行复用）并表示建议的方法；数据绑定特定于特定传输绑定。提供了特定于 HTTP 的数据绑定类；另外，您还可以编写定制数据处理程序或数据绑定。

注: 本主题中描述的三个 HTTP 数据绑定类（HTTPStreamDataBindingSOAP、HTTPStreamDataBindingXML 和 HTTPServiceGatewayDataBinding）在 IBM Business Process Manager V7.0 中已不推荐使用。请考虑使用下列数据处理程序来代替本主题中描述的数据绑定:

- 请使用 SOAPDataHandler 代替 HTTPStreamDataBindingSOAP。
- 请使用 UTF8XMLDataHandler 代替 HTTPStreamDataBindingXML。
- 请使用 GatewayTextDataHandler 代替 HTTPServiceGatewayDataBinding。

提供了与 HTTP 导入和 HTTP 导出配合使用的数据绑定: 二进制数据绑定、XML 数据绑定和 SOAP 数据绑定。单向操作不需要响应数据绑定。数据绑定由一个 Java 类的名称表示，这个类的实例可以在 HTTP 与 ServiceDataObject 之间进行双向转换。必须对导出使用函数选择器，此选择器与方法绑定一起确定所使用的数据绑定以及所调用的操作。提供的数据绑定是:

- 二进制数据绑定，此绑定将主体视为非结构化二进制数据。二进制数据绑定 XSD 模式的实现如下所示:

```
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:tns="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="HTTPBaseBody">
    <xsd:sequence/>
  </xsd:complexType>

  <xsd:complexType name="HTTPBytesBody">
    <xsd:complexContent>
      <xsd:extension base="tns:HTTPBaseBody">
        <xsd:sequence>
          <xsd:element name="value" type="xsd:hexBinary"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

- XML 数据绑定，此绑定支持主体作为 XML 数据。XML 数据绑定的实现与 JMS XML 数据绑定类似，并且对接口模式没有限制。
- SOAP 数据绑定，此绑定支持主体作为 SOAP 数据。SOAP 数据绑定的实现对接口模式没有限制。

实现定制 HTTP 数据绑定

本节描述如何实现定制 HTTP 数据绑定。

注: 建议的方法是实现定制数据处理程序，这是因为，此处理程序可以跨传输绑定进行复用。

HTTPStreamDataBinding 是用于处理定制 HTTP 消息的主要接口。此接口设计成允许处理大型有效内容。但是，为了使此类实现能够正常工作，此数据绑定必须先返回控制信息和头，然后才能将消息写入到流中。

定制数据绑定必须实现下列方法及其执行顺序。

要定制数据绑定，请编写实现了 HTTPStreamDataBinding 的类。此数据绑定应该有 4 个私有属性:

- private DataObject pDataObject
- private HTTPControl pCtrl
- private HTTPHeaders pHeaders
- private yourNativeDataType nativeData

HTTP 绑定将按以下顺序调用定制数据绑定:

- 出站处理（DataObject 到本机格式）:
  1. setDataObject(...)
  2. setHeaders(...)
  3. setControlParameters(...)
  4. setBusinessException(...)
  5. convertToNativeData()
  6. getControlParameters()
  7. getHeaders()
  8. write(...)
- 入站处理（本机格式到 DataObject）:
  1. setControlParameters(...)
  2. setHeaders(...)
  3. convertFromNativeData(...)
  4. isBusinessException()
  5. getDataObject()
  6. getControlParameters()
  7. getHeaders()

必须在 convertFromNativeData(...) 中调用 setDataObject(...) 以设置 dataObject 的值，此 dataObject 将从本机数据转换为私有属性“pDataObject”。

```
public void setDataObject(DataObject dataObject)
   throws DataBindingException {
  pDataObject = dataObject;

}
public void setControlParameters(HTTPControl arg0) {
  this.pCtrl = arg0;
}

public void setHeaders(HTTPHeaders arg0) {
```

```
  this.pHeaders = arg0;
}
/*
* Add http header "IsBusinessException" in pHeaders.
* Two steps:
* 1.Remove all the header with name IsBusinessException (case-insensitive) first.
*   This is to make sure only one header is present.
* 2.Add the new header "IsBusinessException"
*/
public void setBusinessException(boolean isBusinessException) {
  //remove all the header with name IsBusinessException (case-insensitive) first.
  //This is to make sure only one header is present.
  //add the new header "IsBusinessException", code example:
  HTTPHeader header=HeadersFactory.eINSTANCE.createHTTPHeader();
  header.setName("IsBusinessException");
  header.setValue(Boolean.toString(isBusinessException));
  this.pHeaders.getHeader().add(header);

}
public HTTPControl getControlParameters() {
  return pCtrl;
}
public HTTPHeaders getHeaders() {
  return pHeaders;
}
public DataObject getDataObject() throws DataBindingException {
  return pDataObject;
}
/*
* Get header "IsBusinessException" from pHeaders, return its boolean value
*/
public boolean isBusinessException() {
  String headerValue = getHeaderValue(pHeaders,"IsBusinessException");
  boolean result=Boolean.parseBoolean(headerValue);
  return result;
}
public void convertToNativeData() throws DataBindingException {
  DataObject dataObject = getDataObject();
  this.nativeData=realConvertWorkFromSDOToNativeData(dataObject);
}
public void convertFromNativeData(HTTPInputStream arg0){
  //Customer-developed method to
  //Read data from HTTPInputStream
  //Convert it to DataObject
  DataObject dataobject=realConvertWorkFromNativeDataToSDO(arg0);
  setDataObject(dataobject);
}
public void write(HTTPOutputStream output) throws IOException {
  if (nativeData != null)
  output.write(nativeData);
}
```

EJB 绑定

Enterprise JavaBeans (EJB) 导入绑定使服务组件体系结构 (SCA) 组件能够调用 Java EE 服务器上运行的 Java EE 业务逻辑所提供的服务。EJB 导出绑允许将 SCA 组件作为 Enterprise JavaBeans 公开，以使 Java EE 业务逻辑能够调用否则不可用的 SCA 组件。

EJB 导入绑定:

EJB 导入绑定通过指定使用者模块与外部 EJB 绑定的方式允许 SCA 模块调用 EJB 实现。从外部 EJB 实现导入服务将允许用户将其业务逻辑放入 IBM Business Process Manager 环境中并参与业务流程。

您可以使用 Integration Designer 来创建 EJB 导入绑定。可以使用以下任一过程来生成这些绑定:

- 使用外部服务向导创建 EJB 导入

  您可以使用 Integration Designer 中的外部服务向导来构建基于现有实现的 EJB 导入。外部服务向导将根据您所提供的条件创建服务。然后，此向导将根据所发现的服务生成业务对象、接口和导入文件。
- 使用组合件编辑器创建 EJB 导入

  您可以使用 Integration Designer 组合件编辑器中的组合件图来创建 EJB 导入。在选用板中，可以使用导入或 Java 类来创建 EJB 绑定。

所生成的导入将具有可建立 Java-WSDL 连接的数据绑定（而不需要 Java 网桥组件）。您可以将具有 Web 服务描述语言 (WSDL) 引用的组件直接连接到通过使用 Java 接口与基于 EJB 的服务进行通信的 EJB 导入。

EJB 导入可以通过使用 EJB 2.1 编程模型或 EJB 3.0 编程模型与 Java EE 业务逻辑交互。

对 Java EE 业务逻辑的调用可以是本地调用（仅限于 EJB 3.0）或远程调用。

- 如果要调用导入所在的服务器上的 Java EE 业务逻辑，那么将使用本地调用。

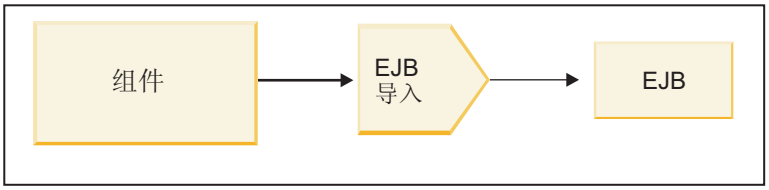


图 27. EJB 的本地调用（仅限于 EJB 3.0）

- 如果要调用的 Java EE 业务逻辑不位于导入所在的服务器上，那么将使用远程调用。

  例如，在下图中，EJB 导入使用基于因特网 ORB 间协议的远程方法调用 (RMI/IIOP) 来调用另一个服务器上的 EJB 方法。

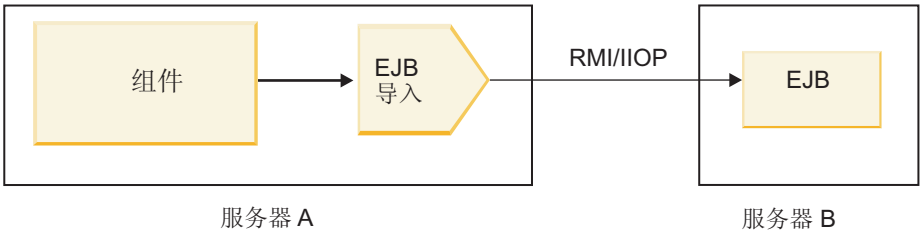


图 28. EJB 的远程调用

配置 EJB 绑定时，Integration Designer 将使用 JNDI 名称来确定 EJB 编程模型级别和调用类型（本地或远程）。

EJB 导入绑定包含下列组要组成部分:

- JAX-WS 数据处理程序
- EJB 故障选择器
- EJB 导入函数选择器

如果您的用户方案未基于 JAX-WS 映射，那么您可能需要定制数据处理程序、函数选择器和故障选择器来执行否则将由 EJB 导入绑定中包含的组件完成的任务。这包括通常由定制映射算法完成的映射。

EJB 导出绑定:

外部 Java EE 应用程序可以通过 EJB 导出绑定调用 SCA 组件。通过使用 EJB 导出，您可以公开 SCA 组件，以便外部 Java EE 应用程序可以使用 EJB 编程模型来调用这些组件。

注: EJB 导出是无状态 Bean。

您可以使用 Integration Designer 来创建 EJB 绑定。可以使用以下任一过程来生成这些绑定:

- 使用外部服务向导创建 EJB 导出绑定

  您可以使用 Integration Designer 中的外部服务向导来构建基于现有实现的 EJB 外部服务。外部服务向导将根据您所提供的条件创建服务。然后，此向导将根据所发现的服务生成业务对象、接口和导出文件。
- 使用组合件编辑器创建 EJB 导出绑定

  您可以使用 Integration Designer 组合件编辑器来创建 EJB 导出。

要点: Java 2 Platform, Standard Edition (J2SE) 客户机无法调用在 Integration Designer 中生成的 EJB 导出客户机。

您可以根据现有 SCA 组件生成绑定，也可以为 Java 接口生成具有 EJB 绑定的导出。

- 为具有现有 WSDL 接口的现有 SCA 组件生成导出时，将为该导出分配一个 Java 接口。
- 为 Java 接口生成导出时，您可以为该导出选择 WSDL 或 Java 接口。

  注: 用于创建 EJB 导出的 Java 接口具有下列与远程调用时作为参数传递的对象（输入和输出参数及异常）相关的限制:
  - 这些对象必须是具体类型（而不是接口或抽象类型）。
  - 这些对象必须符合 Enterprise JavaBean 规范。它们必须可序列化且具有缺省的无参数构造函数，并且所有属性都可通过 getter 方法和 setter 方法进行访问。

    有关 Enterprise JavaBean 规范的信息，请参阅 Sun Microsystems, Inc. Web 站点 (http://java.sun.com)。

  另外，异常还必须是继承自 java.lang.Exception 的已校验的异常，并且必须是单个（即，不支持抛出多个已校验的异常类型）。

  另请注意，Java EnterpriseBean 的业务接口是平面 Java 接口，不能扩展 javax.ejb.EJBObject 或 javax.ejb.EJBLocalObject。该业务接口的方法不应抛出 java.rmi.Remote.Exception。

EJB 导出绑定可以通过使用 EJB 2.1 编程模型或 EJB 3.0 编程模型与 Java EE 业务逻辑交互。

该调用可以是本地调用（仅限于 EJB 3.0）或远程调用。

- 当 Java EE 业务逻辑调用导出所在的服务器上的 SCA 组件时，将使用本地调用。
- 当 the Java EE 业务逻辑不位于导出所在的服务器上时，将使用远程调用。

  例如，在下图中，EJB 使用 RMI/IIOP 调用另一个服务器上的 SCA 组件。

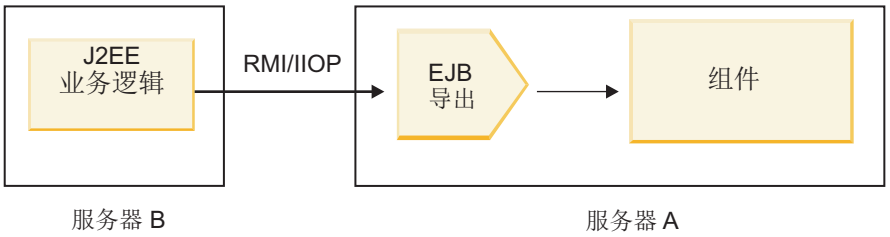


图 29. 客户机通过 EJB 导出远程调用 SCA 组件

配置 EJB 绑定时，Integration Designer 将使用 JNDI 名称来确定 EJB 编程模型级别和调用类型（本地或远程）。

EJB 导出绑定包含下列主要组成部分:

- JAX-WS 数据处理程序
- EJB 导出函数选择器

如果您的用户方案未基于 JAX-WS 映射，那么您可能需要定制数据处理程序和函数选择器来执行否则将需要由 EJB 导出绑定中包含的组件完成的任务。这包括通常由定制映射算法完成的映射。

EJB 绑定属性:

EJB 导入绑定使用其已配置的 JNDI 名称来确定 EJB 编程模型级别和调用类型（本地或远程）。EJB 导入和导出绑定使用 JAX-WS 数据处理程序进行数据变换。EJB 导入绑定使用 EJB 导入函数选择器和 EJB 故障选择器，而 EJB 导出绑定使用 EJB 导出函数选择器。

JNDI 名称和 EJB 导入绑定:

对导入配置 EJB 绑定时，Integration Designer 将使用 JNDI 名称来确定 EJB 编程模型级别和调用类型（本地或远程）。

如果未指定 JNDI 名称，那么将使用缺省 EJB 接口绑定。所创建的缺省名称取决于您调用的是 EJB 2.1 JavaBean 还是 EJB 3.0 JavaBean。

注: 有关命名约定的更详细信息，请参阅 WebSphere Application Server 信息中心: EJB 3.0 应用程序绑定概述。

- EJB 2.1 JavaBean

  Integration Designer 预先选择的缺省 JNDI 名称是缺省 EJB 2.1 绑定，该绑定采用的格式为: **ejb/** 后跟 home 接口（用斜杠分隔）。

  例如，如果 EJB 2.1 JavaBean 的 home 接口为 com.mycompany.myremotebusinesshome，那么缺省绑定如下所示:

  ```
  ejb/com/mycompany/myremotebusinesshome
  ```

  对于 EJB 2.1，仅支持远程 EJB 调用。

- EJB 3.0 JavaBean

  Integration Designer 为本地 JNDI 预先选择的缺省 JNDI 名称是以 **ejblocal:** 作为前缀的本地接口的标准类名。例如，对于本地接口 com.mycompany.mylocalbusiness 的标准接口，预先选择的 EJB 3.0 JNDI 如下所示:

```
ejblocal:com.mycompany.mylocalbusiness
```

对于远程接口 com.mycompany.myremotebusiness，预先选择的 EJB 3.0 JNDI 是标准接口:

```
com.mycompany.myremotebusiness
```

以下位置对 EJB 3.0 缺省应用程序绑定作了阐述: EJB 3.0 应用程序绑定概述。

对于使用 V3.0 编程模型的 EJB，Integration Designer 将使用“短”名称作为缺省 JNDI 位置。

注: 如果由于使用或配置了定制映射而导致目标 EJB 的已部署 JNDI 引用不同于缺省 JNDI 绑定位置，那么必须正确地指定目标 JNDI 名称。您可以在部署之前在 Integration Designer 中指定名称，或者对于导入绑定，您可以（在部署之后）在管理控制台中更改名称以便与目标 EJB 的 JNDI 名称相匹配。

有关创建 EJB 绑定的更多信息，请参阅 Integration Designer 信息中心中专门介绍使用 EJB 绑定的部分。

JAX-WS 数据处理程序:

Enterprise JavaBeans (EJB) 导入绑定使用 JAX-WS 数据处理程序将请求业务对象转换成 Java 对象参数并将 Java 对象返回值转换成响应业务对象。EJB 导出绑定使用 JAX-WS 数据处理程序将请求 EJB 转换成请求业务对象并将响应业务对象转换成返回值。

此数据处理程序通过使用“针对 XML Web Service 的 Java API”(JAX-WS) 规范和 Java XML 绑定体系结构 (JAXB) 规范将来自 SCA 指定的 WSDL 接口的数据映射到目标 EJB Java 接口（反之亦然）。

注: 当前限制为仅支持 JAX-WS 2.1.1 和 JAXB 2.1.3 规范。

在 EJB 绑定级别指定的数据处理程序用于执行请求、响应、故障和运行时异常处理。

注: 对于故障，可以通过指定 faultBindingType 配置属性对每个故障指定特定数据处理程序。这将覆盖在 EJB 绑定级别指定的值。

当 EJB 绑定具有 WSDL 接口时，缺省情况下将使用 JAX-WS 数据处理程序。此数据处理程序无法用于将表示 JAX-WS 调用的 SOAP 消息变换成数据对象。

EJB 导入绑定使用数据处理程序将数据对象变换成 Java 对象数组 (Object[])。出站通信期间，将执行下列处理:

1. EJB 绑定在 BindingContext 中设置需要的类型、需要的元素和目标方法名以便与 WSDL 中定义的那些值相匹配。
2. EJB 绑定对需要数据变换的数据对象调用变换方法。
3. 数据处理程序返回表示该方法参数的 Object[]（按照该方法中这些参数的定义的顺序）。
4. EJB 绑定使用 Object[] 对目标 EJB 接口调用该方法。

该绑定还准备 Object[] 以处理来自 EJB 调用的响应。

- Object[] 中的第一个元素是 Java 方法调用的返回值。
- 后续值表示该方法的输入参数。

这是支持类型为 In/Out 和 Out 的参数所必需的。

对于类型为 Out 的参数，必须在响应数据对象中返回值。

数据处理程序对在 Object[] 中找到的值进行处理和变换，然后向数据对象返回响应。

数据处理程序支持 xs:AnyType、xs:AnySimpleType 和 xs:Any 以及其他 XSD 数据类型。为了启用对 xs:Any 的支持，请使用 Java 代码对 JavaBean 属性使用 **@XmlAnyElement (lax=true)**，如以下示例所示:

```
public class TestType {
 private Object[] object;

 @XmlAnyElement (lax=true)
 public Object[] getObject() {
  return object;
 )

 public void setObject (Object[] object) {
  this.object=object;
 )
)
```

这将使 TestType 中的该属性对象成为一个 xs:any 字段。在 xs:any 字段中使用的 Java 类值应该具有 **@XmlAnyElement** 注释。例如，如果 Address 是用于填充对象数组的 Java 类，那么 Address 类应该具有 **@XmlRootElement** 注释。

注: 要定制从 XSD 类型到 JAX-WS 规范所定义的 Java 类型的映射，请根据您的业务需求更改 JAXB 注释。JAX-WS 数据处理程序支持 xs:any、xs:anyType 和 xs:anySimpleType。

下列限制适用于 JAX-WS 数据处理程序:

- 该数据处理程序不包括对头属性 **@WebParam** 注释的支持。
- 业务对象模式文件（XSD 文件）的名称空间不包括来自 Java 包名的缺省映射。package-info.java 中的 **@XMLSchema** 注释也无效。创建具有名称空间的 XSD 的唯一方法是使用 **@XmlType** 和 **@XmlRootElement** 注释。**@XmlRootElement** 定义 JavaBean 类型中的全局元素的目标名称空间。
- EJB 导入不会为无关类创建 XSD 文件。V2.0 不支持 **@XmlSeeAlso** 注释，因此，如果子类不是直接从父类引用的，那么将不会创建 XSD。此问题的解决方案是对这种子类运行 SchemaGen。

  SchemaGen 是一个命令行实用程序（位于 *WPS_Install_Home*/bin 目录中），用于为给定 JavaBean 创建 XSD 文件。必须手动将这些 XSD 复制到模块，该解决方案才会有效。

EJB 故障选择器:

EJB 故障选择器用于确定 EJB 调用是导致故障、运行时异常还是成功响应。

如果检测到故障，那么 EJB 故障选择器将本机故障名称返回给绑定运行时，以便 JAX-WS 数据处理程序可以将异常对象转换成故障业务对象。

接收到成功（非故障）响应时，EJB 导入绑定将组装 Java 对象数组 (Object[]) 以返回值。

- Object[] 中的第一个元素是 Java 方法调用的返回值。
- 后续值表示该方法的输入参数。

这是支持类型为 In/Out 和 Out 的参数所必需的。

对于异常场景，绑定将组装 Object[] 并且第一个元素表示该方法所抛出的异常。

故障选择器可以返回下列任何值:

表 23. 返回值

类型	返回值	描述
故障	**ResponseType.FAULT**	当所传递的 Object[] 包含异常对象时返回此值。

表 23. 返回值 (续)

类型	返回值	描述
运行时异常	**ResponseType.RUNTIME**	当异常对象与方法中声明的任何异常类型都不匹配时返回此值。
正常响应	**ResponseType.RESPONSE**	在所有其他情况下返回此值。

如果故障选择器返回值 **ResponseType.FAULT**，那么将返回本机故障名称。此本机故障名称由绑定用来确定模型中的相应 WSDL 故障名称并调用正确的故障数据处理程序。

EJB 函数选择器:

EJB 绑定使用导入函数选择器（对于出站处理）或导出函数选择器（对于入站处理）来确定要调用的 EJB 方法。

导入函数选择器

对于出站处理，导入函数选择器将根据连接到 EJB 导入的 SCA 组件所调用的操作的名称派生 EJB 方法类型。该函数选择器将在 Integration Designer 所生成且由 JAX-WS 注释的 Java 类中查找 @WebMethod 注释以确定相关联的目标操作名称。

- 如果 @WebMethod 注释存在，那么该函数选择器将使用 @WebMethod 注释确定 WSDL 方法的正确 Java 方法映射。
- 如果缺少 @WebMethod 注释，那么函数选择器将认为 Java 方法名与所调用的操作名相同。

注: 该函数选择器仅对 EJB 导入上的 WSDL 类型化接口有效，对 EJB 导入上的 Java 类型化接口无效。

该函数选择器将返回表示 EJB 接口的方法的 java.lang.reflect.Method 对象。

该函数选择器将使用 Java 对象数组 (Object[]) 包含来自目标方法的响应。Object[] 中的第一个元素是具有 WSDL 的名称的 Java 方法，而 Object[] 中的第二个元素是输入业务对象。

导出函数选择器

对于入站处理，导出函数选择器将派生要从 Java 方法调用的目标方法。

导出函数选择器将 EJB 客户机所调用的 Java 操作名映射成目标组件的接口中的操作名。方法名将以字符串形式返回，并且将由 SCA 运行时根据目标组件的接口类型进行解析。

EIS 绑定

企业信息系统 (EIS) 绑定在 SCA 组件与外部 EIS 之间提供连通性。这种通信是通过使用支持 JCA 1.5 资源适配器和 Websphere Adapters 的 EIS 导出和 EIS 导入实现的。

SCA 组件可能要求数据传输到外部 EIS 或者接收来自外部 EIS 的数据。创建需要这种连通性的 SCA 模块时，您不仅要包括 SCA 组件，还要包括一个具有 EIS 绑定的导入或导出以便与特定外部 EIS 通信。

WebSphere Integration Developer 中的资源适配器将在导入或导出的上下文内使用。您可以使用外部服务向导来开发导入或导出，在开发过程中，将包括资源适配器。将创建一个使用特定资源适配器的 EIS 导入（该导入允许应用程序调用 EIS 系统上的服务）或 EIS 导出（该导出允许 EIS 系统上的应用程序调用在 WebSphere Integration Developer 中开发的服务）。例如，将创建一个使用 JD Edwards 适配器的导入以调用 JD Edwards 系统上的服务。

使用外部服务向导时，将为您创建 EIS 绑定信息。另外，您也可以使用另一个工具（即，组合件编辑器）来添加或修改绑定信息。有关更多信息，请参阅 Integration Designer 信息中心。

将包含 EIS 绑定的 SCA 模块部署到服务器后，您可以使用管理控制台来查看关于该绑定的信息或者配置该绑定。

EIS 绑定概述:

EIS（企业信息系统）绑定在与 JCA 资源适配器配合使用时，允许您访问企业信息系统中的服务或者使您的服务可供 EIS 使用。

以下示例说明名为 ContactSyncModule 的 SCA 模块如何同步 Siebel 系统与 SAP 系统之间的联系人信息。

1. 名为 ContactSync 的 SCA 组件侦听（通过名为 Siebel Contact 的 EIS 应用程序导出）对 Siebel 联系人所作的更改。
2. SCA 组件 ContactSync 本身使用 SAP 应用程序（通过 EIS 应用程序导入）以相应地更新 SAP 联系人信息。

由于 Siebel 和 SAP 系统中用于存储联系人的数据结构不同，因此，SCA 组件 ContactSync 必须提供映射。

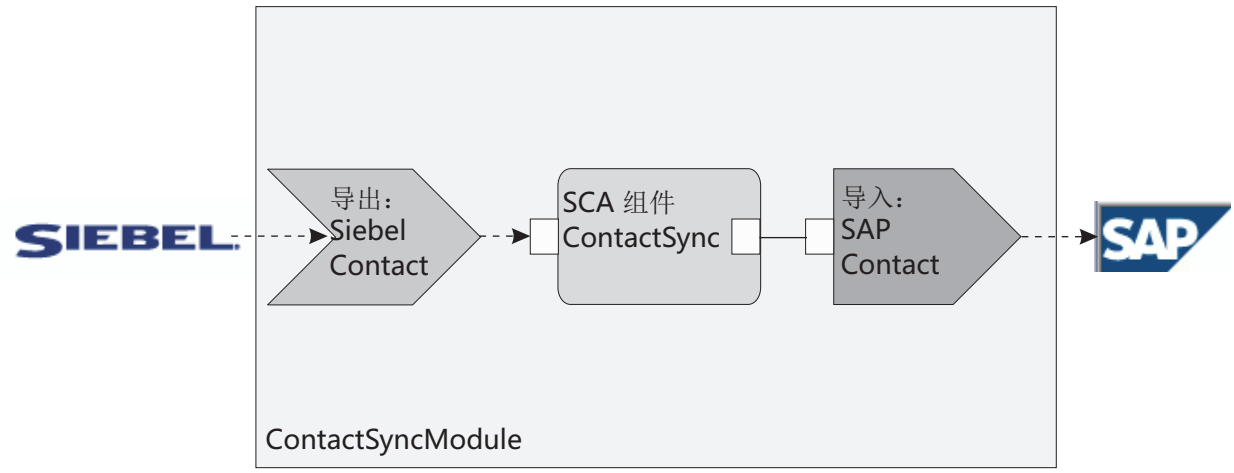


图 30. 从 Siebel 系统到 SAP 系统的流

Siebel Contact 导出和 SAP Contact 导入配置了适当的资源适配器。

EIS 绑定的主要特征:

EIS 导入是一种服务组件体系结构 (SCA) 导入，它允许 SCA 模块中的组件使用在此 SCA 模块外部定义的 EIS 应用程序。EIS 导入用来将数据从 SCA 组件传输到外部 EIS；EIS 导出用来将数据从外部 EIS 传输到 SCA 模块中。

导入

EIS 导入的作用是将 SCA 组件与外部 EIS 系统联系起来。可以将外部应用程序视作 EIS 导入。在这种情况下，EIS 导入将数据发送至外部 EIS，并且可以选择接收响应中的数据。

EIS 导入为 SCA 组件提供了模块外部的应用程序的统一视图。这允许组件使用一致的 SCA 模型与外部 EIS（例如，SAP、Siebel 或 PeopleSoft）进行通信。

在导入的客户端，有一个由 EIS 导入应用程序公开的接口，此接口具有一个或多个方法，每个方法采用数据对象作为自变量和返回值。在实现端，具有由资源适配器实现的公共客户机接口 (CCI)。

EIS 导入的运行时实现用于连接客户端接口和 CCI。导入将对于接口的方法调用映射至对于 CCI 的调用。

绑定是在以下三个级别创建的：首先是接口绑定，接口绑定使用所包含的方法绑定，方法绑定又使用数据绑定。

接口绑定使导入的接口与提供应用程序的 EIS 系统的连接相关。这反映一个事实，即，此接口表示的一组应用程序是由 EIS 的特定实例提供的，并且通过此连接来访问此实例。绑定元素包含一些属性，这些属性具有足够的信息来创建连接（这些属性是 javax.resource.spi.ManagedConnectionFactory 实例的一部分）。

方法绑定使方法与针对 EIS 系统的特定交互相关联。对于 JCA，接口的特征通过 javax.resource.cci.InteractionSpec 接口实现的一组属性来体现。方法绑定的交互元素包含这些属性以及类名，从而提供了足够的信息来执行交互。方法绑定使用数据绑定，而数据绑定描述接口方法的自变量和结果与 EIS 表示之间的映射。

EIS 导入的运行时场景为如下所示：

1. 使用 SCA 编程模型来调用导入接口中的方法。
2. 到达 EIS 导入的请求包含方法的名称及其自变量。
3. 导入首先创建一个接口绑定实现；然后，它使用导入绑定中的数据来创建一个连接工厂，然后使此实现与连接工厂相关联。即，导入将对接口绑定调用 setConnectionFactory。
4. 创建与所调用的方法相匹配的方法绑定实现。
5. 创建并填充了 javax.resource.cci.InteractionSpec 实例；然后，使用数据绑定将方法自变量绑定至资源适配器识别的一种格式。
6. 使用 CCI 接口来执行交互。
7. 当此调用返回时，数据绑定用来创建调用结果，并将结果返回给调用者。

导出

EIS 导出的作用是将 SCA 组件与外部 EIS 联系起来。可以将外部应用程序视作 EIS 导出。在这种情况下，外部应用程序采用定期通知的形式发送其数据。可以将 EIS 导出视作用于侦听来自 EIS 的外部请求的预订应用程序。使用 EIS 导出的 SCA 组件将它视作本地应用程序。

EIS 导出为 SCA 组件提供了模块外部的应用程序的统一视图。这允许组件使用一致的 SCA 模型与 EIS（例如，SAP、Siebel 或 PeopleSoft）进行通信。

导出充当用于接收来自 EIS 的请求的侦听器实现。侦听器将实现特定于资源适配器的侦听器接口。导出还包含一个用于实现接口的组件，此组件通过导出向 EIS 公开。

EIS 导出的运行时实现将侦听器与用于实现接口的组件联系起来。导出将 EIS 请求映射至对组件调用相应的操作。绑定是在以下三个级别创建的：首先是侦听器绑定，然后侦听器绑定使用所包含的本机方法绑定，本机方法绑定又使用数据绑定。

侦听器绑定将用于接收请求的侦听器与通过导出而公开的组件联系起来。导出定义包含组件的名称；运行时将查找此组件名称并将请求转发至该组件。

本机方法绑定使侦听器接收到的本机方法或事件类型与通过导出而公开的组件所实现的操作相关联。对侦听器调用的方法与事件类型之间没有关系；所有事件都通过侦听器的一个或多个方法而到达。本机方法绑定使用导出中所定义的函数选择器从入站数据和数据绑定中抽取本机方法名，以将 EIS 的数据格式绑定至该组件识别的格式。

EIS 导出的运行时场景为如下所示:

1. EIS 请求将触发对侦听器实现调用方法。
2. 侦听器将找到并调用导出，并将调用自变量传递给导出。
3. 导出将创建侦听器绑定实现。
4. 导出将函数选择器实例化并在侦听器绑定中设置此函数选择器。
5. 导出将初始化本机方法绑定并将这些绑定添加至侦听器绑定。对于每个本机方法绑定，还会初始化数据绑定。
6. 导出将调用侦听器绑定。
7. 侦听器绑定将找到已导出的组件，并使用函数选择器来检索本机方法名。
8. 此名称用来查找本机方法绑定，然后本机方法绑定将调用目标组件。

适配器交互方式允许 EIS 导出绑定以异步方式（这是缺省情况）或同步方式调用目标组件。

资源适配器

使用外部服务向导来开发导入或导出，在开发过程中，将包括资源适配器。随 IBM Integration Designer 一起提供的、用来访问 CICS、IMS、JD Edwards、PeopleSoft、SAP 和 Siebel 系统的适配器仅用于开发和测试目的。也就是说，您使用适配器来开发和测试应用程序。

部署应用程序之后，您将需要使用已授权的运行时适配器来运行应用程序。但是，当您构建服务时，可以将适配器嵌入服务中。您的适配器许可证发放规则可能允许您使用嵌入的适配器作为已授权的运行时适配器。这些适配器符合 Java EE 连接器体系结构 (JCA 1.5)。JCA 是一项公开标准，是 Java EE 建立 EIS 连接时遵循的标准。JCA 提供了一个受管框架；即，由应用程序服务器提供服务质量 (QoS)，服务质量对事务提供生命周期管理和安全性。除了 IBM CICS ECI 资源适配器和 IBM IMS Connector for Java 之外，服务质量还符合 Enterprise Metadata Discovery 规范。

此向导还支持 WebSphere Business Integration Adapters（这是一组更早开发的适配器）。

Java EE 资源

可以将 EIS 模块（这是遵循 EIS 模块模式的 SCA 模块）部署到 Java EE 平台。

将 EIS 模块部署到 Java EE 平台时，会导致准备执行的应用程序打包为 EAR 文件并部署到服务器。现在，已经创建了所有 Java EE 工件和资源；已配置应用程序并且准备运行。

JCA 交互规范和连接规范的动态属性:

通过使用定义明确且包含有效内容的子数据对象，EIS 绑定可以为指定的 InteractionSpec 和 ConnectionSpec 接受输入。这允许通过 InteractionSpec 与资源适配器进行动态的请求-响应交互以及通过 ConnectionSpec 进行组件认证。

javax.cci.InteractionSpec 包含有关如何处理要与资源适配器进行交互请求的信息。它还包含有关在发出请求后如何进行交互的信息。有时，将这些通过交互进行的双向通信称为*对话*。

EIS 绑定期望作为资源适配器参数的有效内容包含名为 **properties** 的子数据对象。此属性数据对象将包含名称/值对，其中，名称是特定格式的交互规范属性名称。格式化规则为:

- 名称必须以 **IS** 前缀开头，后跟属性名。例如，名为 **InteractionId** 的 JavaBeans 属性的交互规范应将属性名指定为 **ISInteractionId**。
- 名称/值对表示简单类型的交互规范属性的名称和值。

在此示例中，接口指定操作的输入是 **Account** 数据对象。此接口调用 EIS 导入绑定应用程序，以便发送和接收名为 **workingSet** 且值为 **xyz** 的动态 InteractionSpec 属性。

服务器中的业务图或业务对象包含底层 **properties** 业务对象，该业务对象允许发送特定于协议且包含有效内容的数据。此 **properties** 业务对象是内置的，不必在构造业务对象时在 XML 模式中指定。您只需创建并使用此对象。如果您有自己的基于 XML 模式定义的数据类型，那么需要指定包含期望的名称/值对的 **properties** 元素。

```
BOFactory dataFactory = (BOFactory) \
serviceManager.locateService("com/ibm/websphere/bo/BOFactory");
 //Wrapper for doc-lit wrapped style interfaces,
 //skip to payload for non doc-lit
 DataObject docLitWrapper = dataFactory.createByElement /
 ("http://mytest/eis/Account", "AccountWrapper");
```

创建有效内容。

```
DataObject account = docLitWrapper.createDataObject(0);
  DataObject accountInfo = account.createDataObject("AccountInfo");
  //Perform your setting up of payload


  //Construct properties data for dynamic interaction

  DataObject properties = account.createDataObject("properties");
```

对于名称 workingSet，设置期望的值 (**xyz**)。

```
properties.setString("ISworkingSet", "xyz");


  //Invoke the service with argument

  Service accountImport = (Service) \
  serviceManager.locateService("AccountOutbound");
  DataObject result = accountImport.invoke("createAccount", docLitWrapper);

  //Get returned property
  DataObject retProperties = result.getDataObject("properties");

  String workingset = retProperties.getString("ISworkingSet");
```

可以使用 ConnectionSpec 属性进行动态组件认证。除了属性名称前缀必须为 **CS** 而不是 **IS** 以外，上述规则均适用。ConnectionSpec 属性不是双向的。同一个 **properties** 数据对象可以同时包含 IS 和 CS 属性。

要使用 ConnectionSpec 属性，请将您对导入绑定指定的 **resAuth** 设置为 **Application**。并且，请确保资源适配器支持组件授权。有关更多详细信息，请参阅 J2EE 连接器体系结构规范的第 8 章。

使用 EIS 绑定的外部客户机:

服务器可以向使用 EIS 绑定的外部客户机发送消息或接收来自这些客户机的消息。

外部客户机（例如，Web 门户网站或 EIS）需要向服务器中的 SCA 模块发送消息，或者需要被服务器内的组件调用。

像调用任何其他应用程序一样，客户机使用动态调用接口 (DII) 或 Java 接口调用 EIS 导入。

1. 外部客户机创建 ServiceManager 的实例并使用 EIS 导入的引用名称来查找该 EIS 导入。结果查找到服务接口实现。
2. 客户机创建一个输入参数，此参数是使用数据对象模式动态创建的通用数据对象。此步骤是使用服务数据对象 DataFactory 接口实现完成的。

3. 外部客户机调用 EIS 并获取所需的结果。

或者，客户机也可以使用 Java 接口调用 EIS 导入。

1. 客户机创建 ServiceManager 的实例并使用 EIS 导入的引用名称来查找该 EIS 导入。结果查找到该 EIS 导入的 Java 接口。
2. 客户机创建一个输入参数和一个类型化数据对象。
3. 客户机调用 EIS 并获取所需的结果。

EIS 导出接口定义可供外部 EIS 应用程序使用的已导出 SCA 组件的接口。可以将此接口视为外部应用程序（例如 SAP 或 PeopleSoft）将通过 EIS 导出应用程序运行时的实现进行调用的接口。

该导出使用 EISExportBinding 将已导出的服务与外部 EIS 应用程序绑定。它允许您预订 SCA 模块中包含的应用程序以侦听 EIS 服务请求。EIS 导出绑定指定资源适配器（使用 Java EE 连接器体系结构接口）所理解的入站事件的定义与 SCA 操作调用之间的映射。

EISExportBinding 要求外部 EIS 服务基于 Java EE 连接器体系结构 1.5 入站协定。EISExportBinding 要求在绑定级别或方法级别指定数据处理程序或数据绑定。

JMS 绑定

Java 消息服务 (JMS) 提供程序根据 Java 消息传递服务 API 和编程模型启用消息传递。JMS 提供程序提供 Java EE 连接工厂以便为 JMS 目标创建连接并且发送和接收消息。

提供了三个 JMS 绑定:

- 符合 JMS JCA 1.5 的服务集成总线 (SIB) 提供程序绑定（*JMS 绑定*）
- 符合 JMS 1.1 的非 JCA 的通用 JMS 绑定（*通用 JMS 绑定*）
- 提供对 WebSphere MQ 的 JMS 提供程序支持并允许 Java EE 应用程序互操作性的 WebSphere MQ JMS 绑定（*WebSphere MQ JMS 绑定*）

JMS 导出和导入绑定允许服务组件体系结构 (SCA) 模块调用外部 JMS 系统并接收来自这些系统的消息。

另外，WebSphere MQ 绑定（*WebSphere MQ 绑定*）也受支持，这些绑定允许本机 MQ 用户处理任意入局消息格式和出局消息格式（需要 WebSphere MQ）。

JMS 导入和导出绑定提供与使用基于 JCA 1.5 的 SIB JMS 提供程序（该提供程序包括在 WebSphere Application Server 中）的集成。其他基于 JCA 1.5 的 JMS 资源适配器不受支持。

JMS 绑定概述:

JMS 绑定提供服务组件体系结构 (SCA) 环境与 JMS 系统之间的连接。

JMS 绑定

JMS 导入绑定和 JMS 导出绑定的主要组成部分如下所示:

- 资源适配器: 启用 SCA 模块与外部 JMS 系统之间的受管双向连接
- 连接: 封装客户机与提供程序应用程序之间的虚拟连接
- 目标: 由客户机用来指定它所生成的消息的目标或者它所使用的消息的源
- 认证数据: 用于保护对绑定的访问

JMS 导入绑定

JMS 导入绑定允许您导入要在 SCA 模块内使用的外部 JMS 应用程序。JMS 导入绑定允许 SCA 模块内的组件与外部 JMS 应用程序所提供的服务通信。

与 JMS 目标的相关联 JMS 提供程序的连接是使用 JMS 连接工厂创建的。请使用连接工厂管理对象来管理缺省消息传递提供程序的 JMS 连接工厂。

与外部 JMS 系统的交互包括使用目标来发送请求和接收应答。

根据所调用的操作的类型，支持两种类型的 JMS 导入绑定使用方案:

- 单向: JMS 导入将消息放入导入绑定所配置的发送目标中。未设置 JMS 头的 replyTo 字段。
- 双向（请求-响应）: JMS 导入将消息放入发送目标中，然后坚持等待它将从 SCA 组件接收到的应答。

  可以将导入绑定配置为（通过使用 Integration Designer 中的**响应相关方案**）要求响应消息相关标识是从请求消息标识（缺省值）或者从请求消息相关标识复制的。另外，还可以将导入绑定配置为使用临时动态响应目标使响应与请求相关。将为每个请求创建一个临时目标，并且该导入使用此目标来接收响应。

  在出站消息的 replyTo 头属性中设置接收目标。将部署消息侦听器以侦听接收目标，并且在接收到应答时，消息侦听器将该应答传递回组件。

对于单向和双向使用方案，都可以指定动态和静态头属性。可以从 JMS 导入方法绑定设置静态属性。这些属性中的部分属性对 SCA JMS 运行时具有特殊意义。

请务必注意，JMS 是异步绑定。如果调用组件以同步方式调用 JMS 导入（对于双向操作），那么在 JMS 服务返回响应之前，该调用组件将被阻塞。

第 90 页的图 31 说明导入如何链接到外部服务。

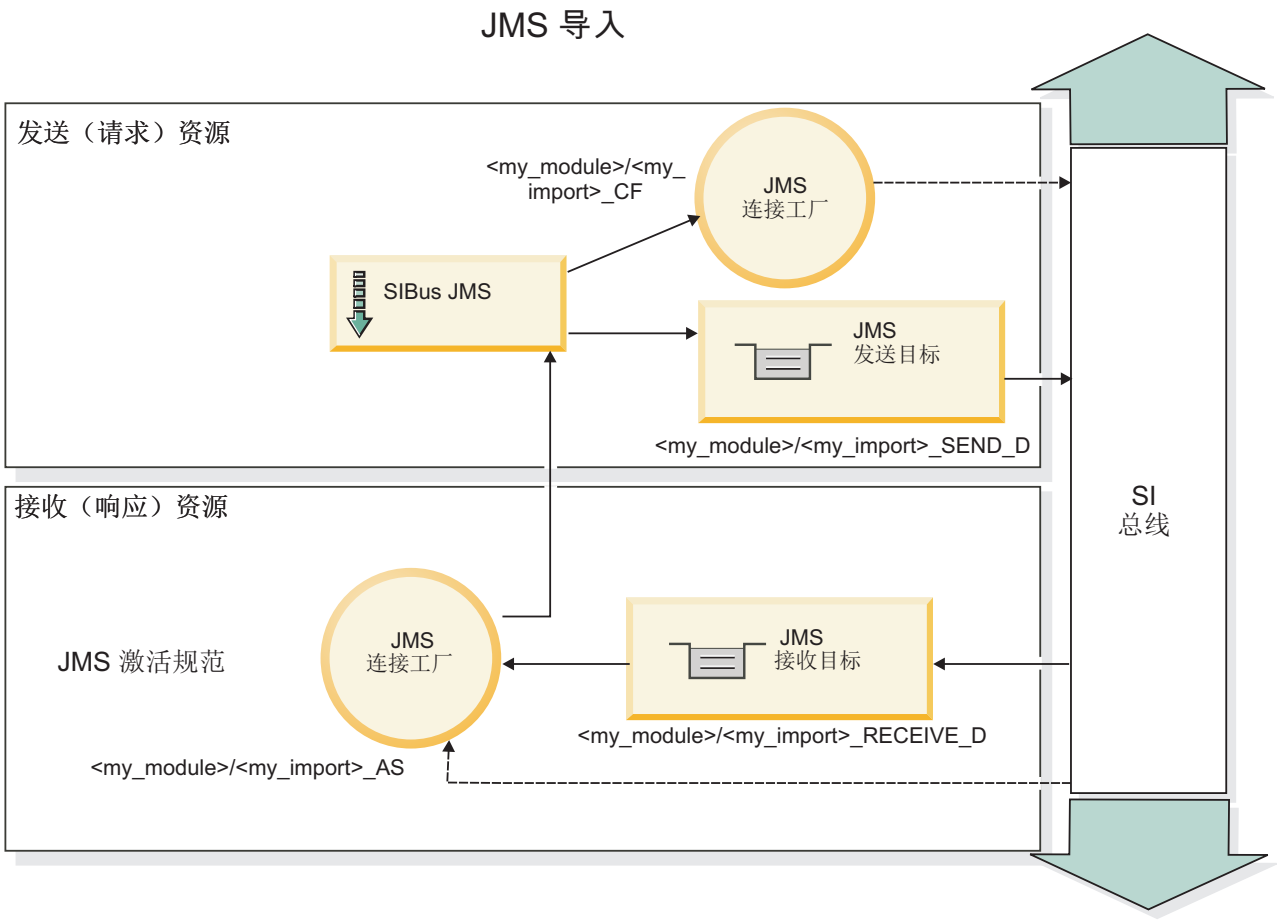


图 31. JMS 导入绑定资源

JMS 导出绑定

JMS 导出绑定为 SCA 模块提供了用于向外部 JMS 应用程序提供服务的方法。

作为 JMS 导出的组成部分的连接是可配置的激活规范。

JMS 导出具有发送和接收目标。

- 接收目标是应该用来放置目标组件的入局消息的位置。
- 发送目标是将发送应答的位置，除非入局消息使用 replyTo 头属性覆盖了此目标。

将部署消息侦听器以侦听传入到导出绑定中指定的接收目标的请求。发送字段中指定的目标用于发送对入站请求的应答（如果所调用的组件提供应答）。在入局消息的 replyTo 字段中指定的目标将覆盖在发送中指定的目标。

第 91 页的图 32 说明外部请求程序如何链接到导出。

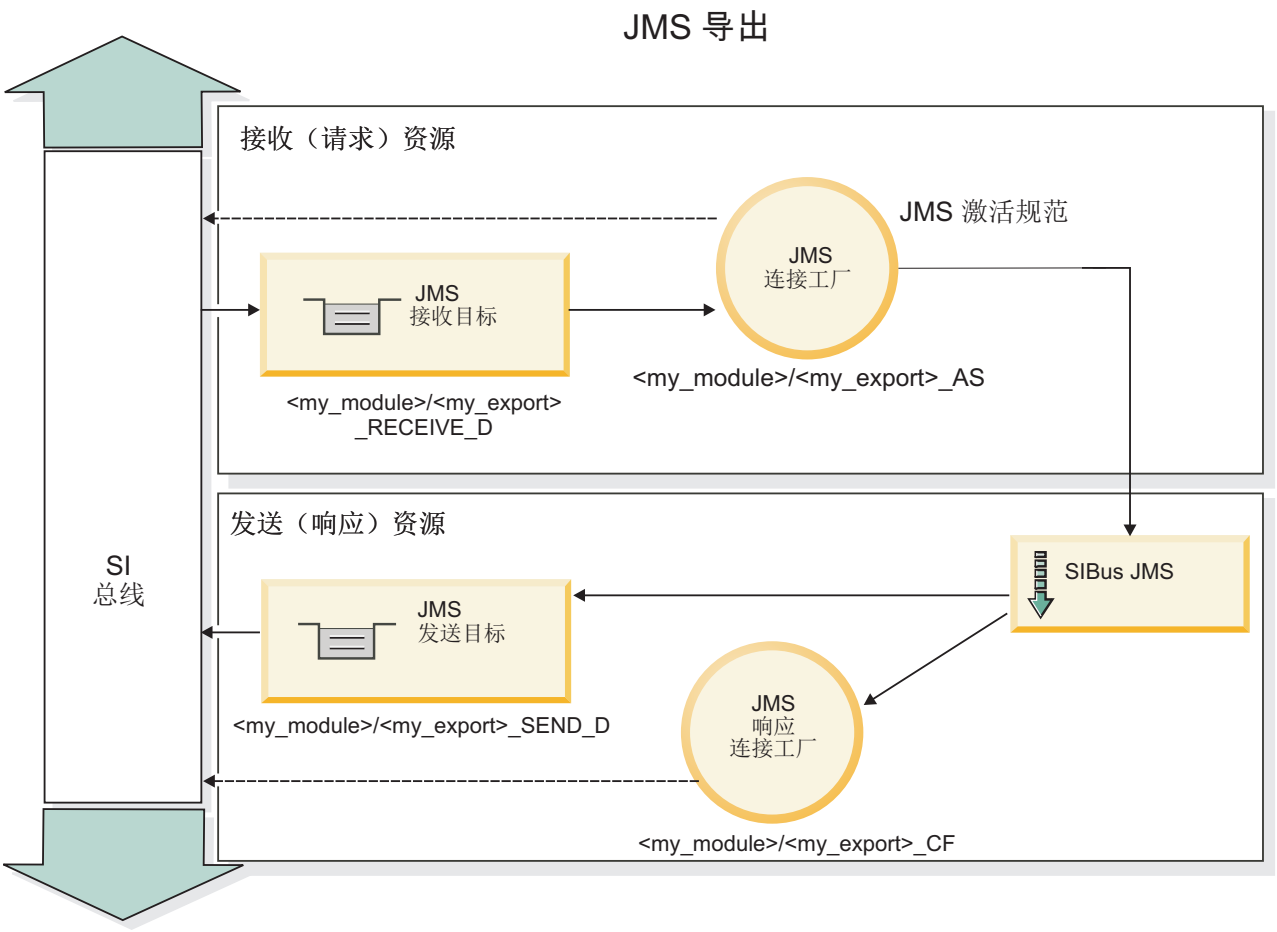


图 32. JMS 导出绑定资源

JMS 集成和资源适配器:

Java 消息服务 (JMS) 通过可用的基于 JMS JCA 1.5 的资源适配器提供集成。为服务集成总线 (SIB) JMS 资源适配器提供了全面的 JMS 集成支持。

如果想要与符合 JCA 1.5 的外部 JMS 系统集成，请将 JMS 提供程序用于 JCA 1.5 资源适配器。符合 JCA 1.5 的外部服务可以通过使用 SIB JMS 资源适配器来接收和发送消息以便与服务组件体系结构 (SCA) 组件集成。

不支持使用其他特定于提供程序的 JCA 1.5 资源适配器。

无法将 JMS 模块部署到 Java SE 环境。这种模块只能部署到 Java EE 环境。

JMS 绑定的关键功能:

JMS 导入和导出绑定的关键功能包括头和已创建的 Java EE 资源。

特殊头

JMS 导入和导出中的特殊头属性用于告诉目标如何处理该消息。

例如，TargetFunctionName 从本机方法映射至操作方法。

Java EE 资源

将 JMS 导入和导出部署至 Java EE 环境时，系统会创建许多 Java EE 资源。

ConnectionFactory
被客户机用于创建 JMS 提供程序的连接。

ActivationSpec
导入使用此项来接收对请求的响应；导出在配置消息端点使用此项，这些端点在其与消息传递系统的交互中表示消息侦听器。

目标

- 发送目标：在导入上，这是发送请求或外发消息的位置；在导出上，这是将发送响应消息的目标（只要未被入局消息中的 JMSReplyTo 头字段替代）。
- 接收目标：应放置入局消息的位置；对于导入，这是响应；对于导出，这是请求。
- 回调目标：用于存储相关信息的 SCA JMS 系统目标。请不要读取或写入至此目标。

安装任务创建 ConnectionFactory 和 3 个目标。它还会创建 ActivationSpec 以允许运行时消息侦听器侦听接收目标上的回复。这些资源的属性是在导入或导出文件中指定的。

JMS 头:

JMS 消息包含两种类型的头：JMS 系统头和多个 JMS 属性。这两种类型的头都可以在服务消息对象 (SMO) 中的调解模块中访问，也可以通过使用 ContextService API 来访问。

JMS 系统头

JMS 系统头在 SMO 中由 JMSHeader 元素表示，该元素包含所有通常在 JMS 头中找到的字段。虽然可以在调解中（或者使用 ContextService）修改这些字段，但在 SMO 中设置的某些 JMS 系统头字段不会在出站 JMS 消息中传播，这是因为这些字段将被系统或静态值覆盖。

可以在调解中（或者使用 ContextService）更新的关键 JMS 系统头字段如下所示:

- **JMSType** 和 **JMSCorrelationID** - 特定预定义的消息头属性的值
- **JMSDeliveryMode** - 传递方式的值（persistent 或 nonpersistent; 缺省值为 persistent）
- **JMSPriority** - 优先级值（0 到 9; 缺省值为 JMS_Default_Priority）

JMS 属性

JMS 属性在 SMO 中表示为“属性”列表中的条目。可以在调解中或者使用 ContextService API 来添加、更新或删除属性。

也可以在 JMS 绑定中以静态方式设置属性。以静态方式设置的属性将覆盖以动态方式设置的同名设置。

从其他绑定（例如 HTTP 绑定）传播的用户属性将在 JMS 绑定作为 JMS 属性输出。

头传播设置

将 JMS 系统头和属性从入站 JMS 消息传播到下游组件或者从上游组件传播到出站 JMS 消息可以由绑定中的“传播协议头”标志控制。

设置“传播协议头”后，将允许头信息流至消息或目标组件，如以下列表所述:

- JMS 导出请求

消息中接收到的 JMS 头将通过上下文服务传播到目标组件。消息中接收到的 JMS 属性将通过上下文服务传播到目标组件。

- JMS 导出响应

  如果在上下文服务中设置的任何 JMS 头字段未被 JMS 导出绑定中设置的静态属性覆盖，那么这些 JMS 头字段将在出站消息中使用。如果在上下文服务中设置的任何属性未被 JMS 导出绑定中设置的静态属性覆盖，那么这些属性将在出站消息中使用。

- JMS 导入请求

  如果在上下文服务中设置的任何 JMS 头字段未被 JMS 导入绑定中设置的静态属性覆盖，那么这些 JMS 头字段将在出站消息中使用。如果在上下文服务中设置的任何属性未被 JMS 导入绑定中设置的静态属性覆盖，那么这些属性将在出站消息中使用。

- JMS 导入响应

  消息中接收到的 JMS 头将通过上下文服务传播到目标组件。消息中接收到的 JMS 属性将通过上下文服务传播到目标组件。

JMS 临时动态响应目标相关方案:

临时动态响应目标相关方案导致为每个发送的请求创建一个唯一的动态队列或主题。

导入中指定的静态响应目标用于派生临时动态目标队列或主题的性质。此目标在请求的 **ReplyTo** 字段中设置，并且 JMS 导入将侦听该目标上的响应。接收到响应时，将使该响应在静态响应目标中重新排队以进行异步处理。响应的 **CorrelationID** 字段未使用，因此无需进行设置。

事务性问题

使用临时动态目标时，必须在使用已发送响应的线程中使用响应。必须在全局事务外部发送请求，并且必须在后端服务接收到该请求且返回响应之前落实该请求。

持久性

临时动态队列是生存期较短的实体，并且无法保证与静态队列或主题相关联的同一级别的持久性。服务器重新启动后，临时动态队列或主题将不再存在，消息也是如此。在使消息在静态响应目标中重新排队后，该消息将保持本身所定义的持久性。

超时

导入将在固定时间段内等待接收临时动态响应目标中的响应。如果设置了“SCA 响应到期时间”限定符，那么此时间间隔将来自该限定符；否则，该时间间隔缺省为 60 秒。超出等待时间后，导入将抛出 `ServiceTimeoutRuntimeException`。

外部客户机:

服务器可以向使用 JMS 绑定的外部客户机发送消息或接收来自这些客户机的消息。

外部客户机（例如 Web 门户网站或企业信息系统）可以向服务器中的 SCA 模块发送消息，也可以被服务器内的组件调用。

JMS 导出组件部署消息侦听器以侦听传入到导出绑定中指定的接收目标的请求。发送字段中指定的目标用于发送对入站请求的应答（如果所调用的应用程序提供应答）。因此，外部客户机能够使用导出绑定调用应用程序。

JMS 导入与外部客户机绑定，并且可以向这些外部客户机传递消息。此消息可能需要来自外部客户机的响应，也可能不需要此响应。

使用外部客户机:

外部客户机（也就是位于服务器外部的客户机）可能需要与安装在此服务器上的应用程序进行交互。

考虑一个非常简单的场景。在此场景中，外部客户机需要与服务器上的一个应用程序进行交互。下图说明了一种典型的简单场景。

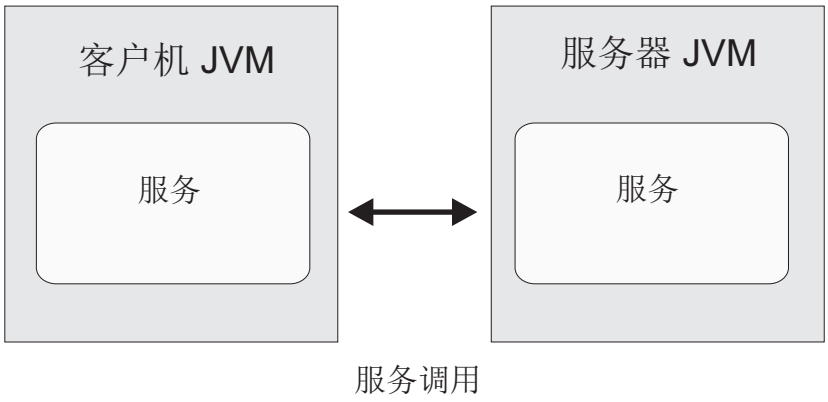


图 33. 简单用例场景：外部客户机与服务器应用程序进行交互。

SCA 应用程序包括导出和 JMS 绑定；这使得此应用程序可供外部客户机使用。

当您使 Java 虚拟机 (JVM) 中的外部客户机与服务器分隔开时，必须执行若干步骤以便建立连接以及与 JMS 导出进行交互。客户机获取具有正确值的初始上下文，然后通过 JNDI 查找资源。然后，客户机使用符合 JMS 1.1 规范的客户机来访问目标以及目标上的发送和接收消息。

由运行时自动创建的资源的缺省 JNDI 名称列示在此部分的配置主题中。但是，如果您具有预先创建的资源，请使用这些 JNDI 名称。

1. 配置 JMS 目标和连接工厂以发送消息。
2. 请确保 JNDI 上下文、SIB 资源适配器的端口以及消息传递引导端口都正确。

   服务器使用一些缺省端口，但是，如果该系统上安装了多个服务器，那么在安装时会创建一些备用端口，以避免与其他服务器实例发生冲突。可以使用管理控制台来确定您的服务器正在使用哪些端口。转至**服务器 → 应用程序服务器 → *your_server_name* → 配置**，然后单击**通信**下面的**端口**。然后，您可以编辑所使用的端口。
3. 客户机获取具有正确值的初始上下文，然后通过 JNDI 查找资源。
4. 通过使用 JMS 1.1 规范，客户机将访问目标以及目标上的发送和接收消息。

诊断 JMS 绑定:

可诊断并修正 JMS 绑定的问题。

实现异常

为响应各种错误情况，JMS 导入和导出实现可返回下列两种异常的其中之一:

- 服务业务异常: 如果在服务业务接口（WSDL 端口类型）上发生了指定的故障，那么系统将返回此异常。

- 服务运行时异常: 在所有其他情况下发生。在大多数情况下，原因异常将包含原始异常 (JMSException)。

  例如，对于每个请求消息，导入只应有一个响应消息。如果多个响应或延迟响应（SCA 响应已到期时响应到达）到达，那么系统将抛出服务运行时异常。事务将回滚，并且响应消息将退出队列或由失败事件管理器处理。

主要失败情况

JMS 绑定的主要失败情况由事务语义、JMS 提供程序配置或对其他组件中的现有行为的引用来确定。主要失败情况包括:

- 无法连接至 JMS 提供程序或目标。

  无法连接至 JMS 提供程序以接收消息将导致消息侦听器无法启动。此情况将记录在 WebSphere Application Server 日志中。持久消息将保留在目标上，直到它们被成功检索或到期。

  无法连接至 JMS 提供程序以发送出站消息将导致控制发送的事务回滚。
- 无法解析入站消息或构造出站消息。

  数据绑定或数据处理程序失败将导致控制该工作的事务回滚。
- 无法发送出站消息。

  无法发送消息将导致相关事务回滚。
- 多个延迟响应消息或意外延迟响应消息。

  对于每个请求消息，导入只应有一个响应消息。而且，可接收响应的有效时间段由请求上的 SCA 响应到期限定符确定。如果响应到达或超过到期时间，那么系统将删除关联记录。如果响应消息意外到达或延迟到达，那么系统将抛出服务运行时异常。
- 使用临时动态响应目标关联方案时，延迟响应导致的服务超时运行时异常。

  经过由 SCA 响应到期限定符确定的时间段（如果未设置，那么它将缺省为 60 秒）后，JMS 导入将超时。

基于 JMS 的 SCA 消息未出现在失败事件管理器中

如果 SCA 消息是因为 JMS 交互失败而产生的，那么您应在失败事件管理器中查找这些消息。如果这类消息未出现在失败事件管理器中，请确保 JMS 目标的底层 SIB 目标的最大失败传送次数值大于 **1**。如果将此值设置为 **2** 或更高，那么意味着系统允许在 JMS 绑定的 SCA 调用期间与失败事件管理器交互。

处理异常:

绑定的配置方式确定处理数据处理程序或数据绑定所发出的异常的方式。另外，调解流的性质规定了抛出这种异常时系统的行为。

在绑定调用数据处理程序或数据绑定时，可能会发生各种问题。例如，数据处理程序可能会接收到有效内容已损坏的消息，或者它可能尝试读取格式不正确的消息。

绑定处理这种异常的方式由您实现数据处理程序或数据绑定的方式确定。建议的行为是，您将数据绑定设置为抛出 **DataBindingException**。

抛出任何运行时异常（包括 **DataBindingException**）时:

- 如果调解流配置为事务性的，那么缺省情况下 JMS 消息将存储在失败事件管理器中以便手动回放或删除。

  注: 您可以更改绑定中的恢复方式，以便回滚消息，而不是将消息存储在失败事件管理器中。

- 如果调解流不是事务性的，那么将会记录该异常并且该消息将丢失。

对于数据处理程序，情况类似。由于数据处理程序被数据绑定调用，因此任何数据处理程序异常都将包装到数据绑定异常中。因此，**DataHandlerException** 将作为 **DataBindingException** 向您报告。

通用 JMS 绑定

通用 JMS 绑定提供面向符合 JMS 1.1 的第三方提供程序的连通性。通用 JMS 绑定的操作方式类似于 JMS 绑定的操作方式。

通过 JMS 绑定提供的服务允许服务组件体系结构 (SCA) 模块调用外部系统或者接收来自外部系统的消息。该系统可以是外部 JMS 系统。

通用 JMS 绑定提供与支持 JMS 1.1 且实现可选的 JMS 应用程序服务器设施但不符合 JCA 1.5 的 JMS 提供程序的集成。通用 JMS 绑定支持那些不支持 JCA 1.5 但支持 JMS 1.1 规范的应用程序服务器设施的 JMS 提供程序（包括 Oracle AQ、TIBCO、SonicMQ、WebMethods、BEA WebLogic 和 WebSphere MQ）。此绑定不支持 WebSphere 嵌入式 JMS 提供程序 (SIBJMS)（此提供程序是 JCA 1.5 JMS 提供程序）；使用 SIBJMS 时，请使用 第 88 页的『JMS 绑定』。

在 SCA 环境中与不符合 JCA 1.5 的基于 JMS 的系统集成时，请使用此通用绑定。这样目标外部应用程序就可以接收和发送消息以便与 SCA 组件集成。

通用 JMS 绑定概述:

通用 JMS 绑定是非 JCA 的 JMS 绑定，用于提供服务组件体系结构 (SCA) 环境与符合 JMS 1.1 且实现可选的 JMS 应用程序服务器设施的 JMS 系统之间的连接。

通用 JMS 绑定

通用 JMS 导入和导出绑定的主要方面包括:

- 侦听器端口: 使非基于 JCA 的 JMS 提供程序能够接收消息并将这些消息分派到消息驱动的 Bean (MDB)
- 连接: 封装客户机与提供程序应用程序之间的虚拟连接
- 目标: 由客户机用来指定它所生成的消息的目标或者它所使用的消息的源
- 认证数据: 用于保护对绑定的访问

通用 JMS 导入绑定

通用 JMS 导入绑定允许 SCA 模块内的组件与不符合 JCA 1.5 的外部 JMS 提供程序所提供的服务通信。

JMS 导入的连接部分是连接工厂。连接工厂（这是客户机用来创建提供程序连接的对象）封装管理员所定义的一组连接配置参数。每个连接工厂都是 ConnectionFactory、QueueConnectionFactory 或 TopicConnectionFactory 接口的实例。

与外部 JMS 系统的交互包括使用目标来发送请求和接收应答。

根据所调用的操作的类型，支持两种类型的通用 JMS 导入绑定使用方案:

- 单向: 通用 JMS 导入将消息放入导入绑定所配置的发送目标中。将不发送任何内容到 JMS 头的 replyTo 字段。
- 双向（请求-响应）: 通用 JMS 导入将消息放入发送目标中，然后坚持等待它将从 SCA 组件接收到的应答。

  在出站消息的 replyTo 头属性中设置接收目标。将部署消息驱动的 Bean (MDB) 以侦听接收目标，并且在接收到应答时，MDB 将该应答传递回组件。

可以将导入绑定配置为（通过使用 Integration Designer 中的**响应相关方案**）要求响应消息相关标识是从请求消息标识（缺省值）或者从请求消息相关标识复制的。

对于单向和双向使用方案，都可以指定动态和静态头属性。可以从通用 JMS 导入方法绑定设置静态属性。这些属性中的部分属性对 SCA JMS 运行时具有特殊意义。

请务必注意，通用 JMS 是异步绑定。如果调用组件以同步方式调用通用 JMS 导入（对于双向操作），那么在 JMS 服务返回响应之前，该调用组件将被阻塞。

图 34 说明导入如何链接到外部服务。

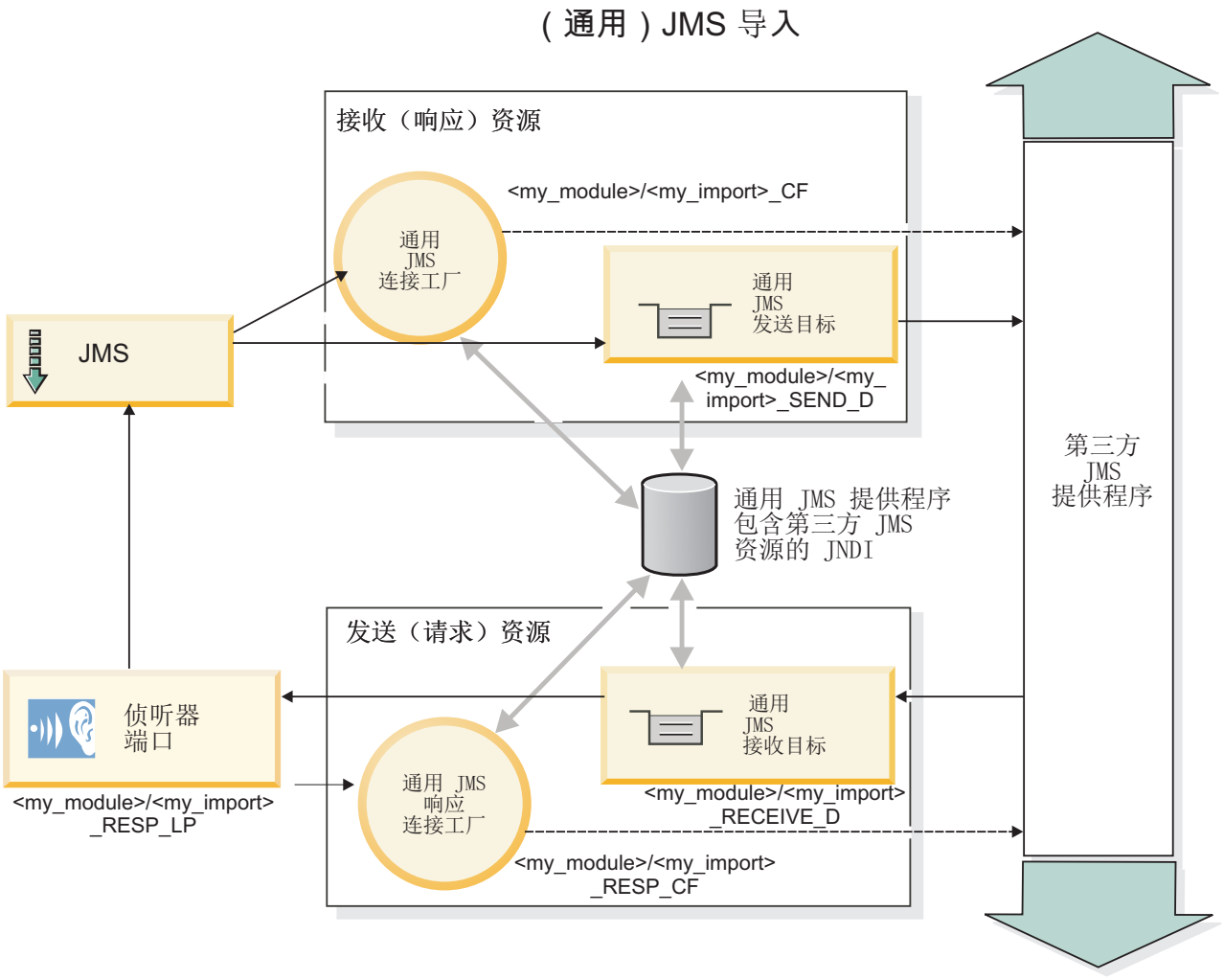


图 34. 通用 JMS 导入绑定资源

通用 JMS 导出绑定

通用 JMS 导出绑定为 SCA 模块提供了用于向外部 JMS 应用程序提供服务的方法。

JMS 导出的连接部分由 ConnectionFactory 和 ListenerPort 组成。

通用 JMS 导出具有发送和接收目标。

- 接收目标是应该用来放置目标组件的入局消息的位置。

- 发送目标是将发送应答的位置，除非入局消息使用 replyTo 头属性覆盖了此目标。

将部署 MDB 以侦听传入到导出绑定中指定的接收目标的请求。

- 发送字段中指定的目标用于发送对入站请求的应答（如果所调用的组件提供应答）。
- 在入局消息的 replyTo 字段中指定的目标将覆盖在发送字段中指定的目标。
- 对于请求-响应方案，可以将导入绑定配置为（通过使用 Integration Designer 中的**响应相关方案**字段）要求响应将请求消息标识复制到响应消息的相关标识字段（缺省值），否则响应可以将请求相关标识复制到响应消息的相关标识字段。

图 35 说明外部请求程序如何链接到导出。

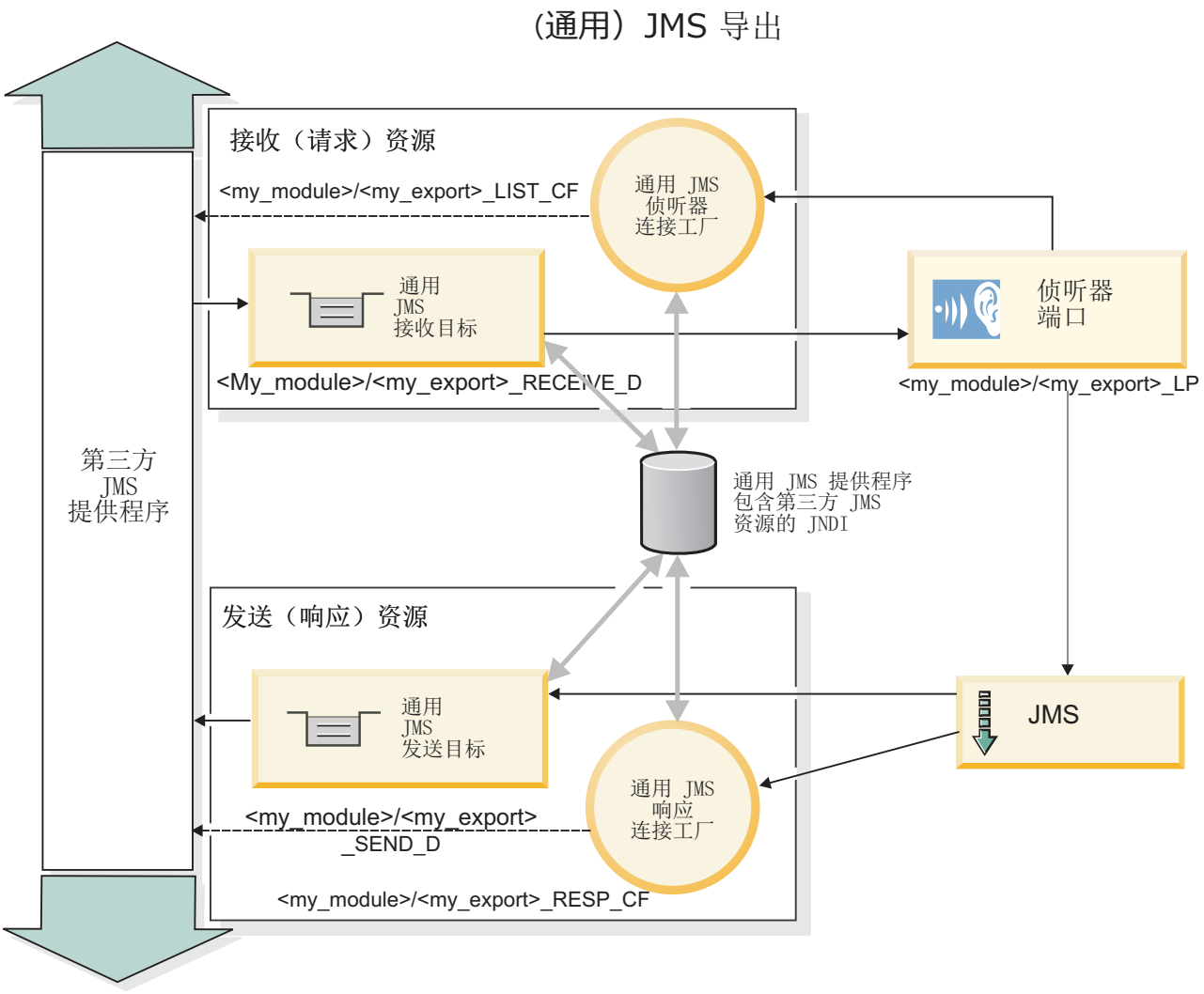


图 35. 通用 JMS 导出绑定资源

通用 JMS 绑定的关键功能:

通用 JMS 导入和导出绑定的功能与 WebSphere 嵌入式 JMS 和 MQ JMS 导入绑定的功能一致。关键功能包括头定义以及对现有 Java EE 资源的访问。但是，因为其通用性质，没有特定于 JMS 提供程序的连接选项，并且此绑定在部署和安装时生成资源的能力受限。

通用导入

与 MQ JMS 导入应用程序一样，通用 JMS 实现是异步的，并且支持三种调用：单向、双向（又称为请求-响应）和回调。

部署 JMS 导入时，系统将部署运行时环境提供的消息驱动的 Bean (MDB)。MDB 将侦听请求消息的应答。MDB 与随 JMS 消息的 replyTo 头字段中的请求发送的目标相关联（即，侦听该目标）。

通用导出

在处理结果返回时，通用 JMS 导出绑定不同于 EIS 导出绑定。通用 JMS 导出将响应显式发送至入局消息中指定的 replyTo 目标。如果未指定，那么系统将使用该发送目标。

部署通用 JMS 导出时，系统将部署消息驱动的 Bean（此 MDB 不同于用于通用 JMS 导入的 MDB）。它将侦听接收目标上的入局请求，然后将这些请求分派给 SCA 运行时处理。

特殊头

通用 JMS 导入和导出中的特殊头属性用于告诉目标绑定如何处理该消息。

例如，缺省函数选择器使用 TargetFunctionName 属性来标识导出接口中要调用的操作的名称。

注：导入绑定可配置为将 TargetFunctionName 头设置为每个操作的名称。

Java EE 资源

将 JMS 绑定部署至 Java EE 环境时，系统会创建许多 Java EE 资源。

- 侦听器端口，用于在接收（响应）目标（仅双向）上侦听导入以及在接收（请求）目标上侦听导出。
- 用于 outboundConnection（导入）和 inboundConnection（导出）的通用 JMS 连接工厂
- 用于发送（导入）目标和接收（导出；仅双向）目标的通用 JMS 目标
- 用于 responseConnection 的通用 JMS 连接工厂（仅双向且可选；或者，对导入使用 outboundConnection，对导出使用 inboundConnection）
- 用于接收（导入）目标和发送（导出）目标（仅双向）的通用 JMS 目标
- 用于访问 SIB 回调队列目标的缺省消息传递提供程序回调 JMS 目标（仅双向）
- 用于访问回调 JMS 目标的缺省消息传递提供程序回调 JMS 连接工厂（仅双向）
- SIB 回调队列目标，用于存储要在响应处理期间使用的请求消息的信息（仅双向）

安装任务根据导入和导出文件中的信息创建 ConnectionFactory、三个目标和 ActivationSpec。

通用 JMS 头:

通用 JMS 头是包含通用 JMS 消息属性的所有属性的服务数据对象 (SDO)。这些属性可以来自入站消息，也可以是将应用于出站消息的属性。

JMS 消息包含两种类型的头：JMS 系统头和多个 JMS 属性。这两种类型的头都可以在服务消息对象 (SMO) 中的调解模块中访问，也可以通过使用 ContextService API 来访问。

以静态方式设置了 methodBinding 中的下列属性:

- JMSType
- JMSCorrelationID
- JMSDeliveryMode
- JMSPriority

通用 JMS 绑定还支持使用与 JMS 和 MQ JMS 绑定相同的方式动态修改 JMS 头和属性。

某些通用 JMS 提供程序对应用程序可以设置的属性以及属性组合进行了限制。您必须参阅第三方产品文档以了解更多信息。但是，已向 methodBinding 添加另一个属性 ignoreInvalidOutboundJMSProperties，该属性允许传播任何异常。

仅当基本服务组件体系结构 SCDL 绑定开关已开启时，才使用通用 JMS 头和消息属性。开启此开关后，将传播上下文信息。缺省情况下，此开关处于开启状态。要阻止传播上下文信息，请将值更改为 **false**。

启用上下文传播后，将允许头信息流至消息或目标组件。要开启和关闭上下文传播，请对导入和导出绑定的 contextPropagationEnabled 属性指定 **true** 或 **false**。例如:

```
<esbBinding xsi:type="eis:JMSImportBinding" contextProgagationEnabled="true">
```

缺省值为 **true**。

诊断通用 JMS 绑定:

可诊断并修正通用 MQ JMS 绑定的问题。

实现异常

为响应各种错误情况，通用 JMS 导入和导出实现可返回下列两种异常的其中之一:

- 服务业务异常: 如果在服务业务接口（WSDL 端口类型）上发生了指定的故障，那么系统将返回此异常。
- 服务运行时异常: 在所有其他情况下发生。在大多数情况下，原因异常将包含原始异常 (JMSException)。

诊断通用 JMS 消息传递到期

JMS 提供程序将使请求消息到期。

*请求到期*是指，到达请求消息上的 JMSExpiration 时间时，JMS 提供程序使请求消息到期。与其他 JMS 绑定一样，通用 JMS 绑定通过将导入放置的回调消息的到期设置为外发请求相同来处理请求到期。请注意，回调消息的到期将指示请求消息已到期，并且应通过业务异常来通知客户机。

但是，如果回调目标移至第三方提供程序，那么此类型的请求到期不受支持。

*响应到期*是指，到达响应消息上的 JMSExpiration 时间时，JMS 提供程序使响应消息到期。

通用 JMS 绑定的响应到期不受支持，因为未定义第三方 JMS 提供程序的确切到期行为。但是，如果接收到响应，请在接收到响应时检查该响应是否到期。

对于出站请求消息，JMSExpiration 值将根据等待时间和 asyncHeader 中的 requestExpiration 值（如果设置了此值）计算。

诊断通用 JMS 连接工厂错误

在通用 JMS 提供程序中定义某些类型的连接工厂时，如果尝试启动应用程序，那么您可能会接收到错误消息。可修改外部连接工厂以避免此问题。

启动应用程序时，您可能会接收到以下错误消息:

```
MDB 侦听器端口 JMSConnectionFactory 类型不匹配
JMSDestination 类型
```

您定义外部连接工厂时可能发生此问题。具体地说，您创建 JMS 1.0.2 主题连接工厂而不是 JMS 1.1（统一）连接工厂（即，能够支持点到点和发布/预订通信的连接工厂）时，可能会抛出该异常。

要解决此问题，请执行以下步骤:

1. 访问您要使用的通用 JMS 提供程序。
2. 将您定义的 JMS 1.0.2 主题连接工厂替换为 JMS 1.1（统一）连接工厂。

使用新定义的 JMS 1.1 连接工厂启动应用程序时，不应再接收到错误消息。

通用的基于 JMS 的 SCA 消息未出现在失败事件管理器中

如果 SCA 消息是因为通用 JMS 交互失败而产生的，那么您应在失败事件管理器中查找这些消息。如果这些消息未出现在失败事件管理器中，请确保底层侦听器端口上的最大重试次数值等于或大于 1。如果将此值设置为 1 或更高，那么意味着系统允许在通用 JMS 绑定的 SCA 调用期间与失败事件管理器交互。

处理异常:

绑定的配置方式确定处理数据处理程序或数据绑定所发出的异常的方式。另外，调解流的性质规定了抛出这种异常时系统的行为。

在绑定调用数据处理程序或数据绑定时，可能会发生各种问题。例如，数据处理程序可能会接收到有效内容已损坏的消息，或者它可能尝试读取格式不正确的消息。

绑定处理这种异常的方式由您实现数据处理程序或数据绑定的方式确定。建议的行为是，您将数据绑定设置为抛出 **DataBindingException**。

对于数据处理程序，情况类似。由于数据处理程序被数据绑定调用，因此任何数据处理程序异常都将包装到数据绑定异常中。因此，**DataHandlerException** 将作为 **DataBindingException** 向您报告。

抛出任何运行时异常（包括 **DataBindingException** 异常）时:

- 如果调解流配置为事务性的，那么缺省情况下 JMS 消息将存储在失败事件管理器中以便手动回放或删除。

  注: 您可以更改绑定中的恢复方式，以便回滚消息，而不是将消息存储在失败事件管理器中。
- 如果调解流不是事务性的，那么将会记录该异常并且该消息将丢失。

对于数据处理程序，情况类似。由于数据处理程序被数据绑定调用，因此数据处理程序异常将生成到数据绑定异常内。因此，**DataHandlerException** 将作为 **DataBindingException** 向您报告。

WebSphere MQ JMS 绑定

WebSphere MQ JMS 绑定提供与使用基于 WebSphere MQ JMS 的提供程序的外部应用程序的集成。

在服务器环境中，可使用 WebSphere MQ JMS 导出和导入绑定直接与外部 JMS 或 MQ JMS 系统交互。这样就不需要使用服务集成总线的 MQ 链接或客户机链接功能。

当组件通过导入与基于 WebSphere MQ JMS 的服务交互时，WebSphere MQ JMS 导入绑定将使用数据将发送至的目标以及将接收应答的目标。将数据转换成 JMS 消息以及将 JMS 消息转换成数据是通过 JMS 数据处理程序或数据绑定边缘组件完成的。

当 SCA 模块为 WebSphere MQ JMS 客户机提供服务时，WebSphere MQ JMS 导出绑定将使用可以接收到请求以及可以发送响应的目标。将数据转换成 JMS 消息以及将 JMS 消息转换成数据是通过 JMS 数据处理程序或数据绑定完成的。

函数选择器提供到要调用的目标组件上的操作的映射。

WebSphere MQ JMS 绑定概述:

WebSphere MQ JMS 绑定提供与使用 WebSphere MQ JMS 提供程序的外部应用程序的集成。

WebSphere MQ 管理任务

在运行包含 WebSphere MQ JMS 绑定的应用程序之前，WebSphere MQ 系统管理员应该创建这些绑定将使用的顶层 WebSphere MQ 队列管理器。

WebSphere MQ JMS 导入绑定

WebSphere MQ JMS 导入允许 SCA 模块内的组件与基于 WebSphere MQ JMS 的提供程序所提供的服务通信。您必须正在使用受支持的 WebSphere MQ 版本。您可以在 IBM 支持页面上找到详细的硬件和软件需求。

根据所调用的操作的类型，支持两种类型的 WebSphere MQ JMS 导入绑定使用方案:

- 单向: WebSphere MQ JMS 导入将消息放入导入绑定所配置的发送目标中。将不发送任何内容到 JMS 头的 replyTo 字段。
- 双向（请求-响应）: WebSphere MQ JMS 导入将消息放入发送目标中。

  在 replyTo 头字段中设置接收目标。将部署消息驱动的 Bean (MDB) 以侦听接收目标，并且在接收到应答时，MDB 将该应答传递回组件。

  可以将导入绑定配置为（通过使用 Integration Designer 中的**响应相关方案**）要求响应消息相关标识是从请求消息标识（缺省值）或者从请求消息相关标识复制的。

对于单向和双向使用方案，都可以指定动态和静态头属性。可以从 JMS 导入方法绑定设置静态属性。这些属性中的部分属性对 SCA JMS 运行时具有特殊意义。

请务必注意，WebSphere MQ JMS 是异步绑定。如果调用组件以同步方式调用 WebSphere MQ JMS 导入（对于双向操作），那么在 JMS 服务返回响应之前，该调用组件将被阻塞。

第 103 页的图 36 说明导入如何链接到外部服务。

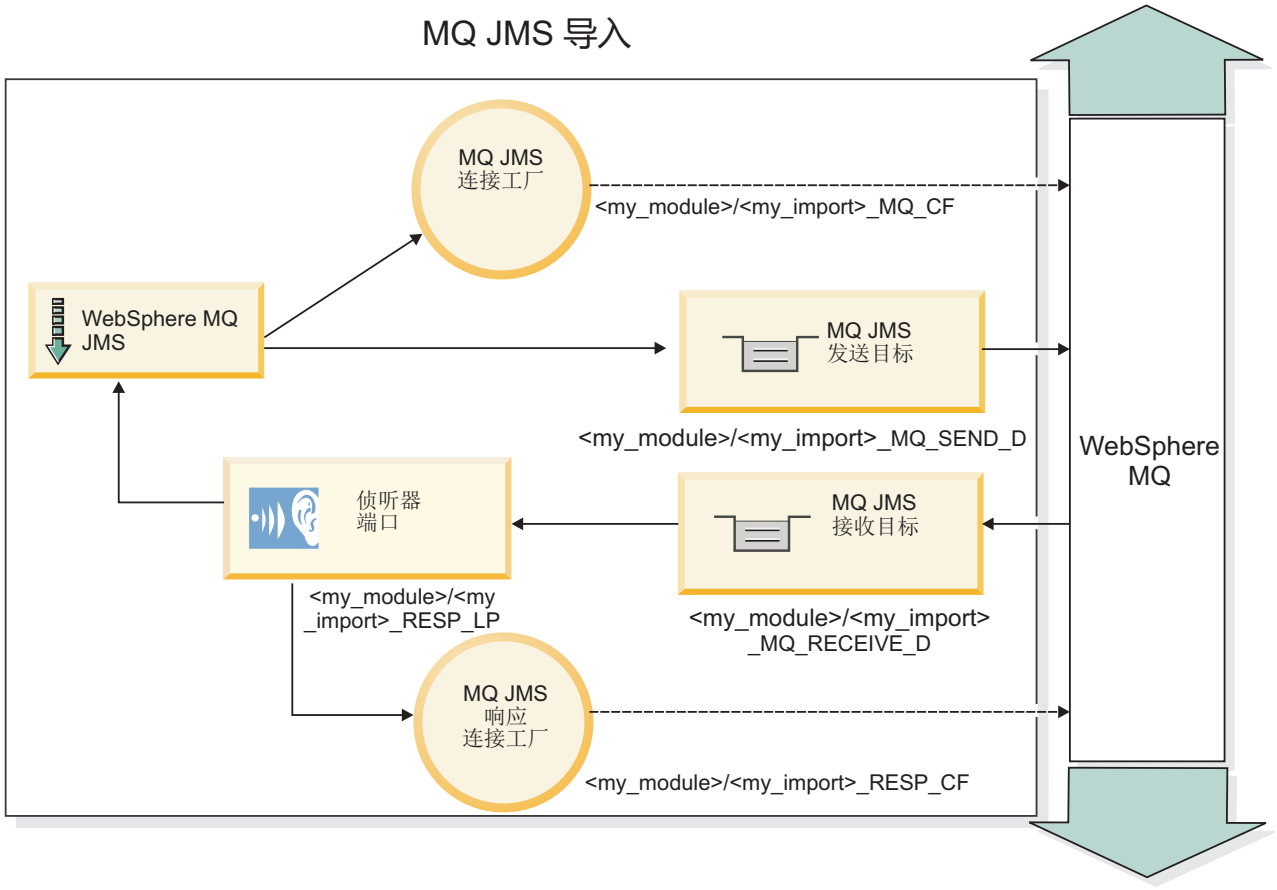


图 36. WebSphere MQ JMS 导入绑定资源

WebSphere MQ JMS 导出绑定

WebSphere MQ JMS 导出绑定为 SCA 模块提供了用于向基于 WebSphere MQ 的 JMS 提供程序中的外部 JMS 应用程序提供服务的方法。

将部署 MDB 以侦听传入到导出绑定中指定的接收目标的请求。发送字段中指定的目标用于发送对入站请求的应答（如果所调用的组件提供应答）。在响应消息的 replyTo 字段中指定的目标将覆盖在发送字段中指定的目标。

第 104 页的图 37 说明外部请求程序如何链接到导出。

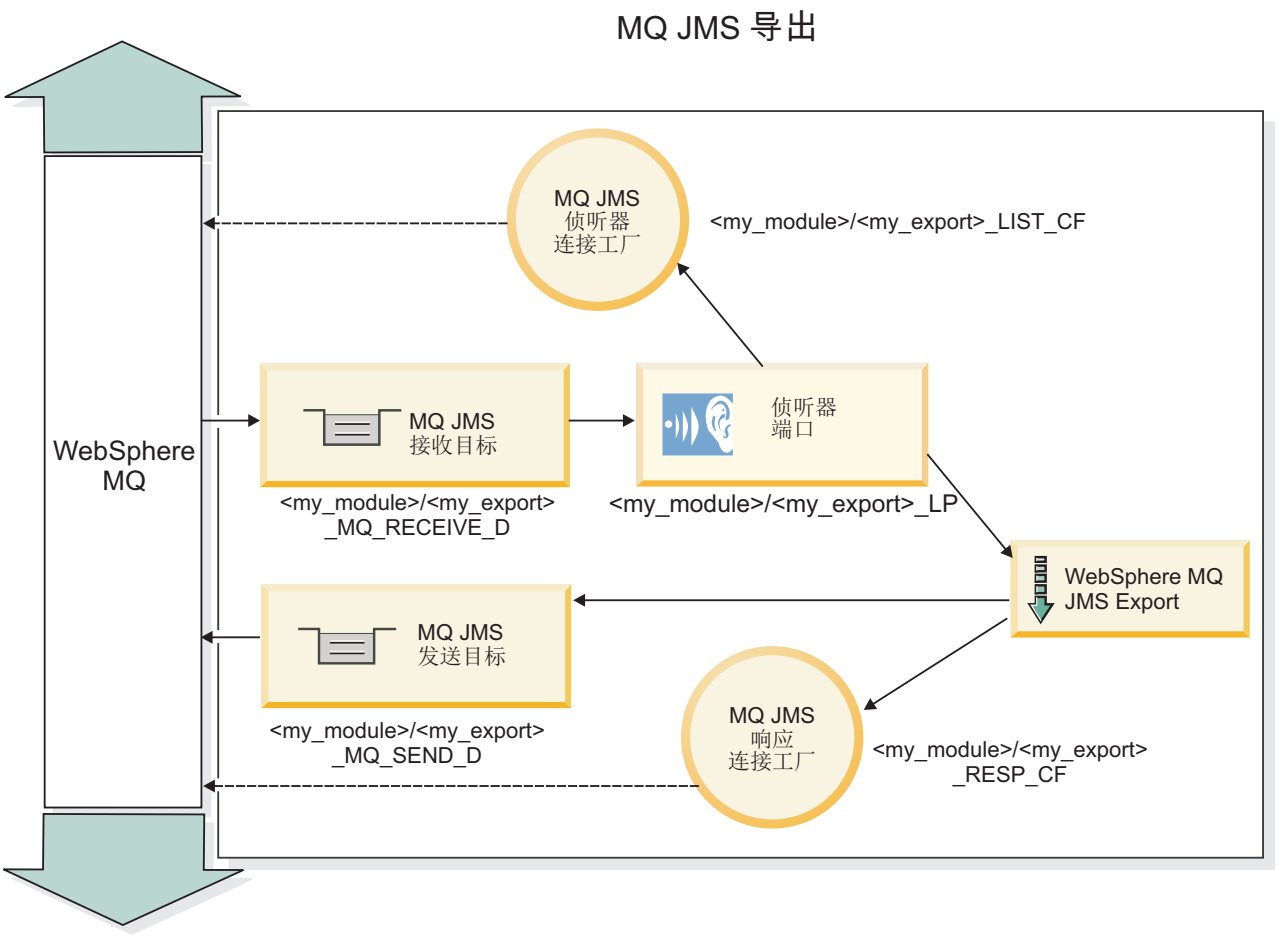


图 37. WebSphere MQ JMS 导出绑定资源

注: 第 103 页的图 36 和图 37 说明先前版本的 IBM Business Process Manager 中的应用程序如何链接至外部服务。对于为 IBM Business Process Manager V7.0 开发的应用程序，将使用激活规范来代替侦听器端口和连接工厂。

WebSphere MQ JMS 绑定的主要特征:

WebSphere MQ JMS 绑定的主要特征包括头、Java EE 工件和已创建的 Java EE 资源。

头

JMS 消息头包含许多预定义的字段，这些字段中包含客户机和提供程序用来标识和路由消息的值。可以使用绑定属性为这些头配置固定值，也可以在运行时动态指定这些头。

JMSCorrelationID
链接至相关消息。通常，此字段设置为要应答的消息的消息标识字符串。

TargetFunctionName
所提供的其中一个函数选择器使用此头来标识要调用的操作。在发送至 JMS 导出的消息中设置 JMS 头属性 TargetFunctionName 将允许使用此函数选择器。可以直接在JMS 客户机应用程序中设置此属性，也可以在将具有 JMS 绑定的导入连接至这样的导出时设置此属性。在这种情况下，应配置 JMS 导入绑定，以将接口中每项操作的 TargetFunctionName 头设置为此操作的名称。

关联方案

WebSphere MQ JMS 绑定提供了各种关联方案，这些方案用来确定如何使请求消息与响应消息关联。

RequestMsgIDToCorrelID
JMSMessageID 已复制到 JMSCorrelationID 字段。这是缺省设置。

RequestCorrelIDToCorrelID
JMSCorrelationID 已复制到 JMSCorrelationID 字段。

Java EE 资源

将 MQ JMS 导入部署到 Java EE 环境时，将创建许多 Java EE 资源。

参数

MQ 连接工厂
客户机用来与 MQ JMS 提供程序建立连接。

响应连接工厂
当发送目标与接收目标位于不同的队列管理器上时，SCA MQ JMS 运行时将使用响应连接工厂。

激活规范
MQ JMS 激活规范与一个或多个消息驱动的 Bean 相关联，并提供使这些 Bean 接收消息所必需的配置。

目标

- 发送目标:
  - 导入: 将发送请求或出局消息的位置。
  - 导出: 将发送响应消息的位置（前提是此消息未被入局消息的 JMSReplyTo 头字段取代）。
- 接收目标:
  - 导入: 应该将响应或入局消息放置到的位置。
  - 导出: 应该将入局消息或请求消息放置到的位置。

JMS 头:

JMS 消息包含两种类型的头: JMS 系统头和多个 JMS 属性。这两种类型的头都可以在服务消息对象 (SMO) 中的调解模块中访问，也可以通过使用 ContextService API 来访问。

JMS 系统头

JMS 系统头在 SMO 中由 JMSHeader 元素表示，该元素包含所有通常在 JMS 头中找到的字段。虽然可以在调解中（或者使用 ContextService）修改这些字段，但在 SMO 中设置的某些 JMS 系统头字段不会在出站 JMS 消息中传播，这是因为这些字段将被系统或静态值覆盖。

可以在调解中（或者使用 ContextService）更新的关键 JMS 系统头字段如下所示:

- **JMSType** 和 **JMSCorrelationID** - 特定预定义的消息头属性的值
- **JMSDeliveryMode** - 传递方式的值（persistent 或 nonpersistent; 缺省值为 persistent）
- **JMSPriority** - 优先级值（0 到 9; 缺省值为 JMS_Default_Priority）

JMS 属性

JMS 属性在 SMO 中表示为“属性”列表中的条目。可以在调解中或者使用 ContextService API 来添加、更新或删除属性。

也可以在 JMS 绑定中以静态方式设置属性。以静态方式设置的属性将覆盖以动态方式设置的同名设置。

从其他绑定（例如 HTTP 绑定）传播的用户属性将在 JMS 绑定作为 JMS 属性输出。

头传播设置

将 JMS 系统头和属性从入站 JMS 消息传播到下游组件或者从上游组件传播到出站 JMS 消息可以由绑定中的“传播协议头”标志控制。

设置“传播协议头”后，将允许头信息流至消息或目标组件，如以下列表所述:

- JMS 导出请求

  消息中接收到的 JMS 头将通过上下文服务传播到目标组件。消息中接收到的 JMS 属性将通过上下文服务传播到目标组件。
- JMS 导出响应

  如果在上下文服务中设置的任何 JMS 头字段未被 JMS 导出绑定中设置的静态属性覆盖，那么这些 JMS 头字段将在出站消息中使用。如果在上下文服务中设置的任何属性未被 JMS 导出绑定中设置的静态属性覆盖，那么这些属性将在出站消息中使用。
- JMS 导入请求

  如果在上下文服务中设置的任何 JMS 头字段未被 JMS 导入绑定中设置的静态属性覆盖，那么这些 JMS 头字段将在出站消息中使用。如果在上下文服务中设置的任何属性未被 JMS 导入绑定中设置的静态属性覆盖，那么这些属性将在出站消息中使用。
- JMS 导入响应

  消息中接收到的 JMS 头将通过上下文服务传播到目标组件。消息中接收到的 JMS 属性将通过上下文服务传播到目标组件。

外部客户机:

服务器可以向使用 WebSphere MQ JMS 绑定的外部客户机发送消息或接收来自这些客户机的消息。

外部客户机（例如 Web 门户网站或企业信息系统）可以通过导出向应用程序中的 SCA 组件发送消息，也可以被应用程序中的 SCA 组件通过导入进行调用。

WebSphere MQ JMS 导出绑定部署消息驱动的 Bean (MDB) 以侦听传入到该导出绑定中指定的接收目标的请求。发送字段中指定的目标用于发送对入站请求的应答（如果所调用的应用程序提供应答）。因此，外部客户机能够通过导出绑定调用应用程序。

WebSphere MQ JMS 导入与外部客户机绑定，并且可以向这些外部客户机传递消息。此消息可能需要来自外部客户机的响应，也可能不需要此响应。

您可以在 WebSphere MQ 信息中心中找到有关如何与使用 WebSphere MQ 的外部客户机交互的更多信息。

诊断 WebSphere MQ JMS 绑定:

可诊断并修正 WebSphere MQ JMS 绑定的问题。

实现异常

为响应各种错误情况，MQ JMS 导入和导出实现可返回下列两种异常的其中之一:

- 服务业务异常: 如果在服务业务接口（WSDL 端口类型）上发生了指定的故障，那么系统将返回此异常。
- 服务运行时异常: 在所有其他情况下发生。在大多数情况下，原因异常将包含原始异常 (JMSException)。

  例如，对于每个请求消息，导入只应有一个响应消息。如果多个响应或延迟响应（SCA 响应已到期时响应到达）到达，那么系统将抛出服务运行时异常。事务将回滚，并且响应消息将退出队列或由失败事件管理器处理。

基于 WebSphere MQ JMS 的 SCA 消息未出现在失败事件管理器中

如果 SCA 消息是因为 WebSphere MQ JMS 交互失败而产生的，那么您应在失败事件管理器中查找这些消息。如果这些消息未出现在失败事件管理器中，请确保底层侦听器端口上的最大重试次数值等于或大于 **1**。如果将此值设置为 **1** 或更高，那么意味着系统允许在 MQ JMS 绑定的 SCA 调用期间与失败事件管理器交互。

误用场景: 与 WebSphere MQ 绑定比较

WebSphere MQ JMS 绑定用来与针对 WebSphere MQ 部署的 JMS 应用程序互操作，它根据 JMS 消息模型展示消息。但是，WebSphere MQ 导入和导出主要用来与本机 WebSphere MQ 应用程序交互并对调解展示 WebSphere MQ 消息体的完整内容。

以下场景应使用 WebSphere MQ JMS 绑定（而不是 WebSphere MQ 绑定）构建:

- 从 SCA 模块调用 JMS 消息驱动的 Bean (MDB)，其中 MDB 是针对 WebSphere MQ JMS 提供程序部署的。请使用 WebSphere MQ JMS 导入。
- 允许通过 JMS 从 Java EE 组件 Servlet 或 EJB 调用 SCA 模块。请使用 WebSphere MQ JMS 导出。
- 在 WebSphere MQ 中传输时调解 JMS MapMessage 的内容。将 WebSphere MQ JMS 导出和导入与相应数据处理程序或数据绑定配合使用。

有时 WebSphere MQ 绑定和 WebSphere MQ JMS 绑定可能应该互操作。特别是，当您在 Java EE 与非 Java EE WebSphere MQ 应用程序之间进行桥接时，应将 WebSphere MQ 导出和 WebSphere MQ JMS 导入（或相反）与相应数据绑定和/或调解模块配合使用。

处理异常:

绑定的配置方式确定处理数据处理程序或数据绑定所发出的异常的方式。另外，调解流的性质规定了抛出这种异常时系统的行为。

在绑定调用数据处理程序或数据绑定时，可能会发生各种问题。例如，数据处理程序可能会接收到有效内容已损坏的消息，或者它可能尝试读取格式不正确的消息。

绑定处理这种异常的方式由您实现数据处理程序或数据绑定的方式确定。建议的行为是，您将数据绑定设置为抛出 **DataBindingException**。

对于数据处理程序，情况类似。由于数据处理程序被数据绑定调用，因此任何数据处理程序异常都将包装到数据绑定异常中。因此，**DataHandlerException** 将作为 **DataBindingException** 向您报告。

抛出任何运行时异常（包括 **DataBindingException** 异常）时:

- 如果调解流配置为事务性的，那么缺省情况下 JMS 消息将存储在失败事件管理器中以便手动回放或删除。

  注: 您可以更改绑定中的恢复方式，以便回滚消息，而不是将消息存储在失败事件管理器中。

- 如果调解流不是事务性的，那么将会记录该异常并且该消息将丢失。

对于数据处理程序，情况类似。由于数据处理程序被数据绑定调用，因此数据处理程序异常将生成到数据绑定异常内。因此，**DataHandlerException** 将作为 **DataBindingException** 向您报告。

WebSphere MQ 绑定

WebSphere MQ 绑定提供与 WebSphere MQ 应用程序的服务组件体系结构 (SCA) 连接。

在服务器环境中，可使用 WebSphere MQ 导出和导入绑定直接与基于 WebSphere MQ 的系统交互。这样就不需要使用服务集成总线的 MQ 链接或客户机链接功能。

当组件通过导入与 WebSphere MQ 服务交互时，WebSphere MQ 导入绑定将使用数据将发送至的队列以及将接收应答的队列。

当 SCA 模块为 WebSphere MQ 客户机提供服务时，WebSphere MQ 导出绑定将使用可以接收请求以及可以发送响应的队列。函数选择器提供到要调用的目标组件上的操作的映射。

将有效内容数据转换成 MQ 消息以及将 MQ 消息转换成有效内容数据是通过 MQ 主体数据处理程序或数据绑定完成的。将头数据转换成 MQ 消息以及将 QA 消息转换成头数据是通过 MQ 头数据绑定完成的。

有关受支持的 WebSphere MQ 版本的信息，请参阅系统需求 Web 页面: IBM Business Process Manager 系统需求。

WebSphere MQ 绑定概述:

WebSphere MQ 绑定提供与基于 MQ 的本机应用程序的集成。

WebSphere MQ 管理任务

在运行包含 WebSphere MQ 绑定的应用程序之前，WebSphere MQ 系统管理员应该创建这些绑定将使用的顶层 WebSphere MQ 队列管理器。

WebSphere 管理任务

您必须将 Websphere 中的 MQ 资源适配器的**本机库路径**属性设置为服务器所支持的 WebSphere MQ 版本，然后重新启动服务器。这将确保正在使用受支持的 WebSphere MQ 版本的库。您可以在 IBM 支持页面上找到详细的硬件和软件需求。

WebSphere MQ 导入绑定

WebSphere MQ 导入绑定允许 SCA 模块内的组件与基于 WebSphere MQ 的外部应用程序所提供的服务通信。您必须正在使用受支持的 WebSphere MQ 版本。您可以在 IBM 支持页面上找到详细的硬件和软件需求。

与外部 WebSphere MQ 系统的交互包括使用队列来发送请求和接收应答。

根据所调用的操作的类型，支持两种类型的 WebSphere MQ 导入绑定使用方案:

- 单向: WebSphere MQ 导入将消息放入导入绑定的**发送目标队列**字段中配置的队列中。将不发送任何内容到 MQMD 头的 replyTo 字段。
- 双向（请求-响应）: WebSphere MQ 导入将消息放入在**发送目标队列**字段中配置的队列中。

  在 replyTo MQMD 头字段中设置接收队列。将部署消息驱动的 Bean (MDB) 以侦听接收队列，并且在接收到应答时，MDB 将该应答传递回组件。

可以将导入绑定配置为（通过使用**响应相关方案**）要求响应消息相关标识是从请求消息标识（缺省值）或者从请求消息相关标识复制的。

请务必注意，WebSphere MQ 是异步绑定。如果调用组件以同步方式调用 WebSphere MQ 导入（对于双向操作），那么在 WebSphere MQ 服务返回响应之前，该调用组件将被阻塞。

图 38 说明导入如何链接到外部服务。

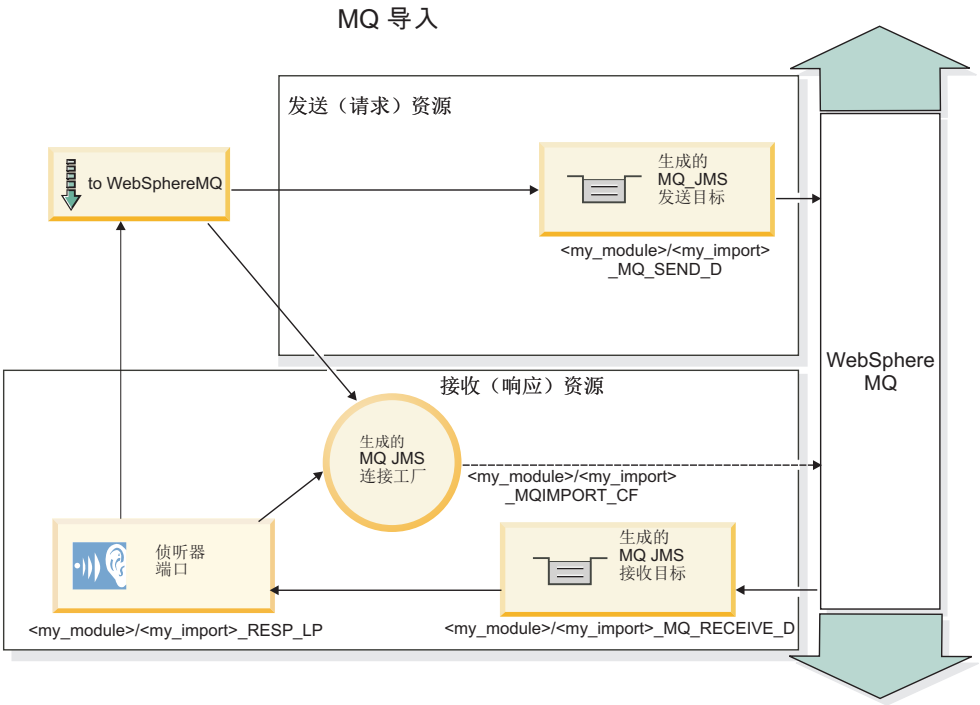


图 38. WebSphere MQ 导入绑定资源

WebSphere MQ 导出绑定

WebSphere MQ 导出绑定为 SCA 模块提供了用于向基于 WebSphere MQ 的外部应用程序提供服务的方法。

将部署 MDB 以侦听传入到导出绑定中指定的**接收目标队列**的请求。**发送目标队列**字段中指定的队列用于发送对入站请求的应答（如果所调用的组件提供应答）。在响应消息的 replyTo 字段中指定的队列将覆盖在**发送目标队列**字段中指定的队列。

第 110 页的图 39 说明外部请求程序如何链接到导出。

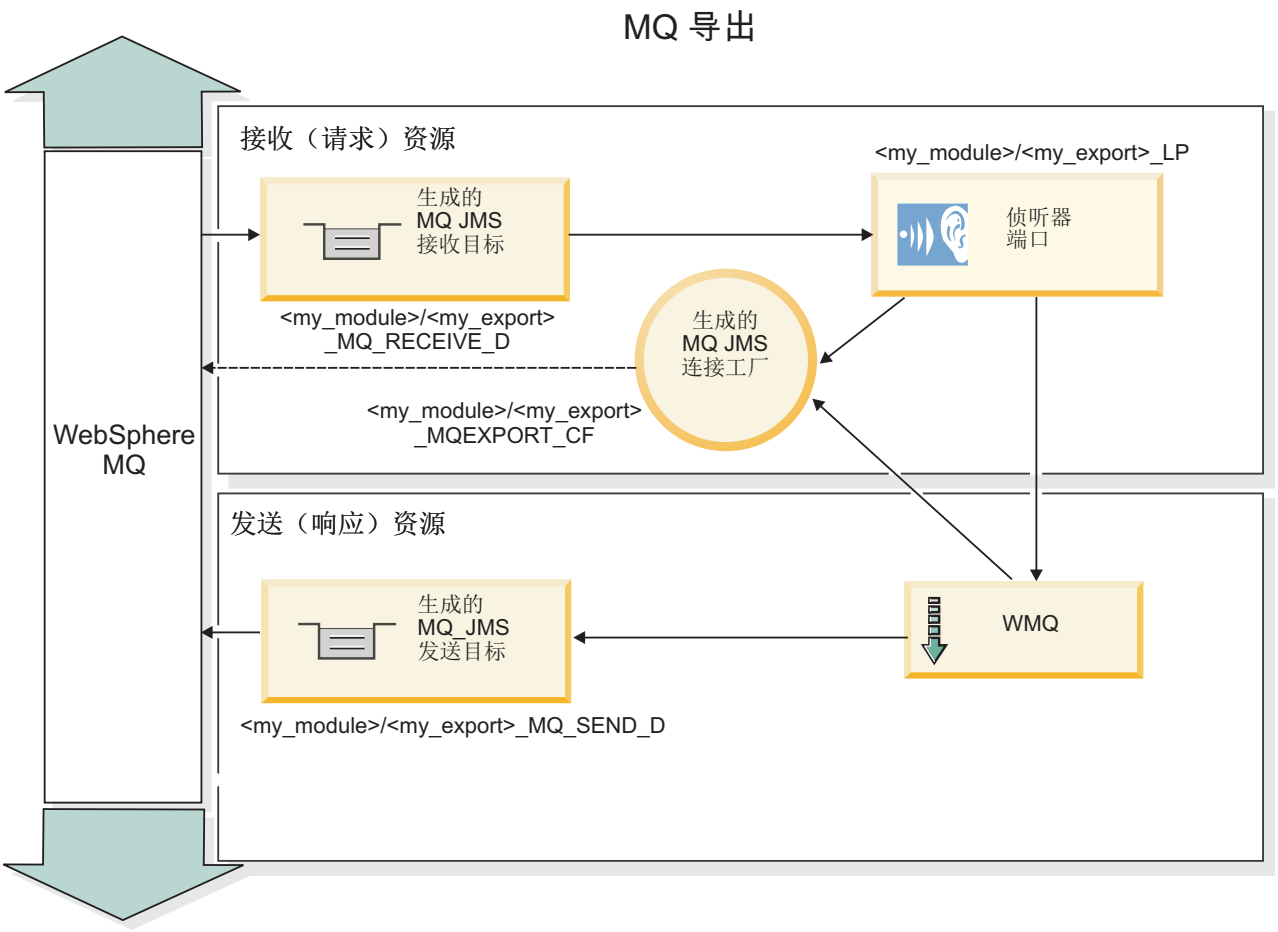


图 39. WebSphere MQ 导出绑定资源

注: 第 109 页的图 38 和图 39 说明先前版本的 IBM Business Process Manager 中的应用程序如何链接至外部服务。对于为 IBM Business Process Manager V7.0 开发的应用程序，将使用激活规范来代替侦听器端口和连接工厂。

WebSphere MQ 绑定的主要特征:

WebSphere MQ 绑定的主要特征包括头、Java EE 工件和已创建的 Java EE 资源。

关联方案

WebSphere MQ 请求-应答应用程序可以使用许多技术当中的一种技术使响应消息与针对 MQMD MessageID 和 CorrelID 字段构建的请求进行关联。在绝大多数情况下，请求者允许队列管理器选择 MessageID 并期望作出响应的应用程序将此标识复制到响应的 CorrelID 中。在大多数情况下，请求者和作出响应的应用程序隐式知道正在使用的关联技术。有时候，作出响应的应用程序将允许在请求的报告字段中使用用来描述如何处理这些字段的各种标志。

可以使用下列选项来配置 WebSphere MQ 消息的导出绑定:

响应 MsgId 选项:

新建 `MsgID`
允许队列管理器为响应选择唯一的 MsgId（这是缺省情况）。

从请求 MsgID 中复制
从请求中的 MsgId 字段复制 MsgId 字段。

从 SCA 消息中复制
将 MsgId 设置为 SCA 响应消息的 WebSphere MQ 头中包含的标识；或者，如果不存在值，那么允许队列管理器定义新的 Id。

作为报告选项
根据如何处理 MsgId 来检查提示的请求中 MQMD 的报告字段。MQRO_NEW_MSG_ID 和 MQRO_PASS_MSG_ID 选项受支持，它们的行为分别类似于新建 MsgId 和从请求 MsgID 中复制。

响应 CorrelId 选项:

从请求 MsgID 中复制
从请求中的 MsgId 字段复制 CorrelId 字段（这是缺省情况）。

从请求 CorrelID 中复制
从请求中的 CorrelId 字段复制 CorrelId 字段。

从 SCA 消息中复制
将 CorrelId 设置为 SCA 响应消息的 WebSphere MQ 头中包含的标识；或者，如果不存在值，那么让它保留空白。

作为报告选项
根据如何处理 CorrelId 来检查提示的请求中 MQMD 的报告字段。MQRO_COPY_MSG_ID_TO_CORREL_ID 和 MQRO_PASS_CORREL_ID 选项受支持，它们的行为分别类似于从请求 MsgID 中复制和从请求 CorrelID 中复制。

可以使用下列选项来配置 WebSphere MQ 消息的导入绑定:

请求 MsgId 选项:

新建 MsgID
允许队列管理器为请求选择唯一的 MsgId（这是缺省情况）。

从 SCA 消息中复制
将 MsgId 设置为 SCA 响应消息的 WebSphere MQ 头中包含的标识；或者，如果不存在值，那么允许队列管理器定义新的 Id。

响应关联选项:

响应具有从 MsgId 复制的 CorrelID
期望响应消息根据请求的 MsgId 来设置 CorrelId 字段（这是缺省情况）。

响应具有从 MsgId 复制的 MsgID
期望响应消息根据请求的 MsgId 来设置 MsgId 字段。

响应具有从 CorrelId 复制的 CorrelID
期望响应消息根据请求的 CorrelId 来设置 CorrelId 字段。

Java EE 资源

将 WebSphere MQ 绑定部署到 Java EE 环境时，将创建许多 Java EE 资源。

参数

MQ 连接工厂
客户机用来与 WebSphere MQ 提供程序建立连接。

响应连接工厂

当发送目标与接收目标位于不同的队列管理器上时，SCA MQ 运行时将使用响应连接工厂。

激活规范

MQ JMS 激活规范与一个或多个消息驱动的 Bean 相关联，并提供使这些 Bean 接收消息所必需的配置。

目标

- 发送目标：这是指发送请求或外发消息的位置（导入）；如果未被入局消息中的 MQMD ReplyTo 头字段取代，那么发送目标是将发送响应消息的位置（导出）。
- 接收目标：这是指应该用来存放响应/请求或者入局消息的位置。

WebSphere MQ 头:

为了支持转换为服务组件体系结构 (SCA) 消息，WebSphere MQ 头结合了特定的约定。

WebSphere MQ 消息由系统头 (MQMD)、零个或零个以上的其他 MQ 头（系统头或定制头）以及消息体组成。如果消息包含多个消息头，那么这些头的顺序并不重要。

每个头都包含用于描述后一个头的结构的信息。第一个头由 MQMD 描述。

MQ 头的解析方式

“MQ 头”数据绑定用于解析 MQ 头。自动支持下列头:

- MQRFH
- MQRFH2
- MQCIH
- MQIIH

以 **MQH** 开始的头的处理方式有所不同。不会对头中的特定字段进行解析；它们将一直作为未解析的字节存在。

对于其他 MQ 头而言，您可以编写定制 MQ 头数据绑定以解析这些头。

MQ 头的访问方式

在产品中，可以通过两种方法来访问 MQ 头:

- 通过调解中的服务消息对象 (SMO)
- 通过 ContextService API

在内部，MQ 头由 SMO MQHeader 元素表示。MQHeader 是头数据的容器，它扩展了 MQControl，但包含类型为 anyType 的值元素。它包含 MQMD、MQControl（MQ 消息体控制信息）以及其他 MQ 头的列表。

- MQMD 表示 WebSphere MQ 消息描述的内容，但不包含用于确定主体结构和编码的信息。
- MQControl 包含用于确定消息体结构和编码的信息。
- MQHeaders 包含 MQHeader 对象列表。

MQ 头链是松散的，因此，在 SMO 中，每个 MQ 头都携带自己的控制信息（CCSID、编码和格式）。 您可以轻松方便地添加或删除这些头，而不必修改其他头数据。

设置 MQMD 中的字段

您可以使用 Context API 或通过调解中的服务消息对象 (SMO) 来更新 MQMD。下列字段将自动传播到出站 MQ 消息:

- Encoding
- CodedCharacterSet
- Format
- Report
- Expiry
- Feedback
- Priority
- Persistence
- CorrelId
- MsgFlags

请对导入或导出配置 MQ 绑定，以便将下列属性传播到出站 MQ 消息:

MsgID
将**请求消息标识**设置为从 SCA 消息复制。

MsgType
清除**将消息类型设置为 MQMT_DATAGRAM 或 MQMT_REQUEST 以表示请求-响应操作**复选框。

ReplyToQ
清除**覆盖对请求消息队列的应答**复选框。

ReplyToQMgr
清除**覆盖对请求消息队列的应答**复选框。

从 V7.0 开始，可以通过在 JNDI 目标定义中使用定制属性来覆盖上下文字段。请对发送目标设置值为 SET_IDENTITY_CONTEXT 的定制属性 MDCTX，以便将下列字段传播到出站 MQ 消息:

- UserIdentifier
- AppIdentityData

请对发送目标设置值为 SET_ALL_CONTEXT 的定制属性 MDCTX，以便将下列属性传播到出站 MQ 消息:

- UserIdentifier
- AppIdentityData
- PutApplType
- PutApplName
- ApplOriginData

某些字段不会传播到出站 MQ 消息。在发送消息期间，将覆盖下列字段:

- BackoutCount
- AccountingToken
- PutDate
- PutTime
- Offset
- OriginalLength

在 WebSphere MQ 绑定中以静态方式添加 MQCIH:

IBM Business Process Manager 支持以静态方式添加 MQCIH 头信息，而不必使用调解模块。

您可以通过多种方法对消息添加 MQCIH 头信息（例如，使用“头设置器”调解原语）。以静态方式添加此头信息（而不使用附加的调解模块）可能会有好处。您可以定义和添加静态头信息（包括 CICS® 程序名、事务标识和其他数据格式头详细信息）作为 WebSphere MQ 绑定的组成部分。

要以静态方式添加 MQCIH 头信息，必须配置 WebSphere MQ、MQ CICS Bridge 和 CICS。

您可以使用 Integration Designer 对 WebSphere MQ 导入配置 MQCIH 头信息所需的静态值。

当消息到达并且被 WebSphere MQ 导入所处理时，将进行检查以确定该消息中是否已存在 MQCIH 头信息。如果存在 MQCIH，那么将使用 WebSphere MQ 导入中定义的静态值来覆盖该消息中的相应动态值。如果不存在 MQCIH，那么将在该消息中创建 MQCIH 并添加 WebSphere MQ 导入中定义的静态值。

WebSphere MQ 导入中定义的静态值特定于方法。您可以在同一个 WebSphere MQ 导入中为不同方法指定不同的静态 MQCIH 值。

此功能并非用于在 MQCIH 未包含特定头信息时提供缺省值，这是因为，WebSphere MQ 导入中定义的静态值将覆盖入局消息中提供的相应值。

外部客户机:

IBM Business Process Manager 可以向使用 WebSphere MQ 绑定的外部客户机发送消息或接收来自这些客户机的消息。

外部客户机（例如 Web 门户网站或企业信息系统）可以通过导出向应用程序中的 SCA 组件发送消息，也可以被应用程序中的 SCA 组件通过导入进行调用。

WebSphere MQ 导出绑定部署消息驱动的 Bean (MDB) 以侦听传入到该导出绑定中指定的接收目标的请求。发送字段中指定的目标用于发送对入站请求的应答（如果所调用的应用程序提供应答）。因此，外部客户机能够通过导出绑定调用应用程序。

WebSphere MQ 导入与外部客户机绑定，并且可以向这些外部客户机传递消息。此消息可能需要来自外部客户机的响应，也可能不需要此响应。

您可以在 WebSphere MQ 信息中心中找到有关如何与使用 WebSphere MQ 的外部客户机交互的更多信息。

诊断 WebSphere MQ 绑定:

可诊断并修正 WebSphere MQ JMS 绑定发生的故障和失败情况。

主要失败情况

WebSphere MQ 绑定的主要失败情况由事务语义、WebSphere MQ 配置或对其他组件中的现有行为的引用来确定。主要失败情况包括:

- 无法连接至 WebSphere MQ 队列管理器或队列。

  无法连接至 WebSphere MQ 以接收消息将导致 MDB 侦听器端口无法启动。此情况将记录在 WebSphere Application Server 日志中。持久消息将保留在 WebSphere MQ 队列上，直到它们被成功检索（或由 WebSphere MQ 设为到期）。

  无法连接至 WebSphere MQ 以发送出站消息将导致控制发送的事务回滚。
- 无法解析入站消息或构造出站消息。

  数据绑定失败将导致控制该工作的事务回滚。

- 无法发送出站消息。

  无法发送消息将导致相关事务回滚。
- 多个响应消息或意外响应消息。

  对于每个请求消息，导入只应有一个响应消息。如果多个响应或延迟响应（SCA 响应已到期时响应到达）到达，那么系统将抛出服务运行时异常。事务将回滚，并且响应消息将退出队列或由失败事件管理器处理。

误用场景: 与 WebSphere MQ JMS 绑定比较

WebSphere MQ 导入和导出主要用来与本机 WebSphere MQ 应用程序交互并对调解展示 WebSphere MQ 消息体的完整内容。但是，WebSphere MQ JMS 绑定用来与针对 WebSphere MQ 部署的 JMS 应用程序互操作，它根据 JMS 消息模型展示消息。

以下场景应使用 WebSphere MQ JMS 绑定（而不是 WebSphere MQ 绑定）构建:

- 从 SCA 模块调用 JMS 消息驱动的 Bean (MDB)，其中 MDB 是针对 WebSphere MQ JMS 提供程序部署的。请使用 WebSphere MQ JMS 导入。
- 允许通过 JMS 从 Java EE 组件 Servlet 或 EJB 调用 SCA 模块。请使用 WebSphere MQ JMS 导出。
- 在 WebSphere MQ 中传输时调解 JMS MapMessage 的内容。将 WebSphere MQ JMS 导出和导入与相应数据绑定配合使用。

有时 WebSphere MQ 绑定和 WebSphere MQ JMS 绑定可能应该互操作。特别是，当您在 Java EE 与非 Java EE WebSphere MQ 应用程序之间进行桥接时，应将 WebSphere MQ 导出和 WebSphere MQ JMS 导入（或相反）与相应数据绑定和/或调解模块配合使用。

未传送的消息

如果 WebSphere MQ 无法将消息传送至其预期目标（例如，因为配置错误），那么它会将这些消息传送至名义上的死信队列。

这样做时，它会在消息体的开头添加死信头。此头包含故障原因、原始目标和其他信息。

基于 MQ 的 SCA 消息未出现在失败事件管理器中

如果 SCA 消息是因为 WebSphere MQ 交互失败而产生的，那么您应在失败事件管理器中查找这些消息。如果这些消息未出现在失败事件管理器中，请检查底层 WebSphere MQ 目标的最大失败传送次数值大于 1。如果将此值设置为 2 或更高，那么意味着系统允许在 WebSphere MQ 绑定的 SCA 调用期间与失败事件管理器交互。

对错误的队列管理器重放了 MQ 失败事件

要对出站连接使用预定义连接工厂时，连接属性必须与用于入站连接的激活规范中定义的连接属性相匹配。

预定义连接工厂用于在重放失败事件时创建连接，并且必须因此配置为使用最初接收到该消息的队列管理器。

处理异常:

绑定的配置方式确定处理数据处理程序或数据绑定所发出的异常的方式。另外，调解流的性质规定了抛出这种异常时系统的行为。

在绑定调用数据处理程序或数据绑定时，可能会发生各种问题。例如，数据处理程序可能会接收到有效内容已损坏的消息，或者它可能尝试读取格式不正确的消息。

绑定处理这种异常的方式由您实现数据处理程序或数据绑定的方式确定。建议的行为是，您将数据绑定设置为抛出 **DataBindingException**。

对于数据处理程序，情况类似。由于数据处理程序被数据绑定调用，因此任何数据处理程序异常都将包装到数据绑定异常中。因此，**DataHandlerException** 将作为 **DataBindingException** 向您报告。

抛出任何运行时异常（包括 **DataBindingException** 异常）时:

- 如果调解流配置为事务性的，那么缺省情况下 JMS 消息将存储在失败事件管理器中以便手动回放或删除。

  注: 您可以更改绑定中的恢复方式，以便回滚消息，而不是将消息存储在失败事件管理器中。
- 如果调解流不是事务性的，那么将会记录该异常并且该消息将丢失。

对于数据处理程序，情况类似。由于数据处理程序被数据绑定调用，因此数据处理程序异常将生成到数据绑定异常内。因此，**DataHandlerException** 将作为 **DataBindingException** 向您报告。

绑定的限制

此处列示了绑定在使用方面存在的一些限制。

MQ 绑定的限制:

此处列示了 MQ 绑定在使用方面存在的一些限制。

不支持发布/预订消息分发

虽然 WMQ 本身支持发布/预订，但 MQ 绑定当前不支持用于分发消息的发布/预订方法。但是，MQ JMS 绑定支持这种分发方法。

共享接收队列

多个 WebSphere MQ 导出和导入绑定希望在其已配置的接收队列中出现的任何消息都针对该导出或导入。配置导入和导出绑定时应了解下列注意事项:

- 每个 MQ 导入都必须具有不同的接收队列，这是因为 MQ 导入绑定将接收队列中的所有消息都视为对它所发送的请求的响应。如果接收队列由多个导入共享，那么响应可能被错误的导入接收，并且将无法与原始请求消息相关。
- 每个 MQ 导出都应该具有不同的接收队列，这是因为否则您将无法预测哪个导出将获取任何特定请求消息。
- MQ 导入和导出可以指向同一发送队列。

JMS、MQ JMS 和通用 JMS 绑定的限制:

JMS 和 MQ JMS 绑定有一些限制。

生成缺省绑定的含意

下列各部分讨论关于使用 JMS、MQ JMS 和通用 JMS 绑定的限制:

- 生成缺省绑定的含意
- 响应相关方案
- 双向支持

生成绑定时，系统将为您填写几个字段作为缺省值（如果您未选择自己输入这些值）。例如，将为您创建连接工厂名称。如果您知道要将应用程序放在服务器上并且将通过客户机远程访问此应用程序，那么您应该在创建绑定时输入 JNDI 名称而不是使用缺省值，这是因为您可能要在运行时通过管理控制台控制这些值。

但是，如果您接受缺省值并在随后发现您无法通过远程客户机访问该应用程序，那么可以使用管理控制台来显式设置连接工厂值。请在连接工厂设置中找到提供程序端点字段，然后添加 <server_hostname>:7276 之类的值（如果您使用的是缺省端口号）。

响应相关方案

如果您使用用于使请求-响应操作中的消息相关的“相关标识到相关标识”响应相关方案，那么消息中必须具有动态相关标识。

要使用调解流编辑器在调解模块中创建动态相关标识，请在具有 JMS 绑定的导入之前添加一个 XSLT 节点。打开 XSLT 映射编辑器。已知的服务组件体系结构头将在目标消息中可用。将源消息中包含唯一标识的字段拖放到目标消息的 JMS 头中的相关标识。

双向支持

运行时仅支持对 Java 命名和目录接口 (JNDI) 名称使用 ASCII 字符。

共享接收队列

多个导出和导入绑定希望在其已配置的接收队列中出现的任何消息都针对该导出或导入。配置导入和导出绑定时应了解下列注意事项:

- 每个导入绑定都必须具有不同的接收队列，这是因为导入绑定将接收队列中的所有消息都视为对它所发送的请求的响应。如果接收队列由多个导入共享，那么响应可能被错误的导入接收，并且将无法与原始请求消息相关。
- 每个导出都应该具有不同的接收队列，这是因为否则您将无法预测哪个导出将获取任何特定请求消息。
- 导入和导出可以指向同一发送队列。

业务对象

计算机软件行业已经开发了多种编程模型和框架，在这些编程模型和框架中，*业务对象*提供业务数据的自然呈现方式供应用程序处理。

通常，这些业务对象:

- 使用行业标准进行定义
- 透明地将数据映射到数据库表或企业信息系统
- 支持远程调用协议
- 为应用程序编程提供数据编程模型基础

从工具角度而言，Integration Designer 为开发者提供了一个这样的公共业务对象模型，用于表示不同的域中不同类型的业务实体。在开发期间，此模型使开发者能够将业务对象定义为 XML 模式定义。

在运行时，XML 模式定义所定义的业务数据表示为 Java 业务对象。在此模型中，业务对象松散地基于服务数据对象 (SDO) 规范的早期草稿，并提供了处理业务数据所需的全套编程模型应用程序接口。

定义业务对象

可使用 Integration Designer 中的业务对象编辑器来定义业务对象。业务对象编辑器将业务对象作为 XML 模式定义来存储。

使用 XML 模式来定义业务对象具有以下几大好处:

- XML 模式提供了基于标志的数据定义模型，并且是截然不同的各种各样系统与应用程序之间实现互操作性的基础。XML 模式与 Web 服务描述语言 (WSDL) 一起用来提供组件、应用程序和系统之间的基于标准的接口合同。
- XML 模式定义一种富数据定义模型来表示业务数据。除了其他功能部件以外，此模型包括复杂类型、简单类型、用户定义的类型、类型继承和基数。
- 可以通过使用 Web 服务描述语言定义的业务接口和数据以及由符合业界标准的组织或者其他系统和应用程序提供的 XML 模式来定义业务对象。Integration Designer 可以直接导入这些业务对象。

Integration Designer 还支持在数据库和企业信息系统中发现业务数据，然后生成该业务数据的基于标准的 XML 模式业务对象定义。按此方式生成的业务对象通常称为*特定于应用程序的业务对象*，这是因为它们模拟企业信息系统中定义的业务数据的结构。

当流程处理来自许多不同的信息系统的数据时，将截然不同的业务数据表示（例如，CustomerEIS1 和 CustomerEIS2，或者是 OrderEIS1 和 OrderEIS2）变换为单个规范化表示（例如，Customer 或 Order）会很有用。规范化表示通常称为*通用业务对象*。

业务对象定义（尤其是通用业务对象的业务对象定义）被多个应用程序频繁使用。为了支持此复用操作，Integration Designer 允许在库中创建业务对象，然后将这些业务对象与多个应用程序模块关联。

服务组件体系结构 (SCA) 应用程序模块所提供和使用的服务的合同，以及用来创建应用程序模块中的组件的合同，都是使用 Web 服务描述语言定义的。WSDL 可以表示合同的操作和业务对象（XML 模式定义它们来表示业务数据）。

使用业务对象

服务组件体系结构 (SCA) 提供了用于定义应用程序模块的框架、应用程序模板提供的服务、应用程序模块使用的服务以及用于提供应用程序模块的业务逻辑的组件的组合。业务对象在应用程序中扮演重要角色，它将定义用来描述服务合同和组件合同的业务数据以及组件将处理的业务数据。

下图描述了 SCA 应用程序模块，并且说明了开发者将在其中使用业务对象的许多位置。

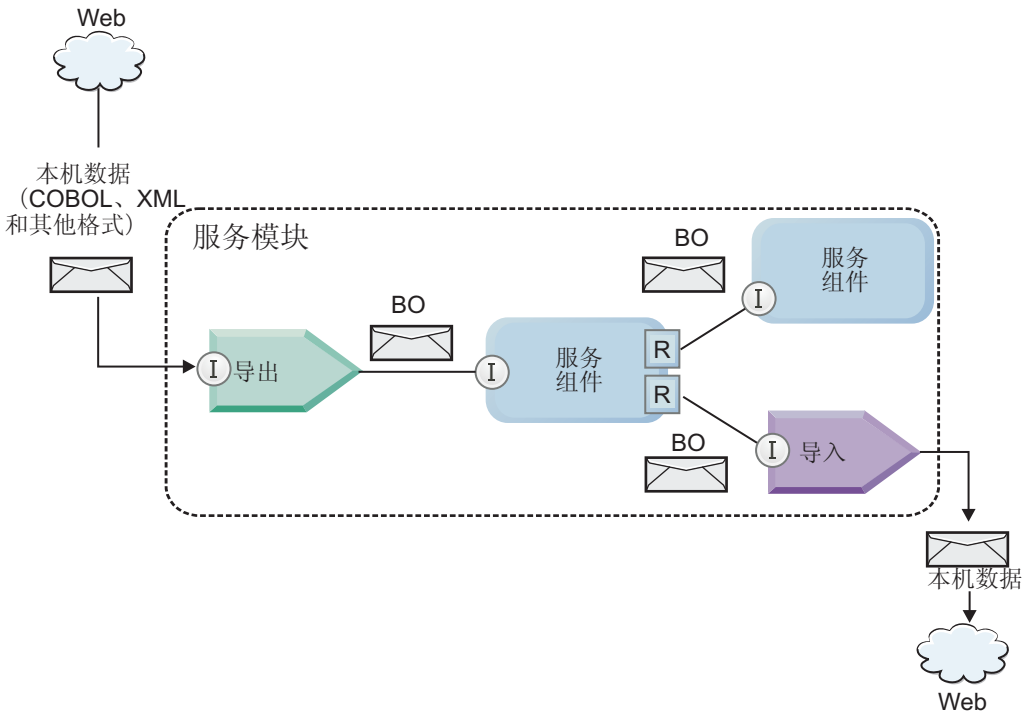


图 40. 业务对象表示在应用程序中的各项服务之间流动的数据

注: 本主题描述 SCA 应用程序模块如何使用业务对象。如果您使用的是 Java 接口，那么 SCA 应用程序模块还可以处理 Java 对象。

业务对象编程模型

业务对象编程模型由一组用于表示下列对象的 Java 接口组成:

- 业务对象定义和实例数据
- 支持对业务对象执行操作的一组服务

业务对象类型定义由 commonj.sdo.Type 和 commonj.sdo.Property 接口表示。业务对象编程模型提供了有关将 XML 模式复杂类型信息映射至类型接口以及将复杂类型定义中的每个元素映射至属性接口的一组规则。

业务对象实例由 commonj.sdo.DataObject 接口表示。业务对象编程模型具有隐式类型，这意味着可以使用同一 commonj.sdo.DataObject 接口来表示不同的业务对象定义（例如，Customer 和 Order）。可以从每个业务对象来设置和检索哪些属性的定义，由每个业务对象的相关联 XML 模式中定义的类型信息来确定。

业务对象编程模型的行为基于服务数据对象 2.1 规范。有关更多信息，请参阅 Web 上提供的 SDO 2.1 for Java 规范、教程和 Javadoc: http://osoa.org/display/Main/Service+Data+Objects+Specifications。

业务对象服务支持对业务对象执行各种生命周期操作（例如，创建、解析和序列化业务对象以及确定业务对象是否等价）。

有关业务对象编程模型的具体细节，请参阅使用业务对象服务进行编程和 com.ibm.websphere.bo 包。

绑定、数据绑定和数据处理程序

如图 40 中所示，用来调用由 SCA 应用程序模块提供的服务的业务数据变换为业务对象，以便 SCA 组件可以处理业务数据。同样，由 SCA 组件处理的业务对象转换为外部服务所需要的数据格式。

在某些情况下（例如，Web Service 绑定），用来导出和导入服务的绑定将数据自动变换为相应的格式。在其他情况下（例如，JMS 绑定），开发者可以提供数据绑定或数据处理程序，用于将非本机格式转换为由 DataObject 接口表示的业务对象。

有关开发数据绑定和数据处理程序的更多信息，请参阅第 37 页的『数据处理程序』和第 38 页的『数据绑定』。

组件

SCA 组件将 Web 服务描述语言和 XML 模式组合使用来定义它们的提供服务合同和使用服务合同。使用 DataObject 接口将 SCA 在组件之间传递的业务数据表示为业务对象。SCA 将验证这些业务对象类型与要调用的组件所定义的接口合同是否兼容。

组件不同，用于处理业务对象的编程模型抽象也会不同。POJO 组件和调解流组件定制原语通过使用业务对象编程接口和服务直接启用 Java 编程来直接处理业务对象。大多数组件针对处理业务对象提供了更高级别的抽象，但是还提供了 Java 代码段以定义业务对象接口和服务中的定制行为。

可以使用接口流调解与“业务对象映射”组件的组合来变换业务对象，也可以使用调解流组件及其 XML 映射原语来变换业务对象。对于特定于应用程序的业务对象与通用业务对象之间的互相转换，这些业务对象变换功能很有用。

特殊业务对象

服务消息对象和业务图是用于特定用途的两种特殊类型的业务对象。

服务消息对象

服务消息对象 (SMO) 是一种特殊的业务对象，调解流组件使用它来表示与服务调用相关联的数据集合。

SMO 具有一种由头、上下文、主体和附件（如果存在）组成的固定顶层结构。

- 头通过特定协议或绑定来携带与服务调用相关的信息。SOAP 头和 JMS 头就是头的两个示例。
- 在调解流组件处理上下文数据时，上下文数据携带了与调用相关联的其他逻辑信息。此信息通常不是客户机发送或接收的应用程序数据的一部分。
- SMO 的主体携带了有效内容业务数据，它以标准业务对象的形式来表示核心应用程序消息或调用数据。

SMO 还可以使用带有附件的 SOAP 来携带 Web Service 的附件数据。

调解流执行诸如路由请求和数据变换之类的任务，而 SMO 以单个统一结构提供了头和有效内容的组合视图。

业务图

业务图是一种特殊的业务对象，用来支持集成场景中的数据同步。

举个示例来说明：假定两个企业信息系统都表示了某个特定订单。当一个系统中更改了此订单时，就可以向另一个系统发送一条消息以使订单数据同步。业务图支持这样一种概念：仅将订单中已更改的部分发送至另一个系统，并使用更改摘要信息对所作的更改加以注释，以便定义更改类型。

在此示例中，“订单”业务图将对另一个系统传达以下信息：已删除此订单的其中一个排列项，并且已更新此订单的“计划发货日期”属性。

在 Integration Designer 中，很容易将业务图添加至现有业务对象。业务图常用于使用了 WebSphere Adapters 以及支持迁移 WebSphere InterChange Server 应用程序的场景。

业务对象解析方式

Integration Designer 提供了有关模块和库的属性，可用于将业务对象的 XML 解析方式配置为积极或惰性。

- 如果选项设置为*积极*，那么系统会积极解析 XML 字节流以创建业务对象。
- 如果选项设置为*惰性*，那么系统会正常创建业务对象，但 XML 字节流的实际解析被延迟，仅在访问业务对象属性时部分解析。

无论在哪一种 XML 解析方式下，都将始终预先解析非 XML 数据以创建业务对象。

选择业务对象解析方式时的注意事项

业务对象解析方式确定 XML 数据在运行时的解析方式。在创建模块或库时，可以对其定义业务对象解析方式。您可以更改模块或库的解析方式，但您应该了解其影响。

业务对象运行时框架的版本是在模块和库级别设置的。在 V7 之前的 IBM Integration Designer 版本中创建的模块将以预先解析方式运行，而不必进行任何更改。缺省情况下，在 IBM Integration Designer V7 和更高版本中创建的模块和库将根据众多因素被赋予最合适的解析方式，这些因素包括工作空间中现有项目的解析方式或者同一解决方案中的所依赖项目或其他项目的解析方式等等。您可以更改模块或库的业务对象解析方式以使其与您的实现相符，但您应该了解下列注意事项。

注意事项

- 延迟业务对象解析方式处理 XML 数据的速度较快；但是，在更改模块或库的配置之前，您应该了解预先方式与延迟方式之间的兼容性差别。这些差别将影响模块的运行时行为。有关哪种解析方式最适合于您的应用程序的信息，请参阅 Benefits of using lazy versus eager parsing mode。
- 一个模块只能配置为以一种解析方式运行。您可以配置库，以便支持任意一种解析方式或者同时支持这两种解析方式。配置为同时支持这两种解析方式的库既可以由使用预先解析方式的模块引用，也可以由使用延迟解析方式的模式引用。库在运行时的解析方式由引用该库的模块确定。在运行时，模块声明其解析方式，该解析方式将由该模块以及该模块所使用的任何库使用。
- 在下列情况下，配置了不同解析方式的模块和库兼容:
  - 配置了延迟解析方式的模块和库与使用延迟解析方式或者同时使用预先和延迟解析方式的库兼容。
  - 配置了预先解析方式的模块和库与使用预先解析方式或者同时使用预先和延迟解析方式的库兼容。
  - 配置了延迟和预先解析方式的库只与同时使用延迟和预先解析方式的库兼容。
- 对于使用 SCA 绑定进行通信的交互模块，请使用同一个框架。如果这些模块使用不同的解析方式进行通信，那么可能会引起性能问题。

使用惰性解析方式和积极解析方式的优点

某些应用程序受益于惰性 XML 解析方式，其他应用程序因为使用积极解析方式而改进了性能。建议同时将应用程序置于两种解析方式下进行测试，以确定哪种方式最适合您的应用程序的特定特征。

下节提供有关受益于每种解析方式的应用程序类型的一般指南:

- 受益于惰性 XML 解析方式的应用程序

  解析大型 XML 数据流的应用程序在使用惰性 XML 解析方式时可能会见到性能改进。XML 字节流的大小增加并且应用程序从字节流访问的数据量减少时，性能改进幅度增加。

  注: 业务对象延迟解析方式在 WebSphere Process Server V7.0.0.3 及更高版本中和 IBM Process Server 中受支持。包括调解流组件的模块和调解模块不受支持。
- 受益于积极解析方式的应用程序

以下应用程序在积极解析方式下可能会获得更好的性能:

– 解析非 XML 数据流的应用程序
– 使用 BOFactory 服务创建的应用程序
– 解析极小 XML 消息的应用程序

应用程序迁移和开发注意事项

如果您正在将最初使用预先解析方式开发的应用程序配置为使用延迟解析方式，或者您打算将应用程序在延迟解析方式与预先解析方式之间进行切换，那么您一定要知道这两种方式之间的差别，以及在这两种方式之间进行切换时的注意事项。

错误处理

如果所解析的 XML 字节流的格式错误，那么将发生解析异常。

- 在 XML 预先解析方式下，一旦从入站 XML 流解析业务对象，就会发生这些异常。
- 如果配置了延迟 XML 解析方式，那么会在访问业务对象属性以及解析格式错误的 XML 部分时潜在地发生解析异常。

要处理格式错误的 XML，请选择下列其中一个选项:

- 在边缘部署企业服务总线，以验证入站 XML
- 在访问业务对象属性的位置编写延迟错误检测逻辑

异常堆栈和消息

因为预先 XML 解析方式与延迟 XML 解析方式具有不同的底层实现，所以虽然业务对象变成接口和服务抛出的堆栈跟踪具有相同的异常类名，但是它们可能不包含相同的异常消息或者封装的一组特定于实现的异常类。

XML 序列化格式

采用延迟 XML 解析方式时可优化性能；进行序列化时，它会尝试将入站字节流中未修改的 XML 复制到出站字节流中。这样会提高性能，但是，如果整个业务对象是以延迟 XML 解析方式进行更新的，或者它先前是采用预先 XML 解析方式运行的，那么出站 XML 字节流的序列化格式可能会不同。

尽管 XML 序列化格式可能在语法上不完全等同，但是，无论采用哪种解析方式，业务对象所提供的语义值都是等同的；在采用不同解析方式运行、但语义等同的应用程序之间可以安全地传递 XML。

业务对象实例验证器

延迟 XML 解析业务对象方式实例验证器对业务对象提供了更高精确度的验证，尤其是对属性值进行构面验证。正是由于进行了这些改进，因此延迟解析方式实例验证器将捕获到采用预先解析方式时未捕获到的其他问题，并且提供更详细的错误消息。

V602 XML 映射

最初在 WebSphere Integration Developer V6.1 之前的版本中开发的调解流可能包含一些 XSLT 原语，这些 XSLT 原语使用了无法直接在延迟 XML 解析方式下执行的映射或样式表。当已将应用程序迁移为用于延迟 XML 解析方式时，迁移向导可以将与 XSLT 原语相关联的映射文件自动更新为采用新的解析方式运行。但是，如果 XSLT 原语直接引用一个已手动编辑的样式表，那么不会迁移此样式表，并且不会以延迟 XML 解析方式执行此样式表。

未发布的专用 API

如果应用程序正在利用未发布且特定于实现的专用业务对象编程接口，那么在切换解析方式时，编译此应用程序可能会失败。在预先解析方式下，这些专用接口通常是由 Eclipse 建模框架 (EMF) 定义的业务对象实现类。

在所有情况下，强烈建议您除去应用程序中的专用 API。

服务消息对象 EMF API

IBM Integration Designer 中的调解组件能够使用 com.ibm.websphere.sibx.smobo 包中提供的 Java 类和接口来处理消息内容。在延迟 XML 解析方式下，仍然可以使用 com.ibm.websphere.sibx.smobo 包中的 Java 接口，但是，直接引用 Eclipse 建模框架 (EMF) 类和接口的方法或者从 EMF 接口继承的方法可能会失败。

在延迟 XML 解析方式下，ServiceMessageObject 及其内容无法强制类型转换为 EMF 对象。

BOMode 服务

BOMode 服务用来确定当前正在执行的 XML 解析方式是预先解析方式还是延迟解析方式。

迁移

V7.0.0.0 之前的所有应用程序都采用预先 XML 解析方式运行。在运行时使用 BPM 运行时迁移工具对它们进行迁移时，它们将继续采用预先 XML 解析方式运行。

为了能够将版本低于 V7.0.0.0 的应用程序配置为采用延迟 XML 解析方式，请首先使用 Integration Designer 来迁移该应用程序的工件。迁移之后，再将该应用程序配置为采用延迟 XML 解析。

请参阅迁移源工件，以了解有关在 Integration Designer 中迁移工件的信息；另请参阅配置模块和库的业务对象解析方式以了解有关设置解析方式的信息。