

管理部署

本页主题

- ▼ 执行摘要
- ▼ 文件简介
- ▼ 部署处理
- ▼ 角色
- ▼ 专案领域
- ▼ 结构
- ▼ 摘要
- ▼ 附录 A 填ML 档案
- ▼ 附录 B 殊SI 套件
- ▼ 附录 C 紐客户执行
- ▼ 参照

Microsoft 企业服务白皮书 电子商务技术准备

附注 在有关将 [Microsoft® 企业服务] 框架套用到电子商务解决方案的一系列白皮书中，此白皮书是其中一本。[电子商务白皮书系列](#)包含此系列中所有文章的完整清单 (包括描述)。

致谢

作者: Quoc Bui

执行摘要

分布式应用程序的管理部署通常是一种事后的想法，作为软件开发的最后阶段之一。一般而言，技术结构假定部署责任以确定已正确放置所有分布式应用程序组件。其范围从档案部署、登录修改及 Web 网站设定延伸至中间层数据库整合。

该计划管理部署策略减轻了对技术结构工程师的要求，他需确定该过程已正确执行，对于仅具备有限技术知识的人，仍可采取相同的责任。此位置是指组建设定管理员 (BCM)。

幸运的是，Microsoft 的新技术，如 Microsoft® Windows® Installer 使得开发更易处理，并使之成为本文所描述之管理部署策略的中心，它假定已对 Windows Installer 有所了解。如需详细信息，请参阅参照区段。

Windows Installer 的主要好处在于：

- 安装程序组建于 Windows 2000 之内。
- 没有独立的 setup.exe 档案。
- 具有一致的安装规则。
- 亦适用于 Windows 95/98 及 Windows NT® 4.0。
- 每个安装工作阶段为一异动系统，该系统在发生错误时可以返回到先前状态。
- 它被当作 Windows NT/2000 的服务执行。
- 已授权的应用程序可使用高级权限安装。

假定该应用程序具有 Microsoft Transaction Server (MTS)/COM+ 套件、COM 组件、ASP/XML/text 档案及 SQL 数据库，该目标为期待有效部署「Windows 分布式 InterNet 应用程序结构 (DNA)」的解决方案。(附注 MTS 整合到 Windows 2000 操作系统的 COM+ 程序模型中。) 此解决方案包含下列主题：

部署策略。任何一种技术都必须使用某种策略来解决问题。部署策略是指将一组程序代码从一个分布式环境移动到另一个分布式环境的过程。计划的策略为将 WinDNA 结构部署为 Windows 2000 及 Windows NT 4.0 服务器平台上的单一项目。

MSI 套件建立的自动控制。计划的工具将自动完成准备、部署、安装 Microsoft Installer (MSI) 套件的建立过程。MSI 套件包含 Windows Installer 组件,其中含有 Windows DNA 应用程序。套件工具简化了 Windows Installer 的应用程序设计界面 (API)。MSI 套件将在指定的位置布署/删除档案,登录/取消登录动态链接库 (DLL),登录/取消登录 MTS/COM+ 套件,建立/移除 Internet Information Services (IIS) 虚拟目录,执行「结构查询语言 (SQL)」查询及执行指令行的操作。

此外,安装后组件,即为「Windows Installer 自订 (WICA)」,包含于套件工具中。WICA 控制非 Windows Installer 本身具有的执行,对于完成安装处理至关重要。

名单, XML 档案。XML 档案格式的名单,对套件工具开放指示以自动完成 MSI 套件的建立。该名单是抽象出来的额外层,来源控制间的连结,如 Visual SourceSafe®,及套件工具。该名单包含一组关于档案部署位置的指示和 MTS 套件信息,以及设定 IIS 虚拟目录的指导和 SQL 查询执行的内容。与 XML 例项所要求的条件一致,该名单将遵循由已定义之 XML 架构所定义的指导,并具有良好的结构。

因为 XML 灵活、自身描述及便携的特性,使其成为名单的理想选择。

此外,该名单还将包含安装后指示,作为 WICA 控制非 Windows Installer 本身具有之执行的指导。非本身具有的执行为 IIS 虚拟 Web 目录安装程序,MTS 套件在 NT 4.0 服务器上的登录,SQL 指令文件及查询的执行方式及指令行执行。

远程部署及安装。远程部署可以下列形式存在 1) 透过 HTTP 传输开放 MSI 套件,和/或 2) 复制到指定网络共享的档案。远程安装藉由简化 Windows 管理仪器设备实现。

「Microsoft Windows 管理仪器设备 (WMI)」与「桌面管理工作小组 (DMTF) 的 Web 型企业管理 (WBEM)」标准兼容,支持基于「公用信息模型 (CIM)」的一致系统及应用程序管理。作为 Microsoft Windows 管理服务的组件,WMI 协助减少 Windows NT 型企业网络管理组件的维护及成本。

Microsoft System Management Server (SMS) 及 Application Center 2000 Server 亦可远程安装套件。

文件简介



部署是指开始于组建设定负责检查来源程序代码是否不在控制系统版本范围之内之管理员 (BCM) 的过程, 结束于 NLB/WLBS MSCS 丛集及执行 IIS、SQL Server 7.0、MTS/COM+ 及多重实体环境下的 Windows NT 4.0/Windows 2000 Servers (如系统整合测试及产品) 之机器中应用程序的安装。应用程序包含 IIS metabase 及 MTS/COM+ 类别目录设定、登录数据项、安全性设定、ASP 档案、SQL 指令文件、不同的档案类型、及需求分析时判定的其它组件。

项目阶段应包含解决方案结构定义、结构确认 (藉由原型化参照执行方式)、详细需求分析、设计、及组建。「Microsoft 解决方案框架」作为理想的项目模型, 指导使用者从初学到成功的掌握对产品使用。

本文建议采取带有执行该提议之技术推荐的部署策略。技术推荐的形式为简化数个技术的单一工具。它开始于 XML 档案的消耗, 代表一组对自动操作不言自明的指示, 结束于 MSI 套件的建立。

而该工具可能包含远程安装 MSI 套件所需之设备, 它并不具备在 Web 建地中安装或拆卸机器的功能。若不使用 Application Center 2000 Server, 将需要对丛集机器进行手动操作。

目前套件及部署技术

目前, 仅有少数类型的套件及部署软件启用了将程序代码从开发环境发布到产品的功能。现存的套件及部署技术, 有其局限性且无法简化最近的进阶作业, 如 Microsoft Windows Installer。虽然, 有些可作为优秀的桌面型应用程序, 但却不能很好地作为 Windows DNA 环境专用之包装及分配应用程序的代用品。

技术推荐

幸运的是, 目前已生产数个技术进阶产品, 并且其种类将与日俱增。软件发布最大的进步为与 Windows 2000 同时出现的 Windows Installer 引擎, 它在 Windows NT 及 Windows 9x 型操作系统中亦可使用。

「Microsoft 系统管理服务器 (SMS)」是优秀的软件发布系统, 但其不为组装导向。直到 Application Center 2000 Server 上市, 才有了综合封装及分送启用器。否则, 此文件的建议封装及部署策略将紧密对齐并补强 Application Center 2000 Server。

高级策略将充分发挥 Windows Installer 的效能。Windows installer 为 Windows 2000 的一部份，Zero Administration Windows (ZAW) 努力降低开发、使用及管理桌面计算机的整体成本。目前，在缺少现存之充分发挥 Windows Installer 技术以部署 Windows DNA 应用程序的封装工具的状况下，为保持此基本原理，建议使用自订封装工具。此工具将建立综合的 Windows Installer 套件，它将不仅在 Windows 2000 操作系统，而且在 Windows NT 4.0 Server 及 Windows 9x 系统成功而充分地完成部署。自订封装工具亦将确定一致且可重复的分布式应用程序封装程序。

自订封装工具亦包含安装后组件，如 WICA。封装工具将所有所需之档案及指示组，如登录修改及 ODBC 设定，封装在一起。Windows Installer 自订执行 (WICA) 是处理非 Windows 自身安装程序的安装后组件。当执行了所有的自身动作，Windows Installer 将 WICA 例示为自订执行组件。

安装后动作包含 Windows NT 4.0 Server (Windows 2000 的 COM+) 的 MTS 套件登录、IIS 虚拟 Web 设定、SQL 查询及支持并组装/修改数据库的指令文件执行方式、及指令行执行，如初始化 Windows Script Host 档案。

分布式系统

理想的分布式环境包含：开发、系统整合测试及生产环境三种环境。开发人员使用开发环境进行模块及整合测试。为了初步确保测试的顺利进行，他们检查源程序代码存放库 (如 Visual SourceSafe) 中的程序代码。目前实际处理的范例包含一检查只读权限的组建小组，并按需要的方式编译二进制程序代码。组建小组继续将分布式程序代码部署到分布式测试服务器，即「系统整合测试」环境。为了成功的完成测试，组建小组将分布式程序代码部署到生产环境，该环境对外部操作开放。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 1 理想化分布式环境的示意图

在此分布式系统中，作为部署策略的一部份，此文件期待推荐部署程序的封装问题。封装分布式程序代码为以一致性及控制性方式管理部署的中心。已封装程序代码的好处包含藉由记录显示部署及安装内容，执行版本控制。理想的分布式系统包含分布式网络管理系统服务器，如 SMS 或 Application Center 2000 Server，它可能会促进已封装 MSI 容器的部署及安装。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 2 理想化分布式环境的封装

「Microsoft 系统管理服务器 (SMS)」可用于部署及初始化 MSI 套件的远程部署。但自动化部署及使用 Microsoft Application Center 2000 Server 安装这些套件可带来更大的益处。Application Center 2000 Server 将 Windows Installer 特定的用于远程软件发布及安装。利用为开发及远程安装设计的预先组建 MSI 套件，Application Center 2000 Server 可以独立及排定为基础，用于自动化 Web 服务器的拆卸及安装。

部署处理程序

封装处理程序开始于来源控制存放库，并依序到达名单建立 (由一组自我描述部署指示组成的 XML 例项)，读取该名单的封装工具，然后产生部署及安装就绪的 MSI 套件。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 3 封装程序概观

封装工具建立一 Shell MSI 套件，消耗 (读取) 名单，开始以与安装指示相关的信息填入 MSI 套件。该工具然后将夺取所有名单中列出的档案，并将其精简为将装入 MSI 套件的压缩 CAB 档。该名单亦将包含于此 MSI 套件中用于安装后指示。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 4 使用封装工具建立 MSI 套件。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 5 封装程序

开发人员使用开发环境测试其程序代码，然后将其来源存放至来源程序代码存放库区域。BCM 编译二进制程序代码，并将位封装至 MSI 套件。然后这些套件部署至各自的系统整合测试 (SIT) 及名单中定义之档案结构中的生产环境。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 6 利用 MSI 部署的理想化环境

将所有档案部署至其适当的目的地，并且执行了所有 Windows Installer 自身操作之后，将会安装「安装后 Windows Installer 自订操作 (WICA)」组件并执行非自身 Windows Installer 操作。这些非自身操作可包含 IIS 虚拟 Web 目录安装程序、NT 4.0 Server、SQL 指令文件及查询执行方式的 MTS 套件登录、及指令行执行。

Windows 2000 系统之 Windows Installer 本身拥有 COM+ 套件登录。为 NT 4.0 服务器设计之 MTS 套件的 WICA 安装程序执行方式亦可以严密地执行在 Windows 2000 上。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 7 自动化公布的安装处理程序

角色



开发人员

开发人员根据名单和/或 BCM 的执行方式指示，以结构格式化提供关于如何组建产品，如 DLL 的指示。开发人员将协助 BCM 产生关于 MTS/COM+ 组件、数据库指令文件建立、及任何成功安装所必需的名单信息。此外，开发人员提供任何依存盘案如 .OCX、.INI、及部署产品所需之登录数据项和协力厂商 (如果存在) 组件。

组建设定管理员

BCM 对封装、部署、及自动化 MSI 套件建立至关重要。此非开发人员形式可使开发人员轻松地进行程序代码开发。此人了解应用程序如何完整地工作，及各部份结合的位置。一般职责包含编译码、确认名单及封装及部署应用程序。此人将特别负责：

- 名单 (XML 档案) 的建立及维护。(可使用文字或 XML 编辑器手动地的建立名单，或程序化的使用来源控制的 API 自动化建立。)
- 定义组建组件的群组 (Windows Installer 组件)
- 组建程序
- 部署/安装程序
- 迁移程序

专案领域

项目领域定义及条件对所有项目都至关重要。本文建议使用 Visual Basic 及 Windows Installer 组建自订封装工具。无论工具组的种类为何，部署策略应保持一致。

领域

领域定义为：

建议的工具组是指此后提及的 WinDNA 封装程序。相关的文件提供将用于产生分散应用程序之 Windows Installer 套件的程序及一组可多次使用的组件。这些将包含提供 MTS/COM+ 的 DLL、Windows Installer 的 IIS 及 SQL 自订操作功能、说明程序 WSH 指令文件、名单架构及架构存取程序代码、及描述该工具组使用方法的文件。

领域包含：

- MSI 套件的自动化建立
- 名单的 XML-架构。

工具 (WinDNA 封装程序)

该工具的结构将帮助 Windows Installer API 读取名单及 XML 档案窗体中的数据，并建立安装就绪的 MSI 套件。MSI 套件将由包含文本文件 (例如，ASP、XML、HTML) 的 Windows Installer 组件、DLL、EXE、MTS/COM+ DLL、支持档案、及该名单本身组成。工具组或部署装备及相关的文件提供将用于产生应用程序之 Windows Installer 套件的程序及一组可多次使用的组件及指令文件。

资源

BCM 将负责产生及维护名单，即 XML 档案，它是来源控制存放库 (如 Visual SourceSafe) 与 WinDNA 封装程序之间的媒体。此人亦将执行来源控制中来源程序代码的只读检验、编译二进制代码、对部署工具开放二进制代码及支持档案、维护名单、并使用 WinDNA 套件建立 MSI 套件。

Patches/Hotfixes/Includes

MSI 套件可包含单一 Windows Installer 组件；因此，将产生更流行的 MSI 套件取代现存的目标组件。BCM 可能选择性的包含合并模块 (将属于 MSI 套件) 中的二进制代码，如最新的 ADO 位。

名单

名单作为 Windows Installer 技术的摘要的中间阶层。BCM 需要了解 Windows Installer 的工作方法、或 MSI 数据库的建立方法。BCM 几乎无需了解 Windows DNA 应用程序对成功部署的要求。

建立名单有两种建议方法。第一种为藉由程序化的提高来源控制 (例如, Visual SourceSafe) 对象模型及 Microsoft XML DOM, 自动化建立例项。此方为两种方法中较可靠的一种, 但它需要初始程序投资。第二种方法为手动建立名单—开发人员逐段建立名单, 每个人负责建立各自的区域。BCM 可将独立例项编译为一个档案。

名单的建立对其相关的 XML 架构必须正确的。判定 XML 架构可为与之相关的程序, 特别是由于名单负责两个区域: 套件建立及安装后指示。名单 XML 及其使用方式详见本文末尾的附录章节。范例的使用方式信息与 WinDNA 封装程序紧密相关。

结构

该结构包含 Custom Action DLL 即「Windows Installer 自订操作 (WICA)」处理非 Windows Installer 自身的工作。已对 MSI 套件建立套用了最佳方法。设计该结构以允许工具灵活的使用在任何 Windows DNA 应用程序的部署及安装。

该结构亦允许支持二进制的安装, 如 MDAC (ADO) 组件的最新版本, 以合并模式包含于 MSI 套件中。

套件工具非常依赖名单提供安装 Windows DNA 应用程序的大量规则。

套件工具

封装工具读取名单以取得用于封装 MSI 套件所需的指示组。封装程序包含所有列出的档案并将其精简到 CAB 压缩档案中, 此档案实体性的储存于 MSI 套件中。

MSI 套件为一包含档案部署信息, 如其大小及位置, 的 SQL 型数据库。该套件亦包含下列信息: 登录设定、ODBC 设定、NT 服务安装程序、Windows Installer 自订操作组件及带有安装后信息的名单。

MSI 套件可与其它 MSI 套件相结合，即为合并模块。合并模块可由「组合」功能设定加以灵活扩充。例如，包含最新的「使用中数据对象 (ADO)」的 MDAC 2.5，它可自行封装，然后并入需要 ADO 2.5 的 Windows DNA 应用程序中。

图 8 显示该结构的构成。由 Windows Installer API 组成的工具将读取该名单，并相应地建立安装 MSI 套件。它亦将该名单及安装后「Windows Installer 自订操作」组件并入该套件。安装程序 MSI 套件可能透过 Windows Installer 转型，结合合并模块或其它 MSI 套件。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 8 架构撰写

图 9 显示如何撰写安装 MSI 套件。如果它结合「MSI/合并模块」套件，将造成如下列图表中显示的转型。合并模块为可与其它 MSI 套件结合的 MSI 套件。

如果您的浏览器不支持内嵌框架，请[按一下这里](#)，以在不同的页面检视。

图 9 安装 MSI 套件，包含合并模块

图 10 显示转型的 MSI 套件。

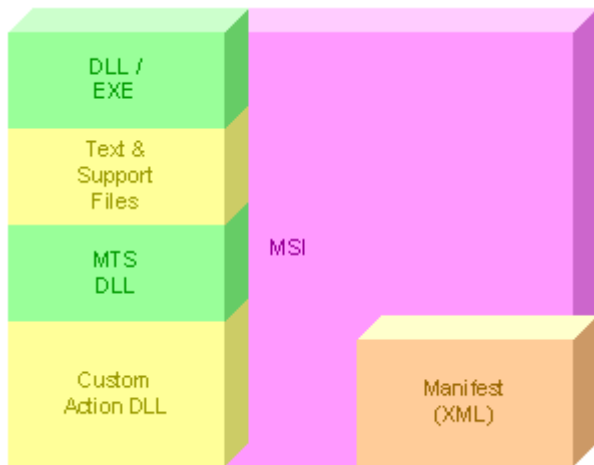


图 10 安装 MSI 套件结合 MSI/合并模块套件的转型结果

Windows Installer 自订操作 (WICA)

安装后 COM 组件 DLL，WICA 读取名单并依存于其安装、执行工作 (非 Windows Installer 自身的) 的内容及类型。这些工作包含：

MTS 登入/取消登入 自订执行 DLL 建立套件并根据名单的内容将组件填入其中。如果出现移除旗标，自订操作 DLL 在执行有效性检查后移除套件。Windows Installer 自身支持 Windows 2000 系统的 COM+ 封装，但不支持 NT 4.0 Server 的 COM+ 封装。

IIS 安装/移除。 自订操作 DLL 根据名单的内容设定虚拟目录。如果出现移除旗标，自订操作 DLL 在执行有效性检查后移除虚拟目录。

数据库安装程序。 自订操作 DLL 执行名单所含的 SQL 指令。它亦寻找名单中的复原指示。名单亦包含复原 SQL 指令。

指令行执行。 自订操作组件执行指令行操作，如 Windows Scripting Host 及批次檔。

有效性检查。 移除套件或虚拟目录前，自订操作 **DLL** 检查目标套件或虚拟目录是否为空。如果如此，则已移除该套件或虚拟目录。

摘要

WinDNA 封装程序为综合的封装解决方案，它提供 **MSI** 套件形式之分散设计应用程序的端对端覆盖。**MSI** 套件为 **SQL** 类型数据库，**Windows Installer** 服务使用它安装应用程序。充分发挥 **Windows Installer** 技术，**Windows DNA** 应用程序将具有与桌面应用程序 (如同 **Microsoft Office 2000**) 同样方便的安装。

XML 例项是指包含用于封装、部署、及安装后指导之自身说明资料集的名单。它亦在 **WinDNA** 封装程序与来源控制机制间作为额外阶层。最后，它允许负责组建程序的人员 **BCM**，将注意集中于 **Windows DNA** 应用程序封装程序，而非建立 **MSI** 套件必需的技术机制。

安装后组件包含于 **WinDNA** 封装程序，即为「**Windows Installer** 自订操作 (**WICA**)」。 **WICA** 负责 **Windows Installer** 自身不执行的操作—安装后步骤—。这些步骤包含 **Windows NT 4.0 Server** 上 **MTS** 套件的登录，它亦可用于 **Windows 2000** (尽管 **Windows Installer** 自身支持 **COM+** 套件建立) 的 **COM+** 套件。安装后的步骤额外地包含 **IIS** 虚拟目录的安装程序及其存取权限，及建立/销毁/修改数据库的 **SQL** 查询及指令文件执行方式，因为大多数 **Windows DNA** 应用程序由数据库驱动。**WICA** 亦将执行命令行操作，如 **Windows Scripting Host** 或批次檔。**WICA** 需要其名单以取得用于安装后指示的信息。

在分布式网络管理系统，如 **Microsoft Systems Manager Server (SMS)** 或 **Microsoft Application Center 2000 Server** 的协助下，**MSI** 套件部署及安装可在远程以排定且自动化方式完成。然而，**WinDNA** 封装程序亦可远程藉由扩充其功能安装 **MSI** 套件，此过程藉由程序化结合「**Windows** 管理设备 (**WMI**)」对象模型来完成。

附录 A—XML 档案

XML 档案中的元素建立可并入较大规格。该工具组特别得寻找某些高层元素标记：

- Project
- Component
- CustomAction

<Project> 标记定义的在阶层顶端，有两个子系节点：<Component> 及 <CustomAction>。在 MSI 组建期间未涉及 <CustomAction> 元素及其树状结构。自订操作安装期间，<CustomAction> 元素标记作为非 Windows Installer 自身工作的指示。

附录 B 为假定的分布式 Web 应用程序定义封装指示。它亦包含其架构定义之标准及基本原理。

附录 C 为假定的分布式 Web 应用程序定义安装后指导的指示设定。它亦包含其架构定义之标准及基本原理。

附录 B—MSI 套件

<Project>

部署工具组向父系 <Product> 标记的属性寻求内容指派。正确的属性标记 (区分大小写) 为：

Schema	// XML 架构版本	As String
Title	// 产品标题	As String
Version	// 产品版本	As String
Subject	// 产品主旨	As String
Comments	// 产品批注	As String
Keywords	// 产品关键词	As String
ProductName	// 产品名称	As String
ProductID	// 产品识别码	As String
ProductCode	// 产品 GUID	As String
DialogCaption	// 产品对话框标题	As String
HelpLink	// 产品说明 URL	As String
HelpPhone	// 产品说明行	As String

Manufacturer	// 产品制造商	As String
About	// 产品关于内容	As String
Info	// 产品信息	As String

例如:

```
Schema="1.0"  
Title="Reference Implementation"  
Version="1.0.0.0"  
Subject="RefImp"  
Comments="Reference Implementation"  
Keywords="Installer; MSI Database"  
ProductName="RefImp"  
ProductID="None"  
ProductCode="{44E68B27-768D-400a-9434-1A8B4C991E98}"  
DialogCaption="Windows Installer"  
HelpLink="http://www.microsoft.com"  
HelpPhone="555-1212"  
Manufacturer="Microsoft"  
About="http://www.microsoft.com"  
Info="http://www.microsoft.com"
```

<Component>

<Component> 标记作为其它元素的容器包含工具组用来成功地填入目标 MSI 数据库的信息。**<Component>** 标记有三个属性:

- Name
- Feature
- Type

name 属性为 `<Component>` 标记容器的唯一识别元。该属性的值为任意的正确的字符串，且不应包含任何空格。

例如: `name="Web"`

feature 功能定义 `<Component>` 为何种容器。正确的值 (区分大小写) 为:

- `Client` // 用于与 web 相关的档案。
- `Server` // 用于 DLL 及 EXE 档案。
- `Support` // 用于任意其它档案 (例如自述文件)

type 属性定义 `<Component>` 为何种容器。正确的值 (区分大小写) 为:

- `Text` // 用于与文字相关的档案。
- `COMDLL` // 用于 DLL 及 EXE 档案。
- `SELFREG_COMDLL` // 用于自身登录的 DLL 及 EXE 档案。
(很少使用)
- `Data` // 用于与数据库相关的档案。

- Registry // 用于更新登录设定。

例如: <Component name="Web" feature="Client" type="Text">

第二级元素为

- Directory
- FileListing
- RegListing

<Directory>

Directory 元素为档案部署定义来源及目标目录之元素标记的容器。正确的元素标记为:

- Source
- DestRoot
- DestParent
- DestRelative

Directory 元素有称为 id 的属性标记。该属性标记的数值为不同于每个 Windows Installer 组件的 GUID 值。此属性为选择性的, 并用于版本控制。

<Source>

Source 标记定义可找到档案的目录位置。

<DestRoot>

DestRoot 标记定义 MSI 数据库中的主键。此标记中不能含有空间，亦不能以数字开始。如果 `Component.type = "Registry"`，"DestRoot" 有特定的值 (区分大小写) 为：

- HKEY_CLASSES_ROOT
- HKEY_CURRENT_USER
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_USERS_SELECTABLE

<DestParent>

DestParent 标记定义根目录结构。此标记正确的值为：

- INSTALLDIR (c:\program files\[product name]\ 或任意其它路径为执行阶段中所定义)
- TARGETDIR (c:\)
- 任意 DestRoot 值 (并非 if Component.type = "Registry")

<DestRelative>

DestRelative 标记定义相关路径。

例如：

```
<Directory>  
<Source>D:\stuff\docs\microsoft\msi\server\</Source>  
<DestRoot>DirServer</DestRoot>  
<DestParent>INSTALLDIR</DestParent>
```

```
<DestRelative>Server</DestRelative>  
</Directory>
```

在此范例中，工具组将在 D:\stuff\docs\microsoft\msi\server\ 目录中搜寻装配的档案，并将将其部署到 C:\program files\Microsoft\RefImp\Server 目录下的目标机器。

<FileListing>

FileListing 元素包含一个属性及子系<File> 标记。RequiredForUninstall 属性定义解除安装过程中 Windows Installer 是否将移除 <Component> 容器中的档案。正确的值为：

- True
- False

例如：<FileListing RequiredForUninstall="False">

<File>

File 子系元素包含所需之理想的部署档案。

例如：

```
<File>Default.htm</File>
```

例如：

```
<FileListing RequiredForUninstall="False">  
<File>Default.htm</File>
```

```
<File>myPage.htm</File>
```

```
</FileListing>
```

<RegListing>

RegListing 元素列出 COMDLL 组件所开放，并有定义主要类型之属性，Type 的所有 Prog Id。

<Reg>

子系 Reg 元素包含一个属性，name，用来定义 Prog Id，并可选择性的将 CLSID 作为数值提供。

例如：

```
<RegListing Type="REG_SZ">
```

```
<Reg name="OrdersBO.Orders">{8594164F-2744-11D3-8FAA-204C4F4F5020}</Reg>
```

```
<Reg name="OrdersBO.OrderChanges" />
```

```
</RegListing>
```

附注 OrdersBO.Orders 特别的定义其本身的 CLSID。此为对二进制兼容性的建议作法，并提供您对管理安装更有效的控制。如果不支持 CLSID，如 OrdersBO.OrderChanges，该工具将取得 CLSID。

范例

此处为四个正确的输入的范例。第一个是对于 Web 档案，第二个是对于 COM 组件，第三个是对于数据库部署，第四个用来更新登录。

```
<Component name="Web" feature="Client" type="Text">
```

```
<Directory>
```

```
<Source>D:\stuff\docs\microsoft\msi\web\</Source>
<DestRoot>DirClient</DestRoot>
<DestParent>INSTALLDIR</DestParent>
<DestRelative>Client</DestRelative>
</Directory>
<FileListing RequiredForUninstall="False">
<File>default.asp</File>
<File>changes.asp</File>
</FileListing>
</Component>
<Component name="OrdersBO" feature="Server" type="COMDLL">
<Directory>
<Source>D:\stuff\docs\microsoft\msi\server\</Source>
<DestRoot>DirServer</DestRoot>
<DestParent>INSTALLDIR</DestParent>
<DestRelative>Server</DestRelative>
</Directory>
<FileListing RequiredForUninstall="False">
<File>OrdersBO.dll</File>
</FileListing>
<RegListing Type="REG_SZ">
<Reg name="OrdersBO.Orders">{8594164F-2744-11D3-8FAA-204C4F4F5020}</Reg>
<Reg name="OrdersBO.OrderChanges" />
</RegListing>
</Component>
<Component name="DB" feature="Support" type="Data">
```

```
<Directory>
<Source>D:\stuff\docs\microsoft\msi\db\</Source>
<DestRoot>DirSupport</DestRoot>
<DestParent>INSTALLDIR</DestParent>
<DestRelative>Support</DestRelative>
</Directory>
<FileListing RequiredForUninstall="False">
<File>restore.sql</File>
</FileListing>
</Component>
<Component name="Registry1" feature="Support" type="Registry">
<Directory>
<DestRoot>HKEY_LOCAL_MACHINE</DestRoot>
<DestParent>Software\Microsoft</DestParent>
<DestRelative>RefImp</DestRelative>
</Directory>
<RegListing Type="REG_SZ">
<Reg name="ManifestLocation">c:\winnt\system32\</Reg>
<Reg name="ManifestFileName">wical.xml</Reg>
</RegListing>
</Component>
```

附录 C—自订执行



当目标 MSI 套件部署在目标机器上，其安装即已初始化并且已部署档案，自订执行组件将开始发挥作用。自订执行组件将读取包含在目标 MSI 套件中的 XML 名单以取得进一步指示。

<CustomAction>

自订执行组件将特别的期望 <CustomAction> 标记元素成为高等级阶层。

<CustomAction> 标记包含两个属性：

- Name
- Type

并有两个子系元素标记：

- Server
- Var

name 属性可为任何正确的字符串。

type 属性包含指定的正确的值，可为：

- IIS // IIS 设定
- MTS // MTS 设定
- DB // SQL 数据库查询
- CMD // 指令行执行

<Server>

子系元素标记, <Server> 包含两个属性:

- Name
- Binding

name 属性的值取决于 **CustomAction.type** 的值。

```
IF CustomAction.type = "IIS" THEN
```

Server.name 用于定义服务器名称。

例如, LocalHost

```
ELSE IF CustomAction.type = "DB" THEN
```

Server.name 用于定义服务器名称。

例如, LocalHost

```
ELSE IF CustomAction.type = "MTS" THEN
```

Server.name 用于定义已授权的使用者

对于 MTS 套件。例如, quocbui。如果此属性

留空白, 自订操作组件将预设为

MTS 交互式使用者设定。

```
END IF
```

binding 属性的值取决于 **CustomAction.type** 的值。

```
IF CustomAction.type = "IIS" THEN
```

Server.binding 用于定义连接埠编号。例如, :80:

```
ELSE IF CustomAction.type = "DB" THEN
```

Server.name 用于定义数据库认证。

例如, Provider=SQLOLEDB; User Id=sa; Password=;

或 -U sa -P

```
ELSE IF CustomAction.type = "MTS" THEN
```

Server.name 用于定义 MTS 套件之授权使用者名称的

密码。例如, myPassword

```
END IF
```

Server 的第三级元素标记取决于 CustomAction.type 的值。正确的标记为:

<SiteDir>

- SiteDir // 仅用于 IIS。

定义虚拟目录的路径。

<SiteRoot>

- SiteRoot // 仅用于 IIS。

定义 Web 服务器节点。

<VirDir>

- VirDir // 仅用于 IIS。

定义虚拟目录名称。

<Source>

- Source // 仅用于 MTS 及 DB。

对于 MTS, 它定义

MTS DLL 档案的位置。对于 DB，它定义数据库名称。

<Var>

<CustomAction> 的第二个子系标记为 Var。包含两个属性：

- name
- binding

并有一个子系元素，

- Value

name 属性的值取决于 CustomAction.type 的值。

IF CustomAction.type = "IIS" THEN

Var.name 用于定义验证参数。正确的值为：

- dwAuthVars
- dwAccessVars

ELSE IF CustomAction.type = "DB" THEN

Var.name 用于定义执行类型。正确的值为：

- sqlFile // 透过 ISQL 执行 SQL 档案
- sqlQuery // 直接执行 SQL 查询

ELSE IF CustomAction.type = "CMD" THEN

Var.name 用于名称执行类型。唯一的正确值为：

- cmdFile // 执行命令行

ELSE IF CustomAction.type = "MTS" THEN

- Var.name 用于定义 MTS 套件的套件名称。

例如, RefImp

END IF

binding 属性的值取决于 **CustomAction.type** 的值。

IF CustomAction.type = "IIS" THEN

Var.name 用于定义验证参数。唯一的正确值为:

- Flag

ELSE IF CustomAction.type = "DB" THEN

Var.name 用于定义执行类型。唯一的正确值为:

- Install // 在期间执行
- Remove // 在解除安装期间执行

ELSE IF CustomAction.type = "CMD" THEN

Var.name 用于定义执行类型。唯一的正确值为:

- Install // 在安装期间执行
- Remove // 在解除安装期间执行

ELSE IF CustomAction.type = "MTS" THEN

Var.name 用于定义套件的安全性类型。正确的值为:

- SecurityEnabled
- <Blank>

END IF

<Value>

Var 的第三等级元素标记, 取决于 **CustomAction.type** 的值。唯一的正确的标记为 **Value**。**Var** 父系中可有許多 **Value** 标记。

IF CustomAction.type = "IIS" THEN

IF CustomAction.Var.name = "dwAuthVars" THEN Valid values are:

- AUTH_INHERIT // 继承父系 AuthFlags
- AUTH_ANON // 仅用于 Web, 匿名验证可用
- AUTH_BASIC // 仅用于 Web, 基本验证可用
- AUTH_SSPI // 仅用于 Web, SSPI 验证配置可用
- ALLOW_ANON // 仅用于 ftpSrv
- ANON_ONLY // 仅用于 ftpSrv

ELSE IF CustomAction.Var.name = "dwAccessVars" THEN Valid values are:

- ACCESS_INHERIT // 继承父系 AccessFlags
- ACCESS_READ // 允许读取存取
- ACCESS_WRITE // 允许写入存取
- ACCESS_EXECUTE // 仅用于 Web, 允许档案执行 (包含指令文件使用权限), 仅用于 Web
- ACCESS_SCRIPT // 仅用于 Web, 允许指令文件执行, 仅用于 Web
- APP_CREATE // 仅用于 Web, 定义应用程序启动符号指针
- APP_INPROC // 仅用于 Web, 需要设定 app_create
- APP_OUTPROC // 仅用于 Web, 需要设定 app_create
- APP_DELETE // 仅用于 Web, 删除应用程序
- DIR_BROWSE // 仅用于 Web, 启用目录浏览
- FRONTPAGE_WEB // 仅用于 webSrv, 启用 fpExtensions, 需要设定 app_create
- INDEX_DIR // 仅用于 Web, 索引目录
- LOG_ACCESS // 记录文件存取

END IF

ELSE IF CustomAction.type = "DB" THEN

- 任意正确的 SQL 查询或路径及 SQL 文件的档案。

ELSE IF CustomAction.type = "CMD" THEN

- 任意正确的指令行

```
ELSE IF CustomAction.type = "MTS" THEN
```

- 利用 MTS 档案总管登录的 COM DLL

```
END IF
```

范例

例如:

这组指示将设定称为 **RefImp** 的应用程序虚拟目录, 其路径为 `c:\program files\Microsoft\client`, and falls under the "Default Web Site" node. 它的安全性设定为: 允许匿名、允许基本验证及 **sspi**。它的档案设定为: 读取、建立及执行。

```
<CustomAction name="IIS1" type="IIS">
<Server name="LocalHost" binding=":80:">
<SiteDir>c:\program files\microsoft\refimp\client</SiteDir>
<SiteRoot>Default Web Site</SiteRoot>
<VirDir>RefImp</VirDir>
</Server>
<Var name="dwAuthVars" binding="Flag">
<Value>AUTH_ANON</Value>
<Value>AUTH_BASIC</Value>
<Value>AUTH_SSPI</Value>
</Var>
<Var name="dwAccessVars" binding="Flag">
<Value>ACCESS_READ</Value>
<Value>ACCESS_EXECUTE</Value>
<Value>APP_CREATE</Value>
```

```
</Var>  
</CustomAction>
```

这组指示将建立称为 **RefImp** 的 **MTS** 套件，该套件中会带有两个 **COM DLL**、**OrdersBO.dll** 及 **OrdersDB.dll**。该套件设定针对交互式使用者，并且启用安全性。

```
<CustomAction name="MTS1" type="MTS">  
<Server name="" binding="">  
<Source>c:\program files\microsoft\refimp\server</Source>  
</Server>  
<Var name="RefImp" binding="SecurityEnabled">  
<Value>OrdersBO.dll</Value>  
<Value>OrdersDB.dll</Value>  
</Var>  
</CustomAction>
```

这组指示将在安装中执行一 **SQL** 档案，其设定如下：

```
isql -H LocalLost -U sa -P -i c:\program files\microsoft\refimp\support\restore.sql  
<CustomAction name="DB1" type="DB">  
<Server name="LocalHost" binding="-U sa -P">  
<Source>MYDATABASE</Source>  
</Server>  
<Var name="sqlFile" binding="Install">  
<Value>c:\program files\microsoft\refimp\support\restore.sql</Value>  
</Var>  
</CustomAction>
```

下列一组指示将在解除安装中执行如下的 SQL 查询:

MYDATABASE 数据库中的 DROP DATABASE MYDATABASE

利用下列联机字符串, Provider=SQLOLEDB; User Id=sa; Password=;

```
<CustomAction name="DB1" type="DB">
```

```
<Server name="LocalHost" binding="Provider=SQLOLEDB; User Id=sa; Password=";>
```

```
<Source>MYDATABASE</Source>
```

```
</Server>
```

```
<Var name="sqlQuery" binding="Remove">
```

```
<Value>DROP DATABASE MYDATABASE</Value>
```

```
</Var>
```

```
</CustomAction>
```

下列一组指示将在安装中执行 dos 型批次檔:

```
<CustomAction name="CMD1" type="CMD">
```

```
<Var name="cmdFile" binding="Install">
```

```
<Value>c:\program files\microsoft\refimp\support\run.bat</Value>
```

```
</Var>
```

```
</CustomAction>
```

参照

[Windows Installer](#)

InstallShield

XML