

# 科技部科技基础性工作专项资金重大项目 研究成果

项目名称：我国数字图书馆标准与规范建设

项目负责人：张晓林

子项目名称：数字图书馆集成服务描述标准规范研究

项目编号：2005DKA43503

研究成果类型：研究报告

成果名称：WSRF 标准规范体系研究

成果编号：[按格式规定]

成果版本：征求意见稿

成果提交日期：2007-02-12

撰写人：韩涛（中国科学院国家科学图书馆）

## 项目版权声明

本报告研究工作属于科技部科技基础性工作专项资金重大项目《我国数字图书馆标准规范建设》的一部分，得到科技部科技基础性工作专项资金资助，项目编号为 2002DEA20018。按照有关规定，国家和《数字图书馆标准规范建设》课题组拥有本报告的版权，依照《中华人民共和国著作权法》享有著作权。

本报告可以复制、转载、或在电子信息系统上做镜像，但在复制、转载或镜像时须注明真实作者和完整出处，并在明显地方标明“科技部科技基础性工作专项资金重大项目《数字图书馆标准规范建设》资助，项目编号 2002DEA20018”的字样。

报告版权人不承担用户在使用本作品内容时可能造成的任何实际或预计的损失。

## 作者声明

本报告作者谨保证本作品中出现的文字、图片、声音、剪辑和文后参考文献等内容的真实性和可靠性，愿按照《中华人民共和国著作权法》，承担本作品发布过程中的责任和义务。科技部有关管理机构对于本作品内容所引发的版权、署名权的异议、纠纷不承担任何责任。

《数字图书馆标准规范建设》课题组网站 (<http://cdls.nstl.gov.cn>) 作为本报告的第一发表单位，并可向其他媒体推荐此作品。在不发生重复授权的前提下，报告撰写人保留将经过修改的项目成果向正式学术媒体直接投稿的权利。

# 目 录

1 引言 .....	1
2 WSRF缘起 .....	1
2.1 WSRF产生背景.....	2
2.1.1 Web Services的演变.....	2
2.1.2 OGIS的发展与不足.....	3
2.2 WSRF与Web Services、OGIS的关系.....	6
3 WSRF标准规范的详细内容.....	7
3.1 WS-Resource和隐性资源模式.....	9
3.2 WS-ResourceLifetime规范.....	11
3.2.1 WS-Resource工厂模式 (WS-Resource Factory Pattern) .....	11
3.2.2 WS-Resource的身份标识 (WS-Resource Identity) .....	12
3.2.3 WS-Resource的销毁 (WS-Resource Destruction) .....	13
3.3 Web 服务资源的特性 (WS-ResourceProperties) .....	14
3.3.1 WS-Resource的特性文档.....	14
3.3.2 WS-Resource的特性组合.....	16
3.3.3 访问WS-Resource特性的值.....	17
3.4 可更新的引用 (WS-RenewableReferences) .....	19
3.5 服务小组 (WS-ServiceGroup) .....	19
3.6 基本错误 (WS-BaseFaults) .....	19
3.7 通知 (WS-Notification) .....	20
4 WSRF与OGIS、WSMF的比较分析.....	21
4.1 WSRF与OGIS比较研究.....	21
4.1.1 相同点.....	21
4.1.2 不同点.....	22
4.1.3 WSRF对OGIS的改进.....	24
4.2 WSRF与WSMF比较研究.....	25
4.2.1 相同点.....	26
4.2.2 不同点.....	27
4.2.3 两者的关系.....	31
5 WSRF的实现 .....	33
5.1 GT4 .....	33
5.2 CGSP .....	36
5.3 Apache WSRF (Apollo), Pubscribe (Hermes) 和Muse.....	38
5.4 WSRF.NET .....	40
6 结论 .....	41
7 名词术语 .....	41
8 参考文献 .....	43

## 插图和附表

图 2.1 OGS I 定义的网格服务接口 .....	6
图 2.2 WSRF 实现 Web Services 与网格的交汇 .....	6
图 2.3 WSRF 和 WS-Notification 替代 OGS I 成为 OGSA 新的基础 .....	7
图 3.1 WSRF 家族规范的关系 .....	8
图 3.2 通过隐性资源模式创建 WS-Resource 的过程 .....	10
图 3.3 端点引用的结构 .....	11
图 3.4 直接发布通知方式 .....	20
图 3.5 代理发布通知方式 .....	21
图 4.1 WSMF 家族关系 .....	26
图 4.2 左图为 WSRF 的隐式资源模式，右图为 WSMF 的资源管理模式 .....	29
图 4.3 实现语义网格服务过程中 WSRF 和 WSMF 互为补充的关系 .....	32
图 5.1 GT4 的整体框架图 .....	34
图 5.2 CGSP 中服务容器体系结构图 .....	37
图 5.3 CGSPv2 beta1 第一版的服务容器体系结构图 .....	37
图 5.4 WSRF.NET 框架中的信息流 .....	40
表 3.1 WSRF 的 5 个子技术规范及其功能描述 .....	9
表 4.1 WSRF 和 OGS I 在生命周期管理方面的比较 .....	22
表 4.2 WSRF 和 OGS I 的全面比较 .....	24
表 4.3 WSRF 和 WSMF 对比小结 .....	32
表 5.1 GT4 主要协议的基本特性和兼容性 .....	34

## 1 引言

WSRF<sup>[1]</sup> (Web Service Resource Framework, Web服务资源框架)是网格计算领域于2004年1月提出的一组Web服务规范,参与的机构和公司有Globus Alliance、IBM、HP、SAP、Akamai、TIBCO、Sonic等。作为OGSI (Open Grid Service Infrastructure, 开放网格服务基础设施)的重构和发展,WSRF引入了有状态资源 (stateful resource)的概念,描述了有状态资源和Web服务之间的关系,给出了两者关联形成WS-Resource (Web服务资源)以及通过Web服务接口访问WS-Resource状态的方法。作为WSRF规范框架中的重要组成部分,WSRF还定义了与WS-Resource分组 (grouping)、寻址 (addressing)、属性操作 (property operation)等相关的机制。

本文将Web Services<sup>[2]</sup>和网格的发展为背景,介绍WSRF产生的原因,在此基础上着重描述WSRF的详细标准规范,对WSRF分别同网格领域的OGSI和Web服务领域的WSMF进行比较分析,最后介绍各机构和公司如何实现WSRF规范。本文具体章节安排是:第一部分引言;第二部分WSRF缘起,简介WSRF的产生背景以及与Web Services、OGSI的关系;第三部分详细描述WSRF的体系架构和标准规范,包括WS-ResourceLifetime (WS-Resource生命周期规范)、WS-ResourceProperties (WS-Resource特性规范)、WS-RenewableReferences (Web服务可更新引用规范)、WS-ServiceGroup (Web服务小组规范)、WS-BaseFaults (Web服务基本错误规范)以及不属于WSRF但与WSRF有密切关系的WS-Notification (Web服务通知规范);第四部分比较分析WSRF与OGSI、WSRF与WSMF的异同点与相互关系;第五部分介绍如何使用WSRF开发Web服务应用,即提供Web服务解决方案的几大供应商和机构如何结合自身的特点开发遵循WSRF的应用;第六部分结语;第七部分给出文中出现的重要名词术语;第八部分参考文献。

## 2 WSRF 缘起

WSRF可以说是OGSI与Web Services结合的产物,因而WSRF的产生与两者有密切的关系。该部分将结合Web Services和OGSI论述WSRF的产生背景,分析WSRF与Web Services、OGSI的关系。

## 2.1 WSRF 产生背景

### 2.1.1 Web Services 的演变

2000 年伴随着 SOAP<sup>[3]</sup> (Simple Object Access Protocol)、WSDL<sup>[4]</sup> (Web Services Description Language)、UDDI<sup>[5]</sup> (Universal Description, Discovery & Integration) 技术的引入, Web Services 出现在人们的视野中。W3C<sup>[6]</sup> 的 Web Services 体系工作组<sup>[7]</sup> (W3C Web Services Architecture Working Group) 给出 Web 服务 (Web Service) 的定义<sup>[8]</sup> 是: Web 服务是一个可以支持网络上计算机与计算机互操作的软件应用, 它使用计算机可处理的格式 (一般使用 WSDL) 描述其接口, 使用 SOAP 与其他系统的 Web 服务进行消息交互, 使用 HTTP 进行网络数据传递。虽然这种定义的初衷试图强调某个 Web 服务的请求者只需关注该服务的接口描述而无需明确服务内部的运行机制 (因为服务的运行仅仅由消息交换驱动), 但是 Web 服务实际应用中通常会有一种 “暗示”: Web 服务的消息请求过程中所操作的有状态资源的确是存在的, Web 服务看似对用户透明, 实际上还管理着与 “状态” 有关的一系列文档、数据, 开发者也不得不从 Web 服务接口消息中获取有状态资源的标识符才能实现状态的调用和操作, 而且有状态资源标识符将成为一个参数携带着状态数据在以后的消息交互中反复往返<sup>[9]</sup>。可见, Web 服务的实际应用过程无法回避状态的存在, 而用一种标准方式明确地表述 Web 服务与状态之间的关系就显得格外重要。

随着 Web Services 的演变, Web 服务出现三种形式<sup>[9]</sup>。一种是无状态服务 (stateless service)。在消息交换时, 无状态服务只从输入的消息中获取信息, 除输入的消息外, 无状态服务无法获取与之交互消息的对象的其他信息。比如压缩/解压缩服务只能从消息中获取待处理的数据, 而不关心这些数据是否已被压缩/解压缩。还有一种是流程服务 (conversational service)。流程服务不只有一个操作而包含一系列操作, 某个操作的结果取决于前一个操作或服务于后一个操作。这种服务会依据一个逻辑消息流决定整个服务的操作流程, 也就是说 Web 服务的运行由按一定逻辑顺序组织的消息队列决定。许多交互式的 Web 网站就运用 HTTP 协议的 Sessions 和 Cookies 技术实现这种服务。第三种是作用于有状态资源的服务。这种服务通过接收、发送消息, 能够访问、操作一系列有状态的资源, 此时服务与有状态资源虽是两种对象, 但被紧密联系在一起, Web 服务就不能再被简单地定义为无状态的或有状态的了。

其实, Web 服务的无状态与有状态都不是绝对的, 应该从 Web 服务的演化过程看两者的相对关系。Web 服务从无状态演变成有状态, 并不是说 Web 服务本身是有状态的, 而是说 Web 服务通过演变应该具备面向有状态资源的能力, 或者说

应该为 Web 服务建立一套机制使其能“直视”状态的存在而不要回避。一个方面，如果资源的状态属性交由系统中的其他组件如文件管理系统、数据库管理系统负责管理，那么能够作用于有状态资源的 Web 服务本质上就是无状态的。Web 服务这种在执行时的无状态性可以提高系统的稳定性和可靠性，因为在遇到错误时无状态的 Web 服务能立刻被重新启动而不需考虑之前的交互情况进行得如何。因此，Web 服务的无状态性在工程应用中许多优点，不应该被抹杀。然而从另一方面说，面对有状态的资源，Web 服务无状态地执行过程必须考虑到消息交换的过程中一定存在一些动态变化的状态属性，这些状态属性必须在消息请求时无论是以直接的数值形式还是以间接的引用形式都要能被明确地提供，而且还允许被与 Web 服务交互的其他一些系统组件“隐性”地维护。由此可见，Web 服务的演变不是抛弃无状态性而一味追求有状态性，其核心思想是创建作用于有状态资源的 Web 服务，这种服务最主要的特性就是：无状态的服务在执行时能频繁地读取、更新资源的状态属性，而资源状态属性的管理和维护交由数据库等其他系统组件来完成；在这种情况下，资源状态属性要能在消息请求中被传递，而且能被 Web 服务当作静态数据操作。总之，在演变中 Web 服务要继续保持其执行过程中的无状态性，而在接口定义上则应该突出更清晰的状态性，使其能与隐藏的资源状态建立必要的联系。

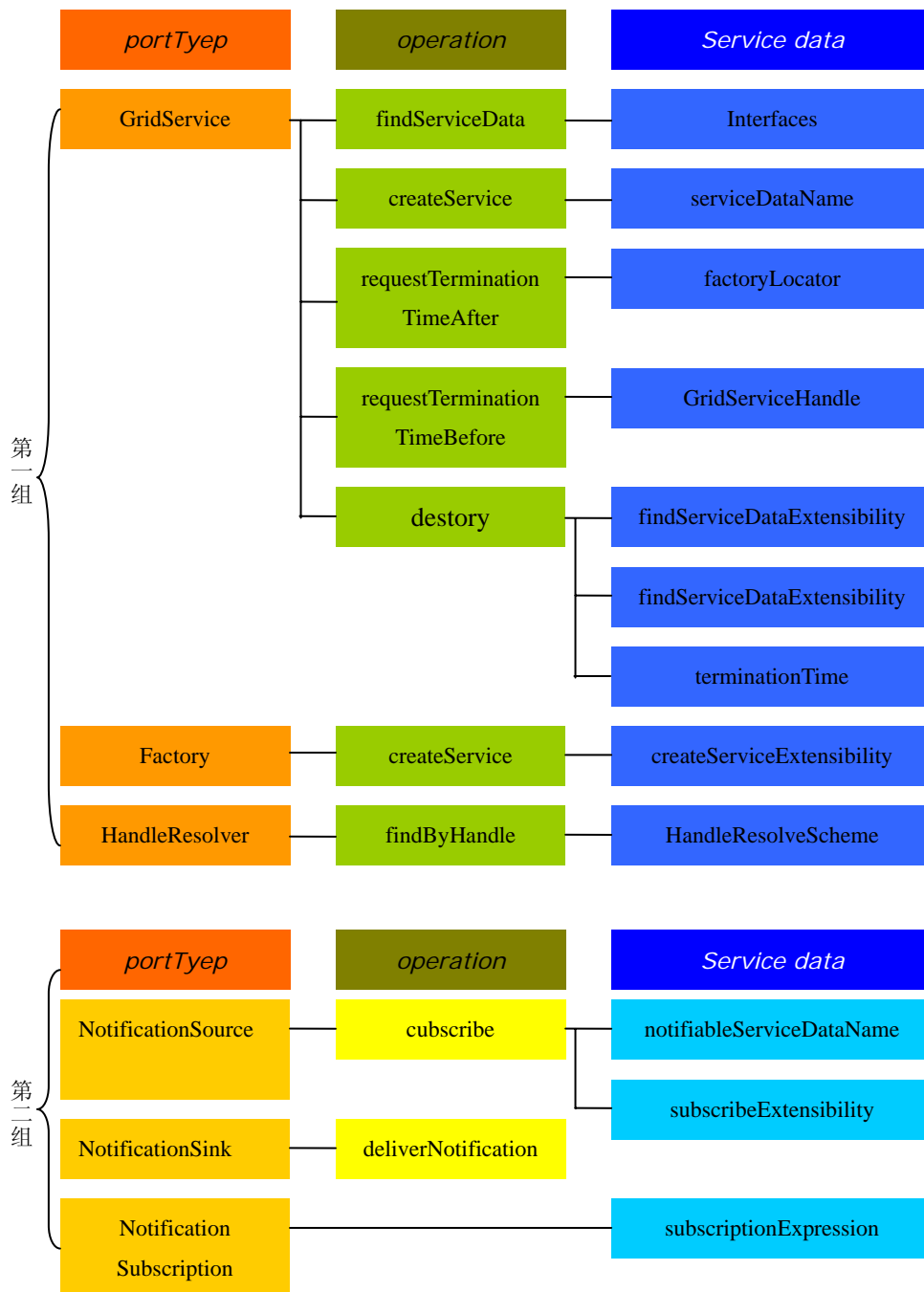
### 2.1.2 OGSII 的发展与不足

OGSI 是构建开放网格服务体系结构 OGSA<sup>[10]</sup> (Open Grid Service Architecture, 为网格环境中要提供的总体结构和服务定义了标准) 的基础设施，定义在网格环境中各种服务之间使用的接口和协议的标准，也定义在使用 OGSA 设计的网格之间提供互操作性的标准<sup>[11]</sup>。OGSI1.0<sup>[12]</sup> 发布于 2003 年 7 月，核心内容是网格服务标准规范。OGSI 引入了一种网格服务 (Grid Service) 的交互模型，通过提供发现、生命周期、状态管理、创建与销毁、事件通知以及引用管理的接口，OGSI 为软件人员提供了一种统一的建模和与网格服务进行交互的方式。根据这些接口的功能，可以把这些接口分为三组<sup>[13]</sup>。第一组是支持网格服务行为、服务数据元素和静态服务数据值的 portType (端口类型)；第二组是关于通知框架的 portType；第三组 portType 提供了网格服务成组的概念。具体如图 2.1 所示。

随着发展，除了上述接口功能外，OGSI 还定义了一组关于利用 WSDL 和 XML Schema 等 Web 服务机制的约定与扩展，用以启动有状态的网格服务 (stateful Grid Service)。它引入了有状态 Web 服务的观念并定义了一些方法，可以用于创建、命名和管理服务实例的生命周期；用于声明和检查服务状态数据；用于服务状态更改的异步通知；用于表示和管理服务实例的集合；以及用于服务调用故障的共同处理。具体地说，OGSI 涉及到以下内容<sup>[14]</sup>：1、WSDL 扩展集，其中一部分支持

WSDL2.0; 2、一个具体服务的WSDL的构建, 以及描述、查询、更改与该服务相关的数据(服务的描述元数据和状态属性数据)的方法; 3、GSH(Grid Service Handle, 网格服务句柄)和GSR(Grid Service Reference, 网格服务引用)的构建, 用于网格服务寻址; 4、常规错误故障信息的定义, 这种定义方法不改动WSDL错误消息模式, 而是利用XML Schema以及与WSDL错误消息有关的语义知识重新定义一个错误消息的基本格式, 用以支持常规错误故障信息的交互和理解; 5、创建和销毁网格服务的方法, 销毁方法包括显性地、直接地销毁和隐性地、间接地销毁两种方法; 6、通过引用机制, 创建和使用与网格服务异质的Web服务的方法, 其目的是在网格服务中引入Web服务; 7、服务状态更改的异步通知请求机制, 主要针对网格服务数据元素值的变更。

在OGSI的实践中, 随着OGSI与Web Services结合日益紧密, Web服务领域指出OGSI有四个不足之处<sup>[14]</sup>。首先, OGSI标准规范中内容繁杂, 各功能模块间没有清晰的界限, 应用中无法分别独立采用, 众多技术内容全部集中在一个规范中, 不利于对不同部分的灵活运用。其次, OGSI与现有的Web Services和XML工具无法很好地配合工作, 它过分地使用XML Schema中xsd:any属性和WSDL面向文档(document-oriented)的操作, 因而不能得到Web服务现有工具、运行环境的有力支持。再次, OGSI过于强调面向对象的思路, 它将有状态资源也抽象为一种Web服务, 而把资源的状态属性封装在这种Web服务中。这样的处理方式导致服务的标识符、生命周期等属性同资源的状态属性混合在同一个Web服务对象中而使这些属性具有强烈的耦合关系, 但这些属性明显分别属于Web服务和有状态资源两种差异很大的对象, 如此以来致使两者之间的关系含混不清, 不利于以后运行中的管理和调配。最后, OGSI发布之时超前引入WSDL2.0, 当时的WSDL2.0还处于草案状态, 正式的WSDL2.0迟迟未推出, 因而基于WSDL1.1的Web Services运行环境和各种工具很难为OGSI提供支持。



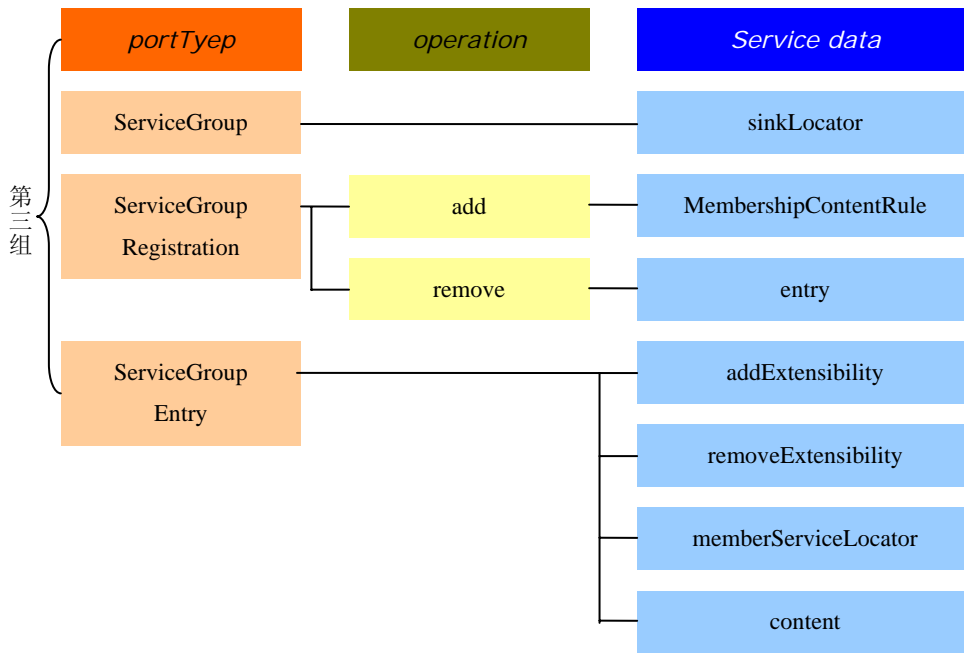


图 2.1 OGSF 定义的网络服务接口

## 2.2 WSRF 与 Web Services、OGSI 的关系

随着 Web Services 体系结构的不断发展和演变，以及 WSDL2.0 与 WS-Addressing<sup>[15]</sup>（Web 服务寻址规范）这样的新兴 Web 服务标准的出现，网络领域的人们开始考虑 OGSF 将如何利用 Web Services 及其扩展标准规范特别是 WS-Addressing，如何将 OGSF 的功能和 Web Services 体系结构整合在一起，为适应这些变化 OGSF 的功能模块又该如何重新划分。Ian Foster 对 Web Services 和网络未来的发展在何处交汇提出了疑问和思考。而图 2.2 表明，Ian Foster 认为 Web Services 与网络会在 WSRF 上交汇<sup>[16]</sup>，至少可以说 WSRF 是它们两者以后走向融合的一个起点。

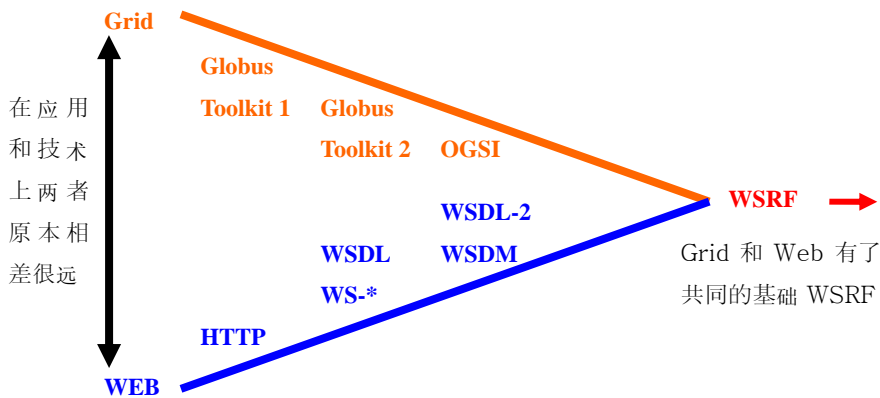


图 2.2 WSRF 实现 Web Services 与网络的交汇

因此, Ian Foster 认为 WSRF 实现了 Web 与 Grid 的融合, WSRF 充分借用 Web Services 的观念和方式成功重构了 OGSI 的结构, 改变了 OGSI 的内容, 丰富了 OGSI 的内涵, 使 OGSI 向着 Web Services 的方向发展。从 WSRF 和 WS-Notification 两部分替代 OGSI 可见, 本质上, WSRF 源于 OGSI, 是利用新的 Web Services 标准特别是 WS-Addressing 对 OGSI 的重构和发展。而且 WSRF 也将取代 OGSI 的位置, 作为一种新的基础设施(如图 2.3 所示), 为基于 Web Services 的 OGSA 提供更广泛、更强大的支持。可以说, WSRF 既充分利用已有 Web 服务领域的各种成果, 又吸纳了网格技术, 可以支持网格的需求, 为网格与 Web 的发展建立了一个共同的基础。

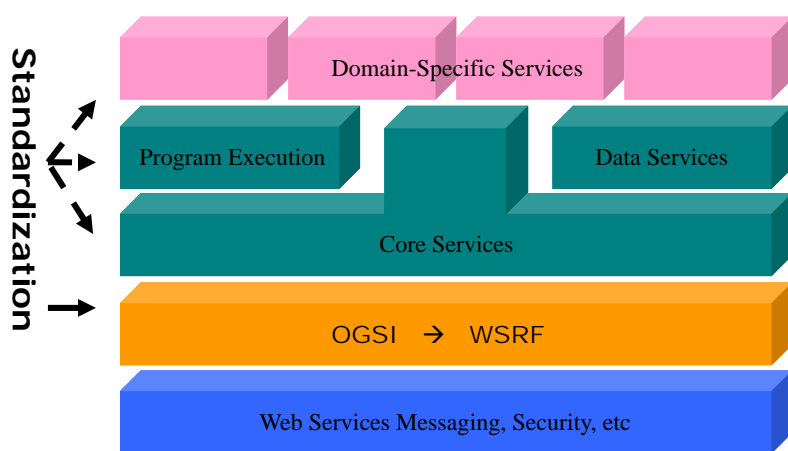


图 2.3 WSRF 和 WS-Notification 替代 OGSI 成为 OGSA 新的基础

### 3 WSRF 标准规范的详细内容

本部分内容主要参考了《The WS-Resource Framework version 1.0》<sup>[17]</sup>、《Web Services Resource 1.2 (WS-Resource) OASIS Standard, 1 April 2006》<sup>[18]</sup>、《Web Services Resource Lifetime 1.2 (WS-ResourceLifetime) OASIS Standard, 1 April 2006》<sup>[19]</sup>、《Web Services Resource Properties 1.2 (WS-ResourceProperties) OASIS Standard, 1 April 2006》<sup>[20]</sup>、《Web Services Service Group 1.2 (WS-ServiceGroup) OASIS Standard, 1 April 2006》<sup>[21]</sup>、《Web Services Base Faults 1.2 (WS-BaseFaults) OASIS Standard, 1 April 2006》<sup>[22]</sup>、《Web Services Addressing (WS-Addressing) W3C Member Submission, 10 August 2004》<sup>[15]</sup>、《Web Services Base Notification 1.3 (WS-BaseNotification) OASIS Standard, 1 October 2006》<sup>[23]</sup>、《Web Services Brokered Notification 1.3 (WS-BrokeredNotification) OASIS Standard, 1 October 2006》<sup>[24]</sup>、《Web Services Topics 1.3 (WS-Topics) OASIS Standard, 1 October 2006》<sup>[25]</sup>等文章。

Web 服务能向用户提供访问和处理状态的能力, 数据值能在 Web 服务的交互

中得以维持和变更。也就是说，Web 服务中的消息交换是我们能访问有状态的资源。然而前面提到，Web 服务所操作的有状态资源的概念特别是在接口定义上并不明确，使得我们在实现 Web 服务时总是隐约感觉它的存在而无法对它进行准确地定位、操作和管理。因此年我们强烈地希望再定一些约定，使 Web 服务能够以一种标准的、可互操作的方式来发现、解读有状态资源并与之交互，而且还能使应用程序在汇编和运行时绑定到特定的有状态资源实例，增强应用程序的健壮性。

这些现实和想法促成了在 Web 服务环境中对状态进行建模，即 WS-Resource 的提出。WS-Resource 被定义为 Web 服务和有状态资源的组合，它具有两个特点：(1) 组件状态用 XML 文档描述，使用 XML 文档定义它和 Web 服务的接口类型；(2) 采用“隐性模式”寻址和访问有状态资源，通过 WS-Addressing 的端点引用 (EPR, Endpoint Reference) 来寻址。在隐性资源模式 (Implied Resource Pattern) 中，有状态资源标识符被封装在端点引用中，用来识别在执行 Web 服务消息交换的过程中所使用的有状态资源。WSRF 通过约定的 Web 服务机制来使 WS-Resource 可以被声明、创建、访问、监测改变和销毁，但是并不需要 WS-Resource 中与有状态资源关联的 Web 服务具有状态消息处理器的功能。

WSRF 是一个包含 5 个技术规范的集合，如图 3.1 所示，它根据特定的 Web 服务消息交换和相关的 XML 定义确定了 WS-Resource 方法的标准化描述。

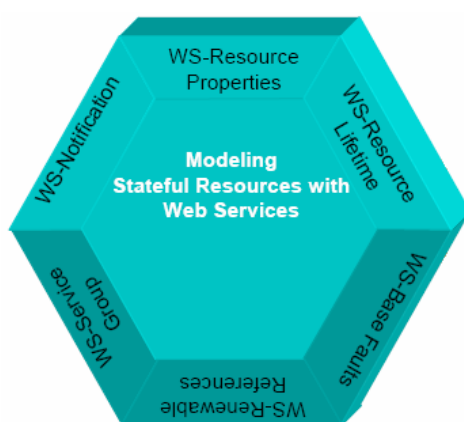


图 3.1 WSRF 家族规范的关系

表 3.1 列出了这 5 个技术规范，总体思想是：

(1) **WS-ResourceLifetime**: WS-Resource 的销毁可以与销毁请求同步，也可以通过调度析构机制来定时销毁，而且指定的资源特性可以被用来检查和监测 WS-Resource 的生存期；

(2) **WS-ResourceProperties**: WS-Resource 的类型定义可以由 Web 服务的接口描述和 XML 资源特性文档组成，并且可以通过 Web 服务消息交换来查询和更改 WS-Resource 的状态属性；

(3) **WS-RenewableReferences**: 如果 Web 服务端点引用所包含的寻址或者

策略信息变得无效或者陈旧可以被更新；

(4) **WS-ServiceGroup**: 无论服务是否是 **WS-Resource** 类型都可以定义成异构的、通过引用的访问的 **Web** 服务集合；

(5) **WS-BaseFaults**: 通过基本错误的 **XML Schema** 的使用，规范基本错误类型的应用规则，使错误报告更加标准化。

本部分我们首先介绍 **WS-Resource** 的构建和隐性资源模式，然后依次描述 **WSRF** 的 5 个子技术规范。此外也将讨论发布/订阅消息的通知机制 (**WS-Notification**) 如何建立在 **WSRF** 之上，如何通过创建对 **WS-Resource** 内部状态改变的订阅来监测和报告资源特性的改变。

表 3.1 **WSRF** 的 5 个子技术规范及其功能描述

名称	功能描述
<b>WS-ResourceLifetime</b> ( <b>Web</b> 服务资源生命期规范)	允许服务请求方销毁或是按预先计划销毁 <b>WS-Resource</b> ，从而灵活地设计 <b>Web</b> 服务应用程序如何清除不再需要的资源
<b>WS-ResourceProperties</b> ( <b>Web</b> 服务资源特性规范)	描述相关的有状态资源和 <b>Web</b> 服务来产生 <b>WS-Resource</b> ，以及 <b>WS-Resource</b> 的公共可见特性如何被获取、更改、删除； <b>WS-Resource</b> 的特性声明是 <b>WS-Resource</b> 状态的一个影射或视图
<b>WS-RenewableReferences</b> ( <b>Web</b> 服务可更新引用规范)	为某个 <b>Web</b> 服务寻址端点引用 ( <b>Address Endpoint Reference</b> ) 标注上相关信息，以便当目前引用无效时可以重新获得新的端点引用
<b>WS-ServiceGroup</b> ( <b>Web</b> 服务服务组规范)	<b>Web</b> 服务和 <b>WS-Resource</b> 可以为了某个领域的特定目的而聚集或组合在一起；为了让请求者能够根据服务组( <b>ServiceGroup</b> )的内容进行有意义的查询，必须以某种方式来限制组中成员的资格
<b>WS-BaseFault</b> ( <b>Web</b> 服务基本错误规范)	为基本错误定义一个 <b>XML</b> 模式类型以及 <b>Web</b> 服务如何使用这种错误类型的规则。当来自不同接口的可用错误信息都一致时，清求者理解错误就更加容易了。与此同时，开发一种通用的工具来帮助处理错误也变得更加可能

### 3.1 **WS-Resource** 和隐性资源模式

**WS-Resource** 的定义根据隐性资源模式 (一组关于 **Web** 服务的规范，尤其是 **XML**、**WSDL** 和 **WS-Addressing**) 确定了 **Web** 服务与有状态资源之间的关系，从而资源的状态可以被定义并与 **Web** 服务接口的描述相关联。资源的状态是由资源特性文档来定义的，因此从编码的角度看，**WS-Resource** 来就是 **Web** 服务与资源特性文档的组合。

隐性资源模式的“隐性”指对客户端来说，不需要了解有状态资源标识符(标识符代表有状态资源的身份信息，用来识别有状态资源)的内容，有状态资源标识符只是对被访问的web服务有意义的，由Web服务以一种特殊方式去识别在请求过程中使用的WS-Resource。“模式”是指它们之间的关系是用现有的常规Web服务技术(如XML、WSDL)来约束的。图3.2说明了通过隐性资源模式创建WS-Resource的过程<sup>[26]</sup>。

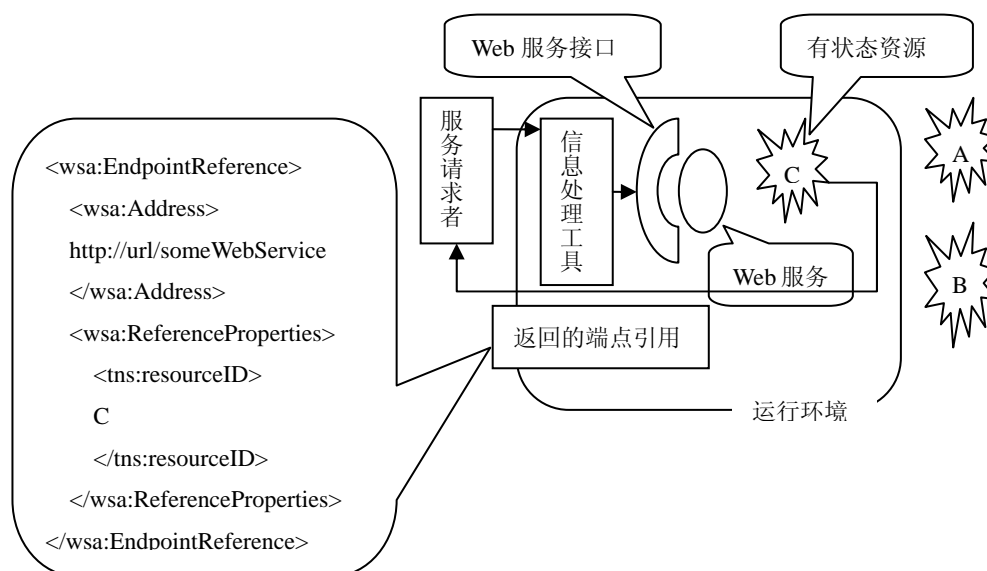


图 3. 2 通过隐性资源模式创建 WS-Resource 的过程

服务请求者发送请求信息给Web服务，由信息处理工具接受请求信息，信息处理工具支持一个或多个网络传输协议，信息处理工具要翻译这些信息并进行封装，封装后的信息能被Web服务识别。然后，传送给Web服务接口，Web服务接收请求信息并创建了一个名为“C”的有状态资源，Web服务返回给服务请求者一个新创建的WS-Resource端点引用，因为这个WS-Resource是由新创建的有状态资源和Web服务合成的，所以把创建有状态资源的Web服务称作WS-Resource制造者。服务请求者通过端点引用来寻址有状态资源，端点引用包含了地址(Address)和引用属性(ReferenceProperties)两部分。地址说明有状态资源的位置(在HTTP协议中，通常是URL)。引用属性描述有状态资源的属性。引用属性包括有状态资源标识符(Identifier)子元素，图3.3说明了端点引用、资源标识符、地址、引用属性间的关系。当服务请求者要访问有状态资源，它所发送的请求信息必须包含端点引用，而且引用属性“隐含”在SOAP的信息头(SOAP header)中，而不是在SOAP的主体信息(SOAP body)中。被访问的web服务从SOAP中析取有状态资源标识符，通过标识符来识别有状态资源。

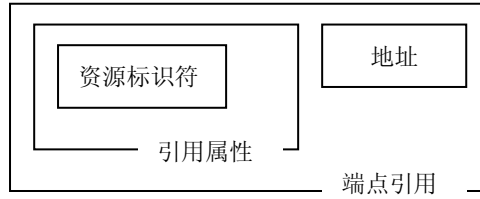


图 3.3 端点引用的结构

由此可见，WS-Addressing 使端点引用机制标准化，从而使端点引用能以规范化的形式表示部署在特定网络端点的 Web 服务。除了 Web 服务的端点地址之外，端点引用还可以包括其他与 Web 服务相关联的元数据，如服务描述信息和引用属性，它们有助于促进 Web 服务地址的规范化使用。引用属性在隐性资源模式中起重要作用。

隐性资源模式遵循 WS-Addressing 的约定，在这里有状态资源被当作 Web 服务消息交换处理过程的隐式输入。端点引用中的引用属性子元素，用来识别在所有利用这个端点引用来执行消息交换的过程中所使用的有状态资源。这种包含有状态资源标识符的端点引用被称为 WS-Resource 有资格的端点引用（WS-Resource-qualified endpoint reference）。那些定向到由 WS-Resource 有资格的端点引用所指定的 Web 服务的请求消息必须包括端点引用及其包含的引用属性信息。

因而，WSRF 使用 WS-Resource 有资格的端点引用表示 WS-Resource 的“网络范围指针”，具有下面几种具体表现形式：WS-Resource 有资格的端点引用可以是向 WS-Resource 工厂请求创建一个新的 Web 服务的返回结果，或者是对服务注册中心搜索查询的返回结果，或者是 Web 服务向某些特定应用程序提出执行的请求。

## 3.2 WS-ResourceLifetime 规范

当一个请求执行完成后，与之相联系的有状态资源应该被销毁，以释放相关的资源。有状态资源并不是无限期存在的，它具有一定的有效期。WS-ResourceLifetime 规范是针对 WS-Resource 生命周期的三个重要方面而制定的，即创建、身份标识和销毁。下面就从这三个方面进行详细阐述。

### 3.2.1 WS-Resource 工厂模式 (WS-Resource Factory Pattern)

WSRF 并不试图去定义请求创建新的 WS-Resource 的消息交换。实际上，它仅仅指出，新的 WS-Resource 可以由某些特殊的机制来创建，或者通过

WS-ResourceLifetime 规范称之为 WS-Resource 工厂 (WS-Resource factory) 的机制来创建 (工厂模式 (Factory Pattern) 通常作为一种常用的方法经常会在创建型模式 (creational pattern) 中出现)。WS-Resource 工厂是任何能够产生一个或者多个 WS-Resource 的 Web 服务。典型地, WS-Resource 工厂操作的激活信息至少包括一个到新的 WS-Resource 的端点引用。

值得注意的是, 网络环境中存在许多种类型的 Web 服务 (如资源注册中心就是一种特殊的 Web 服务), 它们都可以在响应消息中返回 WS-Resource 有资格的端点引用。然而, 并不是所有能返回 WS-Resource 有资格的端点引用的 Web 服务都实现 WS-Resource 工厂的功能。一个消息交换只有引起有资格的端点引用所指定的 WS-Resource 真正被创建时, 才被认为是一个 WS-Resource 工厂操作。

### 3.2.2 WS-Resource 的身份标识 (WS-Resource Identity)

下面将从 WS-Resource 的实现和服务请求者这两个角度来描述和对比 WS-Resource 身份标识的作用和用途。

WS-Resource 的 Web 服务可以通过在 WS-Addressing 端点引用的引用属性中包含的有状态资源标识符来构造 WS-Resource 地址。于是, 这个端点引用被称作是 “WS-Resource-qualified”。此时 WS-Resource 有资格的端点引用对于在分布式系统中的其他实体变得可用, 随后它们就可以利用那个端点引用去引导对 WS-Resource 的请求。逻辑上, 这些请求要经过 WS-Resource 的 Web 服务回应, 因此 Web 服务了解封装在 WS-Addressing 端点引用中实现独立的有状态资源标识符的内容。WS-Addressing 端点引用的一部分能识别服务, 反过来服务利用引用属性来识别在消息执行中所使用的有状态资源。

相反, 有权对 WS-Resource 有资格的端点引用访问的服务请求者不应该检查或者试图解释表示有状态资源标识符的引用属性的内容。甚至服务请求者试图比较两个有状态资源标识符都被认为是无效的。站在服务请求者的角度, WS-Resource 的端点引用的引用属性的内容应该是不对外公布的。

那么, 服务请求者将如何推断出 WS-Resource 的有状态资源的身份标识呢? 一个简单的方法就是, WS-Resource 的有状态资源身份标识的语义含义, 以及定义 WS-Resource 并向请求者公布 WS-Resource 的方式, 是由 WS-Resource 的实现来决定的。WS-Resource 的有状态资源身份标识是否向服务请求者公开是由 WS-Resource 的设计决定的。然而, 我们相信很多 WS-Resource 都有能力获取到这个身份标识。这个身份标识应该是一个可移植的、名称空间作用域受限的值。可移植性很重要, 因为它使一个应用程序可以向另外一个应用程序传送这个身份标识。确定名称空间作用域也很重要, 因为它可以解决因标识相同但来源不同而身份模棱两可的问题。

可以预见，公开 **WS-Resource** 的有状态资源身份标识的通常方法将是把这个身份标识视为一个或者多个在 **WS-Resource** 特性文档中所表示的资源特性。这种方法将使服务请求者能够凭借这个文档来引导查询，使用所理解的特性来表示有状态资源的身份。如果身份标识作为一个或者多个资源特性来被公开，**WS-Resource** 应该确保这些特性只能被读取而不能被改写。典型的情况就是服务请求者改变有状态资源的身份标识是无效的。

### 3.2.3 **WS-Resource** 的销毁 (**WS-Resource Destruction**)

显然，创建新的 **WS-Resource** 的请求者只会在有限的时间内关注那个新创建的 **WS-Resource**。在这段时间之后，销毁 **WS-Resource** 应该成为可能，这样可以回收与它相关联的系统资源或者应用程序资源。**WS-Resource** 采用两种用于销毁 **WS-Resource** 的生命周期管理方法：立即销毁 (**Immediate Destruction**) 和定时销毁 (**Scheduled Destruction**)。

#### 3.2.3.1 立即销毁

在很多场合中，使用 **WS-Resource** 的应用程序请求立即销毁它是适当的。希望立即销毁 **WS-Resource** 的服务请求者必须使用适当的 **WS-Resource** 有资格的端点引用向由该端点引用标识的 **WS-Resource** 发送销毁请求消息。在 **WS-ResourceLifetime** 规范中定义了立即销毁的消息交换方式。

从 **WS-Resource** 那里收到销毁请求消息的回应表明服务请求者和接收销毁请求消息的 **WS-Resource** 之间断开了联系。一旦收到回应消息，在没有其他高优先级错误的条件下，任何进一步与 **WS-Resource** 进行消息交换的尝试必然导致一个“资源未知” (**Unknown Resource**) 的错误消息。

#### 3.2.3.2 定时销毁

请求者可能不愿立即地、同步地销毁 **WS-Resource**，或者可能实际上根本就不允许在一个分布式的计算环境中这样做，因为请求者早已与服务提供者断开连接了。因而，除了能立即销毁 **WS-Resource**，还必须可以在将来的某个特定时间里销毁 **WS-Resource**。利用 **WS-Resource** 有资格的端点引用，服务请求者可以首先建立，随后更新 **WS-Resource** 的销毁时间。当这个时间来临，**WS-Resource** 能自我销毁，而不需要来自于服务请求者的同步的销毁请求。请求者可以按照策略周期性地更新销毁时间以调整基于照策略的生存周期。**WS-ResourceLifetime** 规范定义了标准的消息交换方式，使服务请求者能够先建立、随后改变 **WS-Resource** 的销毁时间，而且还定义了获取 **WS-Resource** 当前设定的销毁时间的方法。

在创建 **WS-Resource** 的时候，**WS-Resource** 工厂可以协商确定 **WS-Resource** 初始的销毁时间。随后，经授权的服务请求者可以通过向 **WS-Resource** 发送请求消息，利用 **WS-Resource** 有资格的端点引用来请求改变这个初始设定的销毁时间。

如果到达该时间，WS-Resource 就可以被销毁，而任何与之相关联的系统资源都将被回收。

WS-Resource 的销毁时间可以非单调性地改变，也就是说，服务请求者重新设定的销毁时间可以比已设定的销毁时间还早。如果所请求的终止时间在当前时间以前，这个请求必须被解释成立立即但不同步地销毁 WS-Resource。

### 3.3 Web 服务资源的特性 (WS-ResourceProperties)

WS-ResourceProperties 规范定义了 WS-Resource 状态的类型和值，服务请求者可以通过 Web 服务接口查看和修改它。WS-ResourceProperties 的关键思路有：

(1) Web 服务资源具有一个使用 XML schema 定义的 XML 资源特性文档 (XML resource property document)。

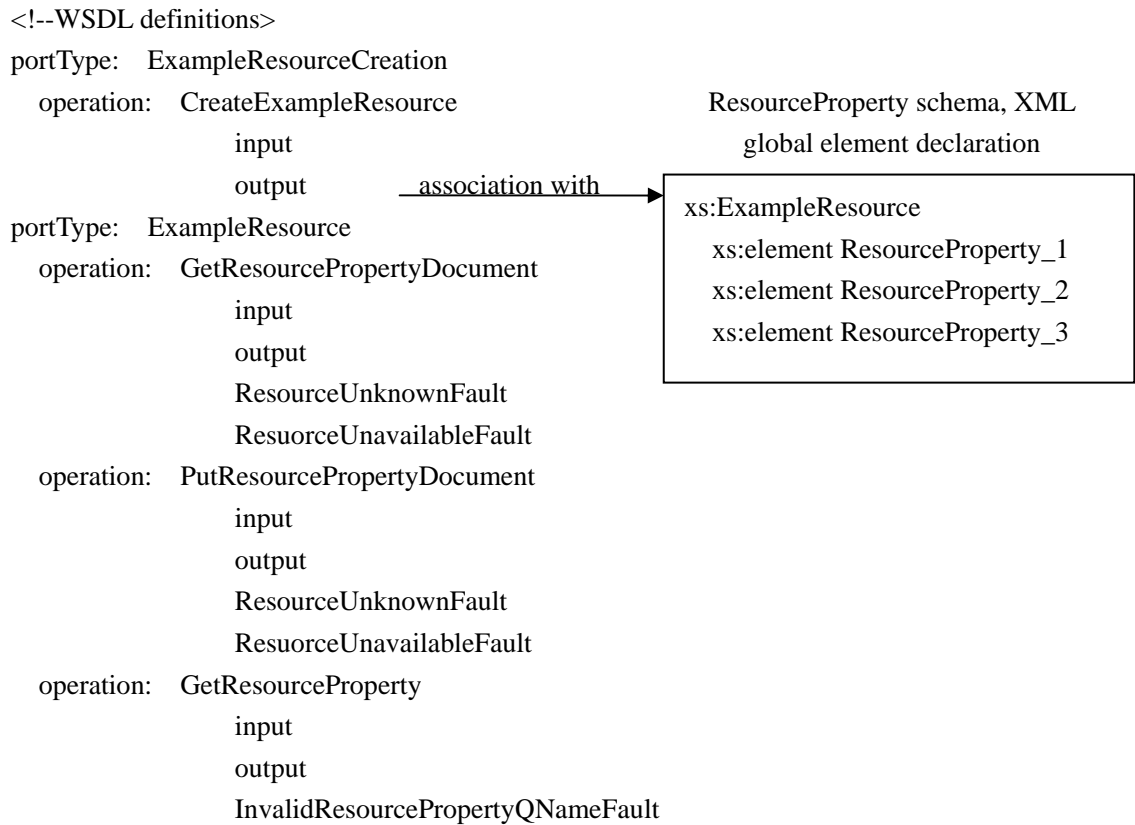
(2) 服务请求者可以通过标准的方法搜索 WSDL portType 定义来确定 WS-Resource 的类型。

(3) 服务请求者可以使用 Web 服务消息交换来读取、修改和查询表示 WS-Resource 状态的 XML 文档。

资源特性 (resource property) 是指表征 WS-Resource 状态的个体组件。描述 WS-Resource 中有状态资源类型的 XML 文档是 WS-Resource 特性文档。每个资源特性都被表示成在 Web 服务资源特性文档中的一个 XML 元素。XML 的使用是逻辑上的，而物理上底层的真实状态可能具有一种或者多种格式，而且位于一个或者多个地点。

#### 3.3.1 WS-Resource 的特性文档

WS-Resource 特性文档可以看作是 WS-Resource 真实状态的视图。在这个文档定义的结构之上由服务请求者发起的查询和更新消息能够被正确引导。任何通过 WS-Resource 特性文档来处理资源特性的操作都必须在实际的 WS-Resource 实现中得到反映。一个完整的 WS-Resource 特性文档结构定义如下<sup>[27]</sup>：



WS-Resource 特性文档是使用 XML Schema 描述的。特别地，在某些 XML 名称空间中，WS-Resource 特性文档被表示为 XML 全局元素声明（GED，XML global element declaration），其中包含一组个体资源特性的 XML GED 的引用。例如，考虑一个被称为“C”的有状态资源，如果“C”的状态包含三个资源特性，分别称为 p1、p2 和 p3，那么它的资源特性文档，被称为“ExampleResourceProperties”，定义如下：

```

<xs:schema
targetNamespace=http://example.com/ResourcePropertiesExample
xmlns:tns="http://example.com/ResourcePropertiesExample"
xmlns:xs="http://www.w3.org/2001/XMLSchema" ...>
  <xs:element name="p1" type= ... />
  <xs:element name="p2" type= .../>
  <xs:element name="p3" type= ... />
  <xs:element name="ExampleResourceProperties">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:p1" />
        <xs:element ref="tns:p2" />
        <xs:element ref="tns:p3" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

.....

</xs:schema>

服务请求者可以通过各种方法（包括由WS-MetaDataExchange<sup>[28]</sup>定义的消息交换）来获得和检验WS-Resource特性文档中表示有状态资源“C”的类型的XML schema定义。

但是服务请求者是如何知道被称为“ExampleResourceProperties”的 GED 定义了 WS-Resource 的资源特性文档呢？WS-Resource 的资源特性文档声明会出现在它的 Web 服务组件的 WSDL portType 定义中。WS-Resource 特性文档声明通过使用 ResourceProperties 属性来与 WSDL portType 定义相关联，如下面的示例：

<wsdl:definitions

targetNamespace="http://example.com/ResourcePropertiesExample"

xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

xmlns:xs="http://www.w3.org/2001/XMLSchema"

xmlns:wsrp=

"http://www.ibm.com/xmlns/stdwip/web-services/ws-resourceProperties"

xmlns:tns="http://example.com/ResourcePropertiesExample".....>

.....

<wsdl:types>

<xs:schema>

<xs:import

namespace="http://example.com/ResourcePropertiesExample"

schemaLocation="....."/>

</xs:schema>

</wsdl:types>

.....

<wsdl:portType name="SomePortTypeName"

**wsrp:ResourceProperties="tns:ExampleResourceProperties" >**

<operation name="....."/>

.....

</wsdl:portType>

.....

</wsdl:definitions>

### 3.3.2 WS-Resource 的特性组合

与各种各样涉及 WS-Resource 的规范相关联的 Web 服务接口已经被设计成是可组合的。在 WSDL 1.1 中，Web 服务接口的设计者能通过“复制-粘贴”方式将众多由 portType 定义的操作组成接口。

除了操作组合以外，设计者也可以聚合 WS-Resource 特性文档中定义的 WS-Resource 特性，来产生由最终的 portType 声明的 WS-Resource 特性文档。设计者可以使用 XML 文档组合中的任何方法来定义最终的文档，示例中包括扩展

和聚合。正如下面的示例所展示的，利用 `xs:ref`，`WS-Resource` 特性文档组合可以通过添加（或称作聚合）附加的 XML 元素声明来完成。

```
<xs:element name="ExampleResourceProperties">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tns:p1" />
      <xs:element ref="tns:p2" />
      <xs:element ref="tns:p3" />
      <xs:element ref="xxxx:SomeAdditionalResourceProperty"
        xmlns:xxxx= ... />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

从示例中看到，通过有状态资源“C”的特性文档的资源特性元素与其他某个名称空间中定义的一个资源特性元素（`SomeAdditionalResourceProperty`）的结合，形成了这个组合后的 `WS-Resource` 特性文档。

### 3.3.3 访问 `WS-Resource` 特性的值

`WS-Resource` 的状态，即在 `WS-Resource` 的资源特性文档中所公开的资源特性的值，可以通过标准的 Web 服务消息来读取、修改和查询。这些消息交换在 `WS-ResourceProperties` 规范中被定义，并且应该作为 WSDL 操作包含在任何使用 `wsrp:ResourceProperties` 属性的 `portType` 中，用来声明 `WS-Resource` 的特性文档。

访问 `WS-Resource` 特性的基本功能是使用简单的 Web 服务请求/响应消息交换来获取单个资源特性，正如先前所描述的那样，使用 `WS-Resource` 有资格的端点引用识别 `WS-Resource`，然后通过它的 GED 的 qualified name 来识别资源特性。这种获取功能如果变得稍微复杂一些的话，将允许使用单一的请求/响应消息交换获取多重资源特性的值。

获取特性值的功能在 `WS-ResourceProperties` 中被定义为“get”操作，正如“get”请求消息中所规定的，它允许请求者检索一个或多个 `WS-Resource` 特性的值，如下所示：

```
<wsrp:GetMultipleResourceProperties>
  <wsrp:ResourceProperty>QName<wsrp:ResourceProperty>
</wsrp:GetMultipleResourceProperties>
```

根据 QName 指定的 `WS-Resource` 的特性值，请求消息的响应将是一个 XML 元素序列。例如，下面的请求消息表示一个从有状态资源（比如“C”）中检索特性“p1”的值。

```
<soap:Envelope>
  <soap:Header>
```

```

    <tns:resourceID> C </tns:resourceID>
</soap:Header>
<soap:Body>
  <wsrp:GetMultipleResourceProperty>
    <wsrp:ResourceProperty>tns:p1</wsrp:ResourceProperty>
  </wsrp:GetMultipleResourceProperty>
</soap:Body>
</soap:Envelope>

```

从上面的代码可以看到，指定请求消息目标的端点引用在 `resourceID` 中包含一个标识符，它可以识别称为“C”的有状态资源。在本示例中，这个标识符信息位于 SOAP 信息头中，它是采用 WS-Addressing 端点引用 `ReferenceProperties` 的标准进行编码。

WS-Resource 将使用一个包含特性值 `p1` 的消息做出响应，如下所示：

```

<soap:Envelope>
  <soap:Body>
    <wsrp:GetMultipleResourcePropertyResponse>
      <p1>xyz</p1>
    </wsrp:GetMultipleResourcePropertyResponse>
  </soap:Body>
</soap:Envelope>

```

WS-ResourceProperties 规范同样也定义了“set”操作，使 WS-Resource 特性能够被插入、更新和删除。下面简要地给出“set”操作请求消息的伪语法（pseudo-syntax）。

```

<wsrp:SetResourceProperties>
{
  <wsrp:Insert >
    {any}*
  </wsrp:Insert> |
  <wsrp:Update >
    {any}*
  </wsrp:Update> |
  <wsrp>Delete ResourceProperty="QName" />
}+
</wsrp:SetResourceProperties>

```

在 WS-ResourceProperties 规范中定义的第四个操作允许服务请求者依靠 WS-Resource 特性文档来执行查询表达式（比如一个 XPath 1.0 查询表达式）。例如，可以使用各种各样的查询表达式类型来支持基于 WS-Resource 状态当前值的资源发现。

最后，WS-ResourceProperties 规范也定义了一些方法，服务请求者使用这些方法可以订阅 WS-Resource 资源特性值改变的通知。这是通过 WS-Notification 规范家族中所定义的功能而实现的。

### 3.4 可更新的引用 (WS-RenewableReferences)

WSRF 必须定义一些机制能够更新已经无效的端点引用。这些机制要能够应用于任何端点引用，对于指向 WS-Resource 的端点引用尤为重要，因为它能够提供持久的、稳定的 WS-Resource 引用，能够允许同一状态随着时间的推移被不断访问。

WS-Addressing 端点引用不仅包含寻址，而且还包含关于如何与服务交互的策略信息。典型的情况下，端点引用由关于寻址和策略的权威信息源构造。当一个端点引用对客户端有效，表明 Web 服务的寻址和策略信息适用于该客户。但是有些情况下这些信息在付诸实现时不能够从头至尾贯彻始终，因为权威的寻址和策略信息源会随着 Web 服务的发展而变化。在这种情况下，能够更新端点引用显得非常重要。

WS-RenewableReferences 还定义了一个特殊的 WS-Policy<sup>[29]</sup> 声明，是为了当端点引用失效时，使端点引用具有足够必需的信息来重新形成一个新的有效的端点引用。

### 3.5 服务小组 (WS-ServiceGroup)

WS-ServiceGroup 规范定义了一种方法用以表示和管理异构的、通过引用聚集在一起的 Web 服务集合。这个规范可以被用来组织 Web 服务集合，例如用来构建 Web 服务注册中心，或者用来构建能够对一组 WS-Resource 实现共同操作的 Web 服务。

WS-ServiceGroup 规范可以使用 WS-ResourceProperties 的资源特性模型来表示服务小组成员资格规则、成员资格约束以及分类。小组成员要能符合由资源特性描述的小组约束。WS-ServiceGroup 规范也定义了管理 Web 服务小组成员资格的接口。

WS-ServiceGroup 定义的接口多是由其它的 Web 服务接口构成，因为这些其他的 Web 服务接口已经就具备专门与服务小组和/或与属于该小组的 Web 服务成员进行交互的能力。例如，专门的查询接口可以被用来查询服务小组的内容，还可以对服务小组的成员进行集体的查询操作。

### 3.6 基本错误 (WS-BaseFaults)

WS-BaseFaults 规范定义当 Web 服务消息交换返回错误的时候所使用的基本错误类型。虽然在这个规范没有对 WS-Resource 提出特殊的要求，但是 WSRF 框

架中其他所有的规范都使用它以保证这些规范中的操作所返回错误是一致的，以及涉及 WS-Resource 定义和使用的错误报告是一致的。

### 3.7 通知 (WS-Notification)

WS-Notification 并不是 WSRF 规范体系中的一员，但是作为 WSRF 的补充，WS-Notification 通常被视作 WSRF 在订阅/发布通知方面不可或缺的重要支撑部分，在实现时 WS-Notification 总是与 WSRF 同时出现。同 WSRF 一样，WS-Notification 也是一个规范家族，它定义了一个基于主题的 Web 服务系统，用于发布和订阅以 WSRF 为基础的信息交互。WS-Notification 所采用的基本方法是定义一些机制和接口，以允许客户端订阅感兴趣的主体，比如某个 WS-Resource 资源特性值的改变。从 WS-Notification 的角度来看，WSRF 为描述和构建通知机制提供了必要的基本条件。从 WSRF 的角度来看，WS-Notification 规范家族能够异步地向请求者通报资源特性值的改变，从而扩展了 WSRF 的效用。

其中一个 WS-Notification 子规范，WS-BaseNotification，描述了订阅者如何向通知生产者注册，表明希望接收通知消息的方法，定义了订阅者所必需的角色、概念和模式。通知可以涉及很多方面，如资源特性值的改变，通知生产者的内部状态的改变，还有环境中某些其他的情况。订阅者通过发出“订阅”消息，指示通知使用者的地址、预订的主题和其他的相关预订信息，来注册希望收到关于一个或者多个主题的通知。在响应中，通知生产者创建一个预订 WS-Resource，并通过这个预订 WS-Resource 建立预订者和通知者的联系，再将这个预订 WS-Resource 有资格的端点引用返回给预订者。最后通知生产者发布一个符合这个主题的通知信息，通知生产者发送通知信息给通知获得者。这种模式称为通知的直接发布方式，整个过程如图 3.4 所示。

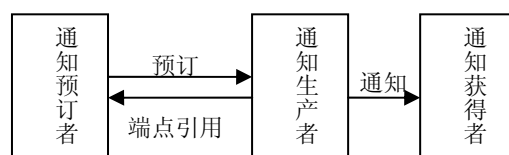


图 3.4 直接发布通知方式

第二个 WS-Notification 子规范，WS-BrokeredNotification，定义了通知代理的接口，规范了发布、订阅、提交通知的信息交换，变化和事件可以通过代理来发布。在有代理的情况下，通知代理起到一个中间桥梁的作用，代替了直接通知模式下的通知生产者，使用通知代理的目的是用来代替和最终通知使用者的互操作。首先，预定者发送标题请求信息给通知代理，通知代理创建一个预定资源，并返

回这个预定资源端点引用给预定者。通知发布者向通知代理请求发布，然后，通知代理传递通知信息给任何符合这个预定的通知获得者。交换通知信息有两个作用：发布通知信息给代理和发布通知信息给通知获得者。这种模式称为通知的代理发布方式，整个过程如图3.5所示。

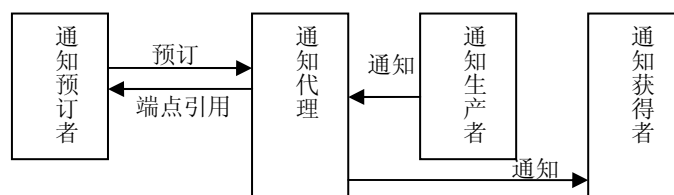


图 3.5 代理发布通知方式

第三个 WS-Notification 子规范，WS-Topics，给出通知主题的 XML 描述和相关的元数据。WS-Topics 是组织通知消息的机制，以便订阅者能够方便地了解可以订阅哪些类型的通知。主题可以按层次来组织，一个主题还可以进一步分解成若干子主题。主题同样也由名称空间划定其适用范围。

## 4 WSRF 与 OGS I、WSMF 的比较分析

### 4.1 WSRF 与 OGS I 比较研究

作为OGSI对新的Web服务标准特别是WS-Addressing的重构和发展，WSRF与OGSI在整体框架、功能划分、功能描述等方面都十分相似，但WSRF也根据Web服务的实际状况和需求作了一些变动和改进，克服了OGSI的一些不足，从整体上看WSRF更能满足Web服务发展到需要。

#### 4.1.1 相同点

WSRF和OGSI都是描述服务和资源的一种基础框架，两者都注重如何通过Web服务接口管理有状态资源。尽管WSRF和OGSI对有状态资源进行建模的方法不同，后者将有状态资源建模为网格服务，而前者则是将有状态资源同Web服务建模为WS-Resource，但它们采取的方法在本质上是一样的，并且两者使用的WSDL接口定义在语法上甚至在语义上也很相似。另外，在WSRF和OGSI中，用来创建、寻址、观测、操作、销毁网格服务和WS-Resource的方法存在本质上也是相同的。

#### 4.1.2 不同点

在对服务和资源的标识问题上，WSRF和OGSI采用了不同的处理方式。OGSI中需要有GSH、GSR来标识和访问服务实例，OGSI实现的每个服务都具有一个或多个基于URI模式的GSH来唯一地标识，但由于GSH中并没有足够的信息可以让客户直接与服务实例通信，就引入了句柄解析器机制来解决抽象名字与具体地址之间的映射问题，使得GSH被解析成GSRL来允许客户通过GSR的信息来访问服务实例。因此GSH、GSR和句柄解析器在相互依赖的机制中提供了各自不同的功能。而WSRF为这些功能定义了一个框架并提供相互独立的机制。WSRF中，WS-Addressing采用端点引用来标识WS-Resource，随后又根据隐性资源模式从端点引用中获取与某个Web服务有关的有状态资源的标识符。因此，OGSI中的GSH与端点引用相似，而GSR则与特定的有状态资源的标识符相似。另外，WSRF的WS-RenewableReferences规范还将一个引用更新策略与某个端点引用相关联，这是OGSI中所没有的。

在生命周期的管理问题上，首先WSRF和OGSI对“实体”的理解就不同。WSRF认为实体是Web服务与有状态资源组合的WS-Resource，OGSI则认为实体是网格服务实例。对任何实体的生命周期管理都会涉及到实体创建、身份标识分配、实体销毁等内容，WSRF和OGSI也不例外，表4.1对两者在生命周期管理方面进行了比较<sup>[30]</sup>。

表 4.1 WSRF 和 OGSI 在生命周期管理方面的比较

功能	OGSI	WSRF
创建新实体	工厂 portType: createService	工厂模式
对实体寻址	网格服务引用和网格服务句柄	带引用特性的 Web 服务寻址端点引用
直接销毁实体	网格服务 portType: destroy	资源生命周期 portType: Destroy
定时销毁实体	网格服务 portType : requestTerminationAfter 和 requestTerminationBefore	资源生命周期 portType : SetTerminationTime, 相当于 OGSI 的 requestTerminationAfter
确定当前时间	网格服务 portType 服务数据元素: currentTime	资源特性: CurrentTime
确定生命周期	网格服务 portType 服务数据元素: terminationTime	资源特性: TerminationTime
销毁实体通知	无	订阅通知: ResourceDestruction

对于实体的创建OGSI和WSRF的方法有所不同：在OGSI中，用户可通过调用一个实现某portType的服务实例上的createService操作来要求创建一个网格服务实例。为了支持状态生命期管理，用户可指定可接受的最早和最晚的终止时间。工厂在前面指定的时间窗口中选择一个初始终止时间并作为它对创建请求响应的一

部分返回给用户。createService操作中需要输入两个可选的参数TerminationTime和CreationParameters，其中TerminationTime代表了用户可接受的网格服务实例的最早和最晚的终止时间，CreationParameters包含提交给工厂的创建服务实例的必要信息。操作结束后，输出指向新创建网格服务实例的Locator，给出新创建的服务当前拟定的终止时间CurrentTerminationTime以及专用于工厂和它所创建的服务的XML扩展性元素ExtensibilityOutput。而在WSRF中，WSRF仅仅定义了代表了一种Web服务的工厂模式，该Web服务可以创建WS-Resource，可以返回多个新的WS-Resource的端点引用。WSRF中的工厂模式和OGSI中的工厂操作提供的功能大致相同，而区别主要体现在具体实现操作上。

对于实体的销毁，在OGSI中，用户要求销毁一个网格服务实例可以向服务实例的外部销毁操作提出用户调用请求(网格服务的portType提供“destroy”操作供用户调用)，或者可以通过一个软状态方法，如果某个网格服务实例的生命周期结束而未收到来自用户的延时要求，那么驻留该服务实例的服务器销毁该服务实例，回收相关资源，并从其控制下的句柄解析器删除该服务实例的所有信息。而在WSRF中，对WS-Resource的销毁分为直接销毁和定时销毁两种，多数情况下是利用应用程序请求立即销毁WS-Resource。希望立即销毁的请求者必须使用适当的WS-Resource有资格的端点引向由该端点引用标识的WS-Resource发送销毁请求消息，端点引用内部的有状态资源标识符被用来识别将被销毁的有状态资源。一旦收到响应消息，在没有其他高优先级错误的条件下，任何进一步与WS-Resource进行消息交换的尝试必然导致一个“资源未知”的错误消息。当销毁操作正确返回时，表明了该资源已被销毁，不能再通过服务被访问到。这一点有别于OGSI，因为在OGSI中，当返回的消息表明销毁操作正确执行时，仅仅表明网格服务实例的销毁动作已经被正常启动，销毁行为刚刚开始。

在服务小组的管理问题上，OGSI的服务组是一个网络服务实例，用以保持与一组网格服务有关的信息。典型的网格服务注册由从网格服务基本行为扩展而来的portType定义，该portType向服务组提供接口：ServiceGroup，ServiceGroupEntry和ServiceGroupRegistration。而在WSRF中，服务组是一组WS-Resource的组合，组成员都必须遵循由资源特性描述的小组约束。服务组规范定义了管理服务组成员资格的接口，这些接口多是由其它的Web服务接口构成，因为这些其他的Web服务接口已经就具备专门与服务小组和/或与属于该小组的Web服务成员进行交互的能力。例如，专门的查询接口可以被用来查询服务小组的内容，还可以对服务小组的成员进行集体的查询操作。OGSI服务组的portType和WSRF服务组规范中的相应接口所起的作用是一样的，但是两者最明显的不同在于：OGSI的portType:ServiceGroupRegistration中有“remove”这个操作，而WSRF服务小组规范中没有。

在错误处理的问题上，OGSI为所有网格服务必须返回的错误信息定义了基本的XML schema以及错误信息的相关语义，即通过创建一个包含所有错误信息的公共基础信息集来解决错误处理的问题。它并没有改变WSDL错误信息模型，所有的OGSI错误都是从ogsi: FaultType这个错误基类继承而来。在WSRF中，WS-BaseFaults规范也是定义了一个Web服务信息交换返回错误时所使用的基本错误类型。虽然在这个规范没有对WS-Resource提出特殊的要求，但是WSRF框架中其他所有的规范都使用它以保证这些规范中的操作所返回错误是一致的，以及涉及WS-Resource定义和使用的错误报告是一致的。

总的来说，从整体上看，由于WSRF是OGSI的重构和发展，两者的总体设计思路十分相近，只是在一些个别问题的具体处理上会有些差异。表4.2将WSRF和OGSI进行比较全面的比较<sup>[14]</sup>，从中可以发现一些细微而具体的不同之处。

表 4.2 WSRF 和 OGSI 的全面比较

OGSI	WSRF	讨论
Grid Service Reference	WS-Addressing Endpoint Reference	WSRF 采用了 Web Services 新的寻址规范 WS-Resource
Grid Service Handle	WS-Addressing Endpoint Reference & WS-RenewableReferences	网格服务的“Handle”和“Handle Resolver”被整合成端点引用的一部分，而且政策评注加强端点引用的稳定性
HandleResolver portType	WS-RenewableReferences	“Handle Resolver services reference”被整合到端点引用中
Service data definition	Resource properties definition	WSRF 更好地遵循 XML Schema，与 WSDL 1.1 兼容
GridService portType service data access	WS-ResourceProperties	WSRF 以多重标准操作代替 OGSI 对基本操作的扩展
GridService portType lifetime management	WS-ResourceLifetime	WSRF 去掉多余的“terminatebefore”
Notification portTypes	WS-Notification	WSRF 使用标准的 WS-Notification
Factory portType	Factory pattern	WSRF 采用工厂模式，不用再另外定义一些特殊的操作
ServiceGroup portType	WS-ServiceGroup	变化不大
Base fault type	WS-BaseFault	变化不大
GWSDL	Cute-and-paste	WSRF 使用 WSDL 1.1 的接口组合方法

#### 4.1.3 WSRF 对 OGSI 的改进

在本文的2.1.2部分谈到了OGSI被Web服务领域指出了四点不足。虽然第四个不足随着时间的推移应该能被克服，但是前三个不足一直是OGSI所无法克服的。针对这三个不足，WSRF从设计时就充分考虑产生这些不足的原因，并逐一克服，

实现了对OGSI的改进。

首先在对两者的理解和掌握方面，OGSI规范太过庞大，这使得读者很难完全了解掌握它的全部内容，也无法很好地理解和分清其中每个部分的内容和特点以及各自起到什么作用。而WSRF更易于理解和掌握，它将OGSI中的各个功能进行划分以便更灵活地进行组合，而且它还更直接地将信息处理从有状态资源中分离。所以WSRF的概念变得很清晰，它的目的就是要通过Web服务的操作实现对有状态资源的管理。

其次在与现有的Web服务和XML工具协同工作方面，OGSI不能很好地与现有的Web服务和XML工具协同工作，它引入了一个可扩展操作的概念，该操作是通过使用XML schema的xsd: any来引入无类型的参数。虽然通过允许网格服务操作使用“any”参数，使得网格服务操作拥有最大的灵活性，但这会导致诸多问题，如在J2EE平台中Web服务核心技术——JAX-RPC中，客户机端对其可以传递给服务什么种类的参数产生混淆。针对这个问题WSRF做了改进。WSRF更规范地利用现有的XML schema，同时也利用新的Web服务标准如WS-Addressing等，从而WSRF更容易使用Web服务工具到达实现的目的，也更容易在许多已定义Web服务接口中得到应用。

最后在如何处理Web服务和有状态资源之间关系这个方面，OGSI在塑造有状态资源为Web服务时，封装了资源的状态、标识、状态、生命周期等，面向对象的思想过于严重，导致一些Web服务不能动态地创建或删除。但事实上，Web服务领域一直认为，纯Web服务是无状态的，而且是实例化的而非抽象的。WSRF一直秉承这个思想，更合理地处理Web服务和有状态资源的关系，将服务和有状态资源明确地区分开来，又利用隐性资源模式将两者联系在一起。

## 4.2 WSRF 与 WSMF 比较研究

Web Services向语义方面发展已经是不争的事实。2005年8月W3C就有工作组提出要将WSRF进行语义扩展（WSRF-S），并开始进行草案的设计工作，其目的就是力图从基础框架层面着手推动Web Services接纳语义网技术，同时促进语义网在Web Services中的应用和发展。除了WSRF目前在向语义性进行扩展外，我们不得不提及Web Services领域中早已关注语义性数年的另一个Web Services基础框架规范——WSMF（Web Services Modeling Framework）。

WSMF在2002年由Dieter Fensel和Christoph Bussler提出，它致力于为Web服务的描述和开发提供一套完备的概念模型<sup>[31]</sup>，增强Web服务的语义性，使得Web服务成为计算机可以理解的实体，支持服务的自动发现、执行和组合，实现语义网和Web服务的融合。WSMF主要包含3个部分，WSMO（Web Services Modeling

Ontology)、WSMX (Web Services Modeling Execution Environment) 和WSML (Web Services Modeling Language)。它强调两个设计原则<sup>[31]</sup>：第一个是Web服务的各组成部分具有严格的非耦合关系；第二个则作为第一个的补充，要求利用中介协调各部分的交互。因此，WSMF将Web服务过程中涉及到的实体定义成4个顶层要素：本体 (Ontologies)、目标 (Goals)、Web服务 (Web Services) 和中介 (Mediators)。作为WSMF这个大家族的一个重要部分，WSMO在WSMF的基础上，主要对这4个要素做更精细、完备的定义和描述，并对其进行扩展，目的就是为WSMF及Web服务提供基于本体的实体描述，为Web服务的语义操作打下基础。另一个重要组成部分WSMX则为WSMF创建一个Web服务运行环境，它被WSMX工作组视为供参考的语义Web服务执行框架。在WSMX构架的运行环境中，不仅有与服务请求方、提供方通信的接口，有存储WSMO四元素、推理规则和其他实体的知识库，更重要的是，还有与WSMO相适应的建模工具包和负责Web服务自动发现、执行、组合等关键应用的核心管理器。第三个组成部分WSML是描述WSMO四要素的形式化建模语言，发布了3个版本，即WSML-Core、WSML-DL和WSML-Flight。WSMF家族的3个成员WSMO、WSMX和WSML的关系如图4.1所示。

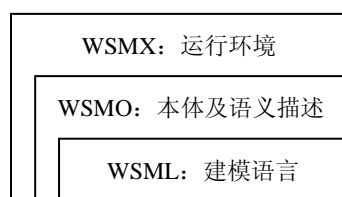


图 4.1 WSMF 家族关系

#### 4.2.1 相同点

WSRF和WSMF都是在Web服务的大环境中提出的规范框架，因此两者的落脚点都是Web服务，它们脱离不了Web服务的各个方面。首先，Web服务环境中存在大量异构而自治的系统，创建WSRF和WSMF的最初动机就是为了整合这些异构而自治的系统从而以一种统一的方式提供服务。其次，在提出WSRF和WSMF之前，Web服务技术已经发展到一定阶段，一些成熟的标准、协议和技术如SOAP、WSDL、UDDI、BPM<sup>[32]</sup> (Business Process Management)、分布式技术 (CORBA和DCOM)、Grid、Semantic Web等都是WSRF和WSMF得以实现的基础。再者，WSRF和WSMF都是对Web服务在某一方面的扩展和充实，WSRF就是Web服务与网格技术的融合，WSMF则是Web服务与语义网技术的融合。另外，Web服务体系对WSRF和WSMF的功能及非功能需求大体上是一致的。在功能需求上，两者都得支持同步和异步消息交换，异构和自治系统整合，还有服务的发布、配置、发现和调用，数据的匹配和过程的协调。在非功能需求上，两者都必须对服务请

求方透明，而且具有充分的有效性、安全性、可靠性，保证网络资源和服务最大限度地利用，网络交互过程能顺利进行<sup>[33]</sup>。

#### 4.2.2 不同点

虽然 WSRF 和 WSMF 都是 Web 服务体系框架中的重要成员之一，都是围绕 Web 服务这个核心，处理与 Web 服务相关的若干问题，但是两者在设计思路和具体细节上仍存在很多差异。本小节将围绕目的、重点、Resource 的含义、耦合性、异构处理和运行环境等 6 个方面进行详细讨论。

##### 4.2.2.1 目的

WSRF是为在网格环境中引入Web服务概念并构建Web服务体系提供一个底层支撑平台。网格是提供数据运算和数据管理的网络基础设施，支持人员同接入网格的海量数据、各种工具和计算资源的交互与合作<sup>[34]</sup>。网格中最常见的问题有：海量的科学数据和工程数据的计算非常困难；这些数据与网格中计算能力的整合问题不易解决；复杂网络环境中资源的共享和本地虚拟化难以实现。网格技术一个重要的特点是将网格计算能力视作有状态的网格服务，其目的在于网格服务能同海量数据和科学运算工具一起都被视为有状态的网格资源。基于这个特点，网格引入了“服务”这个概念，体系结构也遵循SOA（Service Oriented Architecture）基本的设计原则，WSRF应运而生。WSRF中WS-Resource这个概念很好的整合了资源（尤指网格资源）和Web服务这两个概念，并围绕WS-Resource这个核心，WSRF提出了一系列标准规范来描述WS-Resource的各个方面（寻址、创建与销毁、通讯等），进而WSRF替代了OGSI成为网格与Web服务结合后新的底层支撑平台。

WSMF旨在将语义Web技术引入Web服务体系中，通过创建本体（尤其是领域本体）描述Web服务的各个方面，增强Web服务的语义性，形成语义Web服务<sup>[35]</sup>（SWS, Semantic Web Services）。目前Web服务更多的是在语法层面上采用一致的描述语言描述Web服务及其通信和处理流程，但是在领域之间，甚至在某一领域内异构性大量存在，仅仅局限在语法层面上的一致无法处理异构性带来的人与人、计算机与计算机、人与计算机之间在理解层面上的问题。为了解决上述问题，WSMF提供了一套语义Web和Web服务结合的标准规范，最大的特点就是能够考虑特定的Web服务应用领域（如电子商务，电子政务等），为Web服务的各个层面提供相应的领域本体描述。

##### 4.2.2.2 强调的重点

WSRF始终强调的一个关键词是stateful和state，而WSMF则强调关键词semantic和ontology。OGSI视海量数据和计算能力为网格服务，这些服务都是有状态的（stateful），进而OGSI规定了这些有状态服务的描述、发现、调用和通信等方法。然而Web服务规定服务是无状态的（stateless），为了与Web服务机制融合，

Globus Alliance定义了一种特殊的服务,这种服务自身是无状态的,但它可以获取、操作有状态的资源(此时的资源包括有状态的网格服务)。基于此定义,WSRF特别强调资源是有状态的,即资源须有特定的XML描述的状态数据集,须有明确定义的生命周期,以及能被若干Web服务访问和操作<sup>[9]</sup>。同样基于此,WSRF也强调Web服务是无状态的,因为Web服务只需通过标识符操作资源,而对资源状态的管理则交由资源所在的系统(如数据库、文件管理系统等)负责。这样,网格中的有状态和Web服务的无状态在WSRF中全部体现出来。

作为WSMF家族的一个组成部分,WSMO是对WSMF已有内容的精炼和扩展,它包含的与本体有关的一系列Web服务建模规范已成为语义Web服务的核心成分。目前本体已成为构建语义Web强大而成熟的工具,本体库是语义Web重要的构成部分<sup>[36]</sup>。WSMO就是通过构建领域本体,并以此为核心来描述和联系语义Web服务的各个方面<sup>[35]</sup>。在WSMO的8条设计原则<sup>[37]</sup>中有3条与语义或本体相关:

(1) WSMO使用本体作为数据模型,也就是说Web服务中使用到的数据、资源都是由本体描述的,从而在数据层体现Web服务的语义性;(2) 请求者的需求和满足该需求的Web服务在语义上独立,也就是说,相对于Web服务的功能,请求者的需求可以视作Web服务应实现的目标,两者应该应用本体分别描述;(3) WSMO描述的语义要能在一定的环境(如WSMX)中被正确理解进而被贯彻执行。可见,不论是WSMO还是WSMX,WSMF中各个环节都紧紧围绕语义、本体,实现Web服务和语义Web融合这一目标。

#### 4.2.2.3 Resource 的含义

WSRF对Resource的理解可以从WS-Resource这个概念来解读。网格没有截然区分网格服务和网格资源,认为两者都是有状态的。为了建立有状态的Web服务模型,换句话说,为了在无状态的Web服务中引入有状态的概念从而与网格融合,WSRF定义了WS-Resource,即WS-Resource有生命周期,能被创建和销毁,它由Web服务和有状态的资源组成,其中有状态的资源在Web服务的消息交互中被访问和操作,并且通过隐性资源模式将Web服务和有状态资源联系起来<sup>[9]</sup>,如图4.2所示。WSRF的五大标准规范中有两条(WS-ResourceProperties、WS-ResourceLifetime)直接与WS-Resource的描述密切相关,可见WS-Resource是WSRF在Web服务和网格融合的新环境中对Resource这个概念做的扩充,对Resource的一个全新的解读。

WSMF中Resource的概念特指WSMO的4个组成部分:本体、目标、Web服务和中介,每一部分都可以视为一种资源。因此,与WSRF不同的是,针对任何一次具体的Web服务过程,WSMF都将资源分成4类,并分别存放在WSMX中的4个知识库中:本体知识库、目标知识库、Web服务知识库和中介知识库。之所以将资源如此分类,主要是为了遵循WSMF的设计原则之一:框架内的各部

分之间表现强烈的非耦合关系（这一点将在下一小节内讨论）。4 个知识库将在 WSMX 中由资源管理器和资源登记器统一进行管理，如图 4.2 所示。资源管理器只负责维护资源的引用指针，资源实体则交由资源登记器管理，也就是说，任何一次对资源的获取、修改等操作都必须先通过资源管理器得到资源实体的引用指针，然后在通过资源登记器对知识库中的资源实体实现具体的操作。另外，WSMF 还定义了一种特殊的资源，这种资源是在 Web 服务进程中即时生成的有状态的数据，这些数据可能与服务进程有关，可能是 WSMO 四种资源生成的临时数据，相应地都由 WSMX 的数据知识库进行管理。

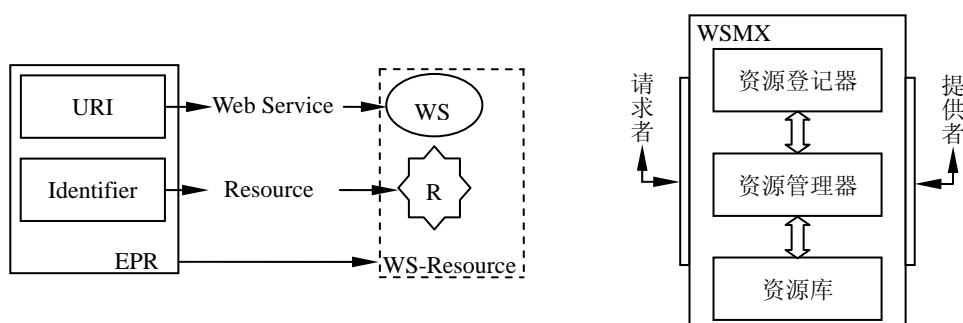


图 4.2 左图为 WSRF 的隐性资源模式，右图为 WSMF 的资源管理模式

#### 4.2.2.4 耦合性

框架内各个单元的松耦合性是 WSRF 和 WSMF 追求的目标。站在服务提供者的角度，为了引入有状态资源这么一个相对于 Web 服务而言的概念，WSRF 内部的耦合性必然会因为 Web 服务与资源各种各样的联系而增强，作为一个系统，WSRF 的内部结构及其内部单元间的关系也会变得复杂起来。为了弥补这一缺陷，WSRF 采用“隐性资源模式”来封装 Web 服务和若干有状态资源。“隐性”指对服务请求者来说不需要了解有状态资源标识符(标识符代表有状态资源的身份信息，用来识别有状态资源)的内容，状态资源标识符只对访问它的 Web 服务有意义<sup>[9]</sup>。这种处理方式使有状态的资源对服务请求者是透明的，而且有状态资源标识符可以视作 Web 服务的一个静态属性通过消息机制（如 SOAP）进行传输。虽然加入有状态资源这个概念使 Web 服务变得复杂，但是“隐性资源模式”对 Web 服务和有状态资源进行封装后，服务请求者面对的依旧是一个对象——WS-Resource，而不是两个独立的对象——Web 服务和有状态资源，WSRF 的耦合度成功降低，实现了松耦合的目标。

在设计理念上与 WSRF 相比，WSMF 更加强调系统内部各组成部分强烈的非耦合关系。WSMO 将 Web 服务过程中涉及的对象分成 4 类，界定清晰，这种做法体现出 WSMF 将 Web 服务过程进行完全分解，目的就是为了遵循其一条重要的设计原则：彻底地解耦。WSMO 的 4 元素中，目标与 Web 服务相分离形成两个独立的

实体最能体现这种解耦特点。Web服务只描述该服务具有哪些功能（能做什么）和哪些接口（如何被调用）。如果说Web服务是站在服务提供方的角度描述Web服务的话，那么目标则是站在服务请求方的角度详细说明客户在请求该服务时希望该服务能提供哪些功能、达到哪些目的<sup>[37]</sup>。目标和Web服务这两个实体的共同存在，不仅没有重复交叠，显得多余，而且还实现了请求方和提供方的解耦。进而，两个实体相互自治的关系不仅促进实体可被重复利用，还保证Web服务发现机制能依据特定而明确的目标自动发现适合的Web服务。值得补充的是，耦合关系的弱化引起实体与实体间相对独立和自治，在这种环境下，Web服务进程的顺利推进必须依赖两种方法：（1）实体提供丰富而友好的接口<sup>[31]</sup>；（2）实体间要有强大的中介促进实体与实体的交互。

#### 4.2.2.5 异构处理

在异构处理方面，WSRF的能力不如WSMF强。作为Web服务体系的支撑技术，WSRF和WSMF都不可避免地会面对Internet中大量而复杂的异构问题。一般而言，异构问题通常表现在数据（data）和过程（process）两个方面，WSRF和WSMF分别采用了一些方法来处理这两个层面的异构问题。在数据层面，WSRF使用WS-Resource统一描述资源，使用WS-ResourceProperties、WS-ResourceLifetime两大子规范提供的统一接口操作资源<sup>[27]</sup>。举例来说，获取WS-Resource某一属性的方法在WSRF中被统一定义为GetResourceProperty，封装在SOAP中的代码为：

请求

```
<SOAP_ENV: Body>
  <wsrf: GetResourceProperty>somePropertyName</wsrf: GetResourceProperty>
</SOAP-ENV: Body>
```

回应

```
<SOAP_ENV: Body>
  <wsrf: GetResourcePropertyResponse>
    somePropertyValue
  </wsrf: GetResourcePropertyResponse>
</SOAP-ENV: Body>
```

采用这种统一的操作接口，WSRF屏蔽双方资源的异构性（数据格式的差异以及由不同的文件管理系统、数据库管理系统造成的数据管理上的差异）。而WSMF在处理数据层异构时主要依靠本体影射和本体管理技术来完成。采用WSML，WSMF使用本体来描述Web服务中的各种成分，这已经在一定程度上迈出了资源由异构向同构转化的第一步。如果说这第一步是只是在语法层面上处理了数据异构问题，那么进一步WSMF将在语义层面上整合不同的概念，现有的工具XS-T rule就可以用来提取语义信息，结合WSMO的中介，还可以合并语义间的差异<sup>[31]</sup>，实现不同术语、不同本体的整合。

在过程层面，WSRF自身没有提供方法来处理过程异构的问题，主要是依靠一

些已有的方法和技术，比如Agent，JNDI<sup>[38]</sup>（Java Naming and Directory Interface），而且基本上都建立在Java平台上。WSMF则力图使用中介来协调不同的通信模式解决过程异构的问题。这种异构通常在Web服务通信时发生，当通信模式不匹配，过程无法进行下去时，WSMO的中介会做出协调，要么将一条消息分解成若干条，要么将若干条消息整合成一条，或者加入或者删除某些消息，帮助过程继续进行下去。WSMF较WSRF有更强的异构处理能力，主要归因于前者有专门处理异构的模块——中介，负责异构对象间的协调。

#### 4.2.2.6 Web 服务运行环境

作为Web服务体系的支撑平台，WSRF和WSMF都应该考虑Web服务的运行环境。WSRF没有将运行环境纳入自身的体系框架中，而是借助其他已有的标准规范，创建Web服务的运行环境，实现Web服务的发现、调用、合成等功能。这些已有的规范包括：WSDL、SOAP、UDDI、WS-CAF（Web Services Composite Application Framework）、WS-BPEL（Web Services Business Process Execution Language）、WS-CDL（Web Services Choreography Description Language）等。

WSMF创建了自己的Web服务运行环境——WSMX，但是WSMX不是完全重新定义一套自己的全新规范，而是对上述规范进行了整合，同时根据WSMO的特点对这些整合的规范进行了语义扩展。其中一个重要的特点是WSMX和WSMO在WS-CDL的基础上丰富了一个“orchestration(综合调整)”概念，作为“choreography（分解编排）”的补充<sup>[39]</sup>：前者说明一个Web服务可以分解成几个子Web服务组合实现同一个目标；后者说明Web服务提供哪些接口与服务请求者交互完成该服务的功能。这样以WSMO为基础，结合本体技术，再加上WSMX中的服务发现引擎、服务挑选引擎和choreography引擎、orchestration引擎，就能更容易发现、调用、组合Web服务<sup>[40]</sup>。

#### 4.2.3 两者的关系

通过前面对WSRF和WSMF的特点和异同点的分析，可以发现两者的立足点都是Web服务，只是WSRF从网格方向作了扩展，WSMF从语义方向作了扩展。近来一些参考文献41-46讨论的话题主要关注在网格环境中如何为网格服务引入语义性。一个关键思路就是在以WSRF为支撑的OGSA框架中的每一个层次引入WSMF的建模思路和基于本体的环境运行，在WSRF和WSMF的共同作用下形成语义网格服务（SGS，Semantic Grid Services）。可见，WSRF和WSMF在语义网格服务的架构过程中是一种互为补充的关系，如图4.3所示：WSRF在水平方向起到基础支撑作用，它解决网格服务中服务和资源的捆绑问题；WSMF在垂直方向起到语义扩展作用，它逐层渗透到网格服务的各个层次中，实现在语义环境中服务的本体描述以及服务的自动发现、调用、组合。

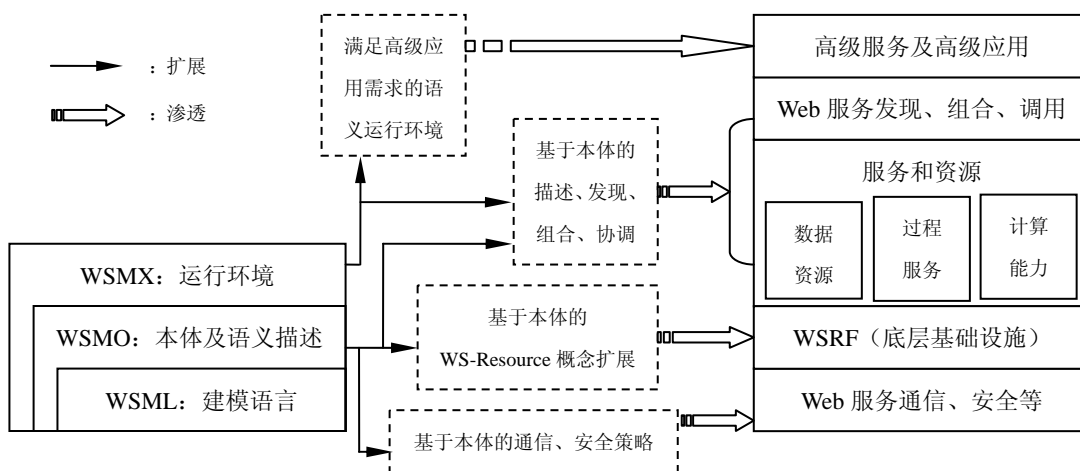


图 4.3 实现语义网络服务过程中 WSRF 和 WSMF 互为补充的关系

综上所述，WSRF 和 WSMF 立足于 Web 服务，分别在网络和语义方面进行扩展，成为网络服务和语义网领域两个重要的技术框架。总的来说，它们的目标定位不同，导致各自的特点也不同。前面分别在目的、重点、Resource 的含义、耦合性、异构处理和运行环境等方面对 WSRF 和 WSMF 作了对比，表 4.3 对两者在这 6 个方面的不同进行了总结。

表 4.3 WSRF 和 WSMF 对比小结

	WSRF	WSMF
目的	网格中引入 Web 服务，明晰服务和资源的概念	Web 服务中引入语义，形成语义网络服务
重点	资源的状态及对状态的操作和维护	利用本体增强 Web 服务的语义性
Resource 的含义	有状态的 WS-Resource	WSMO 四元素：本体，目标，Web 服务，中介
耦合性	松耦合：采用隐式资源模式	非耦合：采用 WSMO 四元素
异构处理	使用 WS-Resource 描述资源，统一接口操作资源；采用 Agent, JNDI, Java 等技术	异构处理能力更强，有专门负责异构对象间协调的中介
运行环境	WSDL、SOAP、UDDI、WS-CAF、WS-BPEL、WS-CDL 等构成运行环境	专有的运行环境 WSMX

WSRF 和 WSMF 两大技术框架各具特色，但又可以互相补充。语义网络服务需要融入 Web 服务技术和语义网技术，WSRF 和 WSMF 搭建好了技术框架，两者互补其长，是融合 Web 服务、网络和语义网的一条主要途径，也是目前的一个研究热点。

## 5 WSRF 的实现

WSRF 是为网格与 Web Services 结合而提出的标准规范，既然作为规范它一定对 Web、服务和网格服务的实践具有指导作用，换句话说，在实践中如何贯彻落实 WSRF 所体现的思想，也是国内外网格、Web Services 领域重点研究的内容。紧紧围绕 WSRF，Globus Alliance、中国教育科研网格（China Grid）、HP（惠普）、Microsoft（微软）、University of Virginia 等机构、公司和大学纷纷启动各具特色的项目，促进 WSRF 的实现。

### 5.1 GT4

GT4 全称是 Globus Toolkit version 4。Globus Toolkit 是一种开放源代码的软件工具包，用来构建网格系统及其应用。工具包中包含安全、资源管理、数据管理、通信、错误检测、移植等多种功能模块，并被封装成一个个组件能被单独或同时调用。Globus Toolkit 通常也称作网格中间件，是研制各类网格平台和网格应用的基础和核心，可以说，Globus Toolkit 是网格得以实现的关键。图 5.1 描述 GT4 的整体框架<sup>[47]</sup>。

作为提出 WSRF 的主要成员之一，Globus Alliance 当然会在最新版本的 Globus Toolkit 即 GT4 中引入 WSRF 的概念。从 Globus Toolkit 的发展来看，GT2 主要面向高性能计算，GT3 是基于 OSGI 实现的网格平台，GT4 则是基于 WSRF 实现的网格平台，总的趋势是与 Web Services 走得越来越近，最后已经融为一体了。相对于 GT3，GT4 从两个方面实现 WSRF 标准规范。首先，对已有 GT3 协议进行改造。GT4 借用了 GT3 的所有重要协议，但不是直接照搬，而是将 GT3 协议如用于数据管理的 RFT<sup>[48]</sup>（Reliable File Transfer，可靠文件传输协议）、用于资源管理的 WS-GRAM<sup>[49]</sup>（Web Service Grid Resource Allocation Manager，基于 Web 服务的网格资源分配管理协议）和用于信息服务的 MDS<sup>[50]</sup>（Monitoring and Discovery Service，监控与发现服务）按照 WSRF 规范进行重新设计。表 5.1 中总结了 GT4 主要协议的一些基本特性和兼容性。

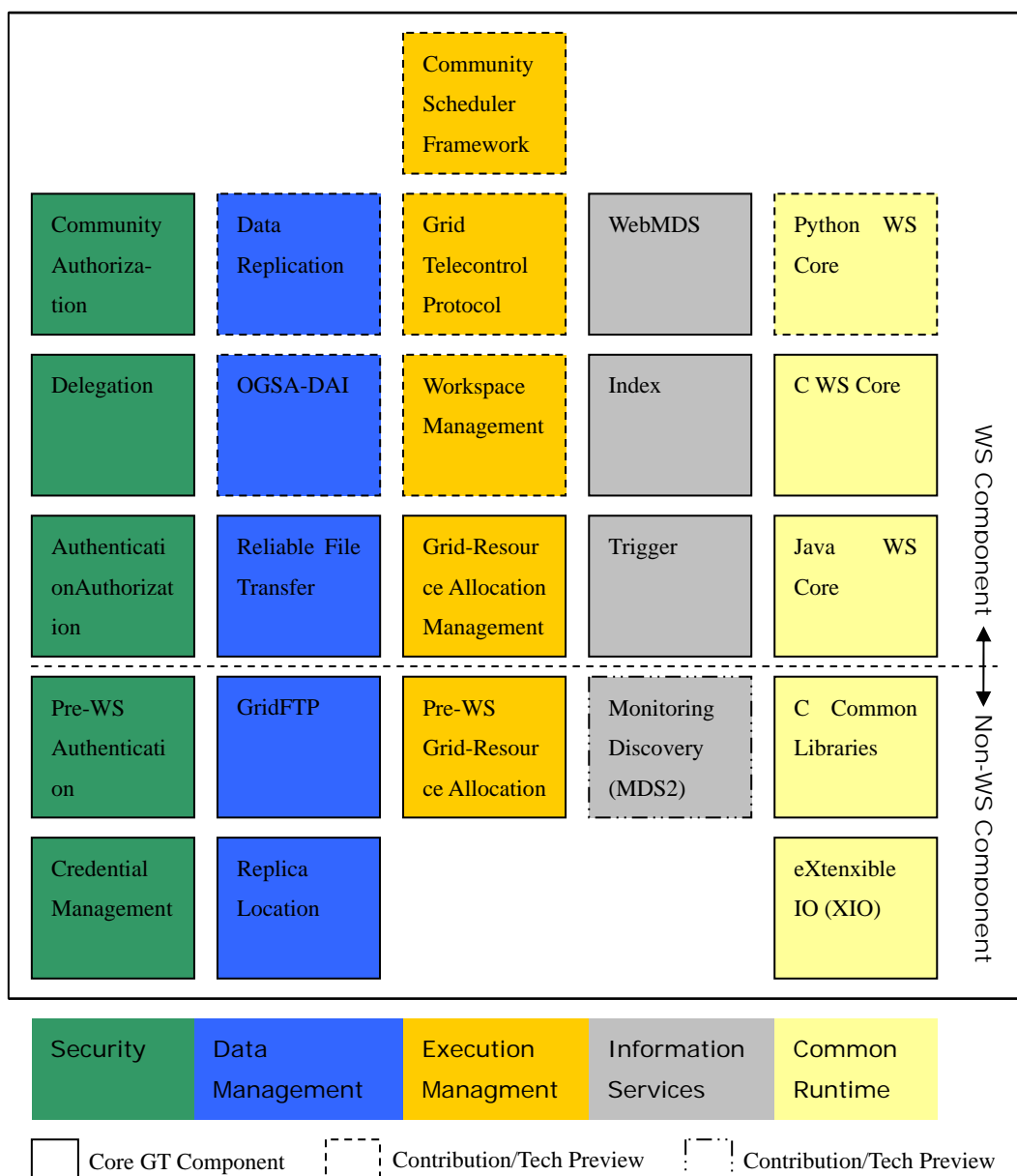


图 5.1 GT4 的整体框架图

表 5.1 GT4 主要协议的基本特性和兼容性

服务	协议	特性	向后兼容性
数据传输	RTF	1.使用 GridFTP 控制和监视第三方文件传输； 2.指数补偿 (Exponential back-off)； 3.全部传输或者全部不传输； 4.并行传输； 5.TCP 缓冲区大小； 6.递归目录传输。	与 OGSI(GT3.2) 不存在向后兼容性。
资源管理	WS-GRAM	1.改进任务性能：并行性、吞吐量和等待时间等； 2.改进可靠性/可恢复性； 3.支持 mpich-g2 任务，包括： (1) 多任务提交；	该协议已经被修改能与 WSRF 兼容，但该版本不能向

		(2) 在一个任务中协调处理; (3) 在一个多任务的子任务之间进行协调。	后兼容性以前的版本。
信息服务	MDS4	索引服务 1.基于 WSRF 而不是 OGSi; 2.已经撤掉 Xindice 支持; 3.已经重构聚合的永久性配置; 全新的服务 1.触发服务; 2.聚合器 (Aggregator); 3.归档服务。	与 GT3.2 的索引服务不兼容, 因为该服务已经使用 WSRF 代替 OGSi 而重新进行了建模。

另外,除了改造 GT3 的重要协议外,GT4 还设计了一些新的 Web 服务组件,以满足网格服务中引入 WSRF 的新需求。GT4 中这些新的 Web 服务组件有:

(1) Pre-WS Authentication Authorization<sup>[51]</sup>

新出现的 Pre-WS Authentication Authorization 取代原来的 Grid Security Infrastructure<sup>[52]</sup>(GSI),划分成两方面内容: Message level security (消息级安全性)和 Authorization framework (授权框架)。Message level security 实现了两个标准: WS-Security 和 WS-SecureConversation。这两个标准提供 SOAP 消息加密、完整性和重放保护等功能。Authorization framework 是一个设计用来处理许多授权机制的组件,例如网格映射文件、访问控制列表 (ACL, Access Control Lists) 以及通过 SAML 协议处理用户授权。

(2) WS Core<sup>[53]</sup>

WS Core 是对两个新标准的实现: WSRF 和 WS-Notification。其他新特性包括基于 Apache Tomcat 的 JNDI<sup>[54]</sup>注册项、HTTP/1.1 客户机服务器支持、进行 WS-Addressing 转换的 URI 解析器服务等。

(3) C WS Core<sup>[55]</sup>

GT4 提供了一组使用 C 编写的基本工具集,可以用该工具集来创建启用 WSRF 的 Web 服务和客户机对 WS-Resource 和 WS-Notification 的确认。该组件中现有特性包括:

- 1) 独立的服务容器。
- 2) 在 C 应用程序中嵌入服务所使用的 API。
- 3) 在服务中管理资源所使用的 Resource API。
- 4) HTTP/1.1 客户机和服务器支持。
- 5) 直接从 WSDL 模式中生成纯 C 存根 (阻塞的或异步的)。
- 6) 可动态加载的基于新的 Extension API 的 Operation Providers 和 Service Modules。

## 5.2 CGSP

CGSP, 即ChianGrid Supporting Platform, 是为ChinaGrid<sup>[56]</sup> (中国教育科研网格) 建设和发展而研制的网格核心中间件。CGSP从 2004 年底发布的第一版起就符合OGSA架构, 并且在实现中参照了当时最新的网格理念WSRF。因为国际上最著名的Globus当时计划要到 2005 年 1 月才能推出具有上述特性的网格中间件, 所以CGSP是国际上第一个遵循OGSA架构, 参照WSRF规范实现的网格中间件<sup>[57]</sup>。

为符合WSRF标准, CGSP在各个功能模块的设计中都融入了WSRF思想。首先是CGSP中比较重要的网格服务容器模块。由于CGSP遵循WSRF规范, 因此必须支持服务和资源的基本特性、部属和运行管理等需求。网格服务容器是CGSP中专门用于解决这一问题的核心组件。网格服务容器是一个WSRF网格服务的基本运行环境, 为整个CGSP提供“面向服务的计算基础设施”, 它将被部署到每一个需要进行服务交互的网格节点之上。网格服务容器实现网格服务的远程部署、运行管理、服务状态监控、SOAP请求处理与转发等核心功能。从功能上看, 网格服务容器是一个扩展的Web服务容器, 除了提供基本的Web服务支撑功能外, 还需要对网格系统所要求的生命周期机制、异步通知机制、远程部署和热部署机制、批作业处理机制等提供必要的支持; 从形式看, 网格服务容器是网格服务的基本运行环境与一些实现共性系统功能的基本服务集合, 这些服务被预先部署在基本运行环境之中, 随着基本运行环境的启动开始提供服务。已正式发布的CGSP第一版中, 服务容器的实现基于Globus Toolkit 3.9.1, 体系结构如图 5.2 所示。Globus Toolkit 3.9.1 本身就是一个开放源码的WSRF网格服务容器中间件, 也是Globus项目组的Java WSRF Core<sup>[58]</sup>实现的早期技术预览版。虽然由于时间关系, 这个版本只包含一个基于WSRF的Java Web服务核心组件的实现, 尚不能提供一些WSRF必须的高层服务(真正完全遵循WSRF规范的Globus Toolkit 4 当时还没有出现), 但是为体现CGSP的先进性和前瞻性, 为满足CGSP的具体需求, CGSP服务容器仍然在Globus Toolkit 3.9.1 上进行设计, 根据实际需要扩展和增强其WSRF Core的实现, 并提供ChinaGrid正常运行所需的必要服务<sup>[59]</sup>。前面提到, CGSP在设计之初就考虑到在实现中参照当时最新的网格理念WSRF, 而且这一思想一直得到贯彻和落实, 在后续的CGSP升级版本中, 尤其是 2006 年 4 月刚刚推出的CGSPv2 beta1 版本, 不仅考虑到要支持最新的网络协议IPv6, 更重要的是及时采用GT4 这种最新的网格服务和Web服务机制, 以实现完全遵循WSRF规范的目的。图 5.3 描述的是CGSPv2 beta1 版本的网络服务容器结构体系<sup>[60]</sup>。从图中可以看到, 服务容器的实现基于Globus Toolkit-WSRF-Core-4.0.1, 而且CGSP的系统服务如配制服务、管理服务、监控服务等也都是采用WSRF标准实现的。

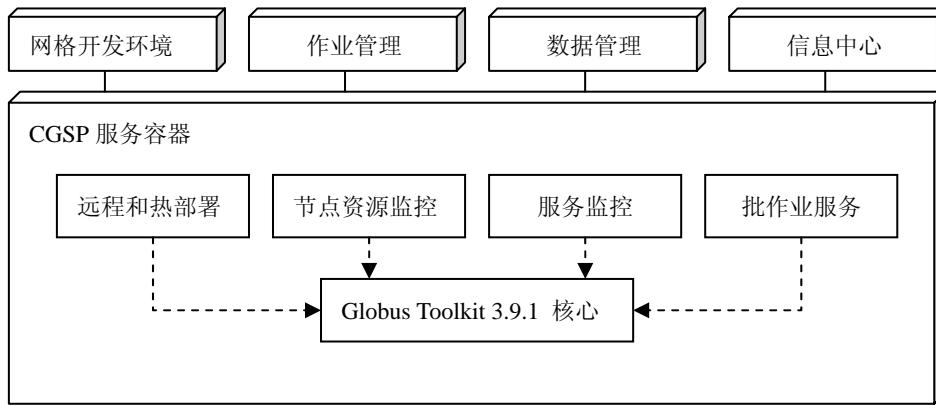


图 5.2 CGSP 中服务容器体系结构图

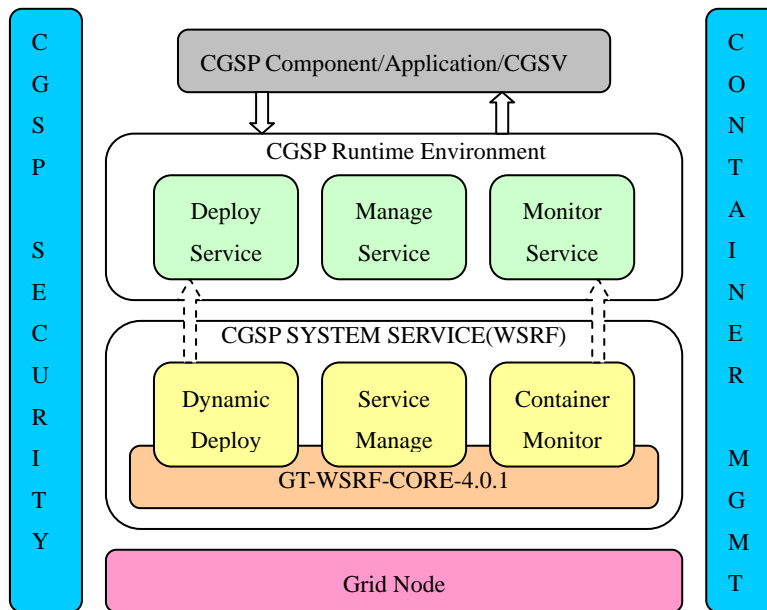


图 5.3 CGSPv2 beta1 第一版的服务容器体系结构图

除了网格服务容器模块，CGSP 的信息服务模块也按照 WSRF 的要求规范自身提供的服务。信息服务主要提供软硬件资源信息组织、资源和服务信息注册、资源信息更新/监控方法和资源发现与查询服务等功能。为实现这些功能，信息服务模块提供超级服务管理、网格服务管理、资源管理、服务调度管理、跨域信息共享、域间拓扑结构维护、服务分类信息维护等服务。这些服务的重要特点是完全符合 WSRF 规范，特别是网格服务管理和资源管理，充分参照 WSRF 规定的服务与资源的相互关系，采用隐性资源模式组织管理网格服务和资源。另外，服务调度管理在进行资源分配时，与网格服务容器协作，考虑资源的实时状态，依据状态为网格服务配制资源，这一点符合 WSRF 中将 Web 服务和有状态资源组成一个有机整体 WS-Resource、便于统筹安排的思想。

最后，CGSP 的安全管理模块中的容器与服务安全管理，主要依赖于 WSRF 的容器与服务的安全管理机制，即采用 Java WSRF 提供的 GSI Secure Conversation

与 GSI Secure Message 两种不同的消息鉴别机制来实现。

### 5.3 Apache WSRF (Apollo), Publish (Hermes) 和 Muse

在 Globus Alliance 为建设网格计算和应用开发新一代 Globus Toolkit, 促进 WSRF 和 WSN 实现的同时, HP 也在致力于为 Apache Software Foundation 创建一套用 Java 实现 WSRF 和 WSN 的方案<sup>[61]</sup>。最初, 这个方案由三个子项目组成, 分别是 Apollo<sup>[62]</sup>、Hermes<sup>[63]</sup> 和 Muse<sup>[64]</sup>。Apollo 项目旨在 WSRF 规范的实现, Hermes 项目旨在扩展 Apollo 项目遵循 WSN 规范, 而 Muse 项目则关注于扩展 Apollo 和 Hermes 项目遵循 WSDM MUWS<sup>[65]</sup> (Web Service Distributed Management) 规范。这三个子项目的共同目标就是促进以 WSRF 为核心的一系列 OASIS 标准规范的实现。2004 年 11 月, Apollo、Hermes 和 Muse 处于 “Apache 孵化器” (Apache incubator) 的起步阶段, 尚不成熟。发展至今, 三个子项目已经成为 Apache 认可的正式项目, 而且前两个变更了项目名称。Apollo 更名为 Apache WSRF, 2005 年 10 月 21 日发布了 Apache WSRF 1.1。Hermes 更名为 Apache Publish, 2005 年 10 月 21 日发布了 Apache Publish 1.1。Muse 没有更名, 2006 年 9 月 29 日发布了 Apache Muse 2.0.0。

Apache WSRF 采用 Java 语言为 WSRF 的实现提供了一种稳定可靠的方式, 具有以下一些特点:

(1) Apache WSRF 提供一种服务、处理框架, 能支持 WSRF 隐性资源模式, 可以用来向特定的 WS-Resource 发送请求; 借助 SOAP body 的最外层元素名或 SOAP header 的 wsa:Action 属性值向 Java 编写的服务方法发送请求; 还可以对请求进行 XML Schema 有效性确认。

(2) Apache WSRF 提供一个 WSDL 到 Java 的转换工具, 这个工具能把 WSRF 的 WSDL 文件转化成相应的 Java 类文件或部署配制文件, 还能结合 Apache XMLBeans<sup>[66]</sup> 为用户自定义的 WSDL 类型生成 Java 绑定。

(3) Apache WSRF 能完整实现由 WS-ResourceProperties、WS-ResourceLifetime 和 WS-MetadataExchange 等规范定义的所有 portType。

Apache Publish 项目是 WSN 规范的一个实现, 它建立在 Apache WSRF 项目之上。在 Publish 中, Notification Producer, Notification Consumer, Subscription Manager 都服务都被实现为 WS-Resource, 它有如下一些特性:

(1) Apache Publish 是一个 Apache WSRF 框架, 实现了 WS-RP (WS Resource Property)、WS-RL (WS Resource Lifecycle)、WS-MeX (WS Metadata Exchange) 等 PortType。

(2) Apache Publish 有一个 WSDL 到 Java 的转换工具, 可以从 WSRF、WSN 的 WSDL 文件中生成需要的类文件和部署描述文件。

(3) Apache Pubscribe 完整地实现了 WS-BaseNotification 规范中定义的所有 PortType。

(4) Apache Pubscribe 完整地实现了 WS-Eventing 规范中定义的所有 PortType。

(5) Apache Pubscribe 支持 WS-Topics 规范中定义的所有主题表达方式(Topic Express Dialect)。

(6) Apache Pubscribe 提供通用的发布/订阅 API，屏蔽底层所使用的通知机制 (WS-Notification/WS-Enhancements<sup>[67]</sup>)。

相对于前两者，Apache Muse 更是一个集大成者，能实现基于 WSRF、WSN 和 WSDM 的应用。Apache Muse 2.0.0 的特点有：

(1) 能完整地实现 WSRF 1.2、WSN 1.3、WSDM 1.1 和 WS-MetadataExchange 的所有 portType。

(2) 能简单地独自实现 WSDM Event Format 1.1。

(3) 能实现 WS-Addressing 1.0 和 SOAP 1.2 规范。

(4) 能配制在 Apache Axis2 (<http://ws.apache.org/axis2/>) 和基于 OSGi<sup>[68]</sup> 的平台上。

(5) 能用一种通用的编码模式定义各种平台上的不同资源。

(6) 将众多 Java bean 类整合到 WSRF 专门的一个状态模式中。

(7) 为通常一些与资源属性、服务分组、消息发布等相关的应用提供相当多的 API 接口。

(8) 提供一个持久化接口，使用户在机器重新启动后能恢复资源的状态。

(9) 提供一个完整的 WSDL 到 Java 的转换工具，能为服务方和客户方共同使用。

(10) 不断努力达到 100% 遵循 WSRF 等公认标准规范的目标。

要使用 Apache WSRF、Pubscribe、Muse 项目来创建应用非常简单。以 Pubscribe 为例，通常只要遵循如下开发步骤就可以轻松创建基于 WSRF 的 Web 服务应用项目：

(1) 创建工作空间，并创建必要的文件。

(2) 参考 Apache Pubscribe 项目提供的模板创建需要的 WSRF、WSN 服务的 WSDL 文件，这一步也是整个过程中最重要的步骤。

(3) 使用 WSDL2Java 转换工具生成必要的类文件和部署描述文件。

(4) 根据具体应用的需要，简单修改生成的类文件和部署描述文件。

(5) 编译生成的类文件，使用 Pubscribe 提供的工具部署 WSRF、WSN 应用。

(6) 最后测试并使用 WSRF、WSN 应用。

## 5.4 WSRF.NET

WSRF.NET项目由University of Virginia的网格计算工作组于2004年开始实施,最新版本是今年发布的WSRF.NET 3.0.1<sup>[69]</sup>。WSRF.NET采用Visual Studio.NET作开发工具,使用IIS(Internet Information Service)和ASP.NET<sup>[70]</sup>作为服务容器,因此它是在Microsoft.NET平台上对WSRF规范的实现<sup>[71]</sup>。WS-Resource与Web服务独立,能通过ADO.NET(如MS SQL Server, MSDE, MySQL)存储在数据库或其他一些用于计算的实体中。WSRF.NET包含一组库文件和工具使得Web服务很容易转换成一个与WSRF兼容的被封装的服务(a wrapper web service),如图5.4所示。这种服务由两部分组成<sup>[72]</sup>:一个是最初创建的原始服务;一个是创建者希望引入的一些功能函数,这个有点类似于WSRF为所有服务明确指定一些通用的portType。这种被封装的服务能够自动地解析由端点引用指定的执行环境并向适当的WS-Resource提供编程接口实现与WS-Resource的交互。当服务开始运行时,端点引用的引用属性指定的资源对象从数据库中调出,Web服务就能像访问自身内部的数据成员一样访问被调出的资源。一旦服务执行完毕,被访问的数据值还能保存到数据库中,而且服务执行的结果由ASP.NET依次编写入SOAP消息中,通过IIS返回给客户端。在安全方面,WSRF.NET目前以及将来都会以WSE<sup>[67]</sup>为基础,同时还支持WS-Security<sup>[73]</sup>、WS-SecureConversation<sup>[74]</sup>、WS-Policy以及新近出现的Web服务安全标准规范。

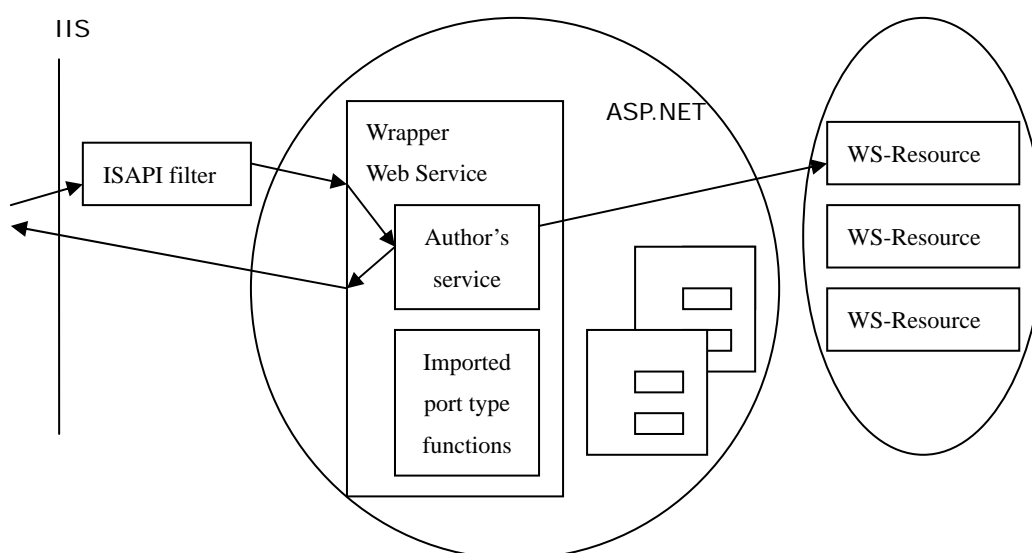


图 5.4 WSRF.NET 框架中的信息流

前面提到的 GT4、CGSP、Apache 系列和 WSRF.NET 是世界各大有影响力的组织机构在实现基于 WSRF 的网格服务和 Web 服务方面所作的研究、开发工作,具有一定的代表性。从另一个角度,即支持的开发语言来划分,WSRF 的实现有

以下五种：GT4-Java、GT4-C、pyGridWare、WSRF::Lite 和 WSRF.NET。GT4-Jave 支持 Java 语言，GT4-C 支持 C 语言，pyGridWare 支持 Python 语言，WSRF::Lite 支持 Perl 语言，WSRF.NET 支持 C#/C++/VBasic 语言。这五种 WSRF 实现的特点和比较分析可以查阅参考文献 75。

## 6 结论

WSRF是一组Web服务规范和约定，用来描述分布式环境中的有状态资源与Web服务的关系。WSRF引入有状态资源的概念，规定通过一个由Web服务地址和资源标识符组成的端点引用来访问WS-Resource，并提出用WS-Resource方式创建、管理有状态资源的方法。WSRF通过对Web服务属性文档操作来对状态进行检查和设置，通过信息交换来管理有状态资源的生命周期<sup>[26]</sup>。WSRF还能在交互中遇到错误时使用统一的错误处理机制处理错误。

WSRF 是 OGSi 对新的 Web 服务标准特别是 WS-Addressing 的重构和发展，在整体框架、功能划分、功能描述等方面两者十分相似，WSRF 的很多内容、思想都是从 OGSi 借鉴过来的。然而 WSRF 也根据 Web 服务的实际状况和需求作了一些变动和改进，特别是 WSRF 克服了 OGSi 的 3 个主要的不足指出，从整体上看 WSRF 更能满足 Web 服务发展到需要。

作为Web Services体系中的两个基本框架规范，WSRF和WSMF都是围绕Web服务这个核心，处理与Web服务相关的若干问题，因此它们存在的意义非常相似。只是两者在设计思路和具体细节上存在很多差异，各具特色。其实，WSRF和WSMF在语义网格服务的架构过程中是一种互为补充的关系，尤其是DERI<sup>[76]</sup>的WSMO工作组已经在 2005 年 8 月提出了WSRF语义扩展的草案<sup>[77]</sup>。可以预计，WSRF未来的一个发展趋势是结合WSMF向语义网方向发展。

自 2004 年 WSRF 被提出到 2006 年被 OASIS 采纳为正式标准，国内外的很多著名组织、机构和公司都在努力促进 WSRF 的实现。值得一提的是，中国教育科研网络开发的 CGSP 较早的遵循 WSRF 标准规范，使得我们国家在网络服务领域走在了国际前列。

## 7 名词术语

1 CGSP: ChianGrid Supporting Platform

2 EPR: Endpoint Reference, 端点引用

3 GED: XML global element declaration, XML 全局元素声明

4 GSH: Grid Service Handle, 网格服务句柄

- 5 GSR: Grid Service Reference, 网格服务引用
- 6 GT4: Globus Toolkit version 4
- 7 Implied Resource Pattern: 隐性资源模式
- 8 JNDI: Java Naming and Directory Interface
- 9 MDS: Monitoring and Discovery Service, 监控与发现服务
- 10 OGSII: Open Grid Service Infrastructure, 开放网格服务基础设施
- 11 portType: 端口类型
- 12 RFT: Reliable File Transfer, 可靠文件传输协议
- 13 SGS: Semantic Grid Services, 语义网格服务
- 14 SOAP: Simple Object Access Protocol, 简单对象访问协议
- 15 SWS: Semantic Web Services, 语义 Web 服务
- 16 UDDI: Universal Description, Discovery & Integration, 统一描述、发现与集成协议
- 17 W3C: World Wide Web Consortium
- 18 WSDL: Web Services Description Language, Web 服务描述语言
- 19 WSMF: Web Services Modeling Framework, Web 服务建模框架
- 20 WSML: Web Services Modeling Language
- 21 WSMO: Web Services Modeling Ontology
- 22 WSMX: Web Services Modeling Execution Environment
- 23 WSRF: Web Service Resource Framework, Web 服务资源框架
- 24 WS-Addressing: Web 服务寻址规范
- 25 WS-BaseFaults: Web 服务基本错误规范
- 26 WS-BPEL: Web Services Business Process Execution Language
- 27 WS-CAF: Web Services Composite Application Framework
- 28 WS-CDL: Web Services Choreography Description Language
- 29 WS-GRAM: Web Service Grid Resource Allocation Manager, 基于 Web 服务的网格资源分配管理协议
- 30 WS-Notification: Web 服务通知规范
- 31 WS-RenewableReferences: Web 服务可更新引用规范
- 32 WS-Resource: Web 服务资源
- 33 WS-ResourceLifetime: WS-Resource 生命周期规范
- 34 WS-ResourceProperties: WS-Resource 特性规范
- 35 WS-Resource-qualified endpoint reference: WS-Resource 有资格的端点引用
- 36 WS-ServiceGroup: Web 服务小组规范

## 8 参考文献

- 1 <http://www.globus.org/wsrf/>,2006-9-20
- 2 <http://www.w3.org/2002/ws/>,2006-9-20
- 3 SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation 24 June 2003[EB/OL].<http://www.w3.org/TR/2003/REC-soap12-part1-20030624/>, 2006-9-20
- 4 Web Services Description Language(WSDL) Version 2.0 Part1: Core Language, W3C Candidate Recommendation 27 March 2006[EB/OL]. <http://www.w3.org/TR/wsdl20/>, 2006-9-20
- 5 UDDI Version 3.0.2, UDDI Spec Technical Committee Draft, Dated 20041019[EB/OL]. <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>, 2006-9-20
- 6 <http://www.w3.org/>, 2006-9-20
- 7 <http://www.w3.org/2002/ws/arch/>, 2006-9-20
- 8 Web Services Architecture, W3C Working Group Note 11 February 2004[EB/OL]. <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>, 2006-9-20
- 9 Ian Foster, Jeffrey Frey, Steve Graham, et al. Modeling Stateful Resources with Web Services[EB/OL]. <http://www-128.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf>, 2006-6-2
- 10 <http://www.globus.org/ogsa/>, 2006-9-20
- 11 Bart Jacob. 网格计算：关键组件是什么？ [EB/OL] <http://www-128.ibm.com/developerworks/cn/grid/gr-overview/>, 2006-9-6
- 12 S.Tueche, K.Czajkowski, I.Foster, et al. Open Grid Services Infrastructure(OGSI) Version 1.0[EB/OL]. <http://xml.coverpages.org/OGSI-SpecificationV110.pdf>,2006-9-6
- 13 徐志伟,冯百明,李伟.网格计算技术[M].北京:电子工业出版社,2004.
- 14 Karl Czajkowski, et al. From Open Grid Services Infrastructure to WS-Resource Framework Refactoring and Evolution[EB/OL]. [http://www.globus.org/wsrf/specs/ogsi\\_to\\_wsrf\\_1.0.pdf](http://www.globus.org/wsrf/specs/ogsi_to_wsrf_1.0.pdf), 2006-6-2
- 15 Don Box, Erik Christensen, Francisco Curbera, et al. Web Services Addressing(WS-Addressing) W3C Member Submission 10 August 2006[EB/OL]. <http://www.w3.org/Submission/ws-addressing/>, 2006-9-20
- 16 Ian Foster. WS-Resource Framework: Globus Alliance Perspectives[EB/OL]. <http://xml.coverpages.org/FosterWSRF200401.pdf>, 2006-9-6
- 17 Karl Czajkowski, Donald F Ferguson, Ian Foster, et al. The WS-Resource Framework version 1.0 03/05/2004[EB/OL]. <http://www.globus.org/wsrf/specs/ws-wsrf.pdf>, 2006-9-2
- 18 Steve Graham, Anish Karmarkar, Jeff Mischkinsky, et al. Web Services Resource 1.2 (WS-Resource) OASIS Standard, 1 April 2006[EB/OL]. [http://docs.oasis-open.org/wsrf/wsrf-ws\\_resource-1.2-spec-os.pdf](http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf), 2006-9-2
- 19 Latha Srinivasan, Tim Banks. Web Services Resource Lifetime 1.2 (WS-ResourceLifetime) OASIS Standard, 1 April 2006[EB/OL]. [http://docs.oasis-open.org/wsrf/wsrf-ws\\_resource\\_lifetime-1.2-spec-os.pdf](http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-os.pdf), 2006-9-2
- 20 Steve Graham, Jem Treadwell. Web Services Resource Properties 1.2 (WS-ResourceProperties) OASIS Standard, 1 April 2006[EB/OL]. [http://docs.oasis-open.org/wsrf/wsrf-ws\\_resource\\_properties-1.2-spec-os.pdf](http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf), 2006-9-2
- 21 Tom Maguire, David Snelling, Tim Banks. Web Services Service Group 1.2 (WS-ServiceGroup)

- OASIS Standard, 1 April 2006[EB/OL].  
[http://docs.oasis-open.org/wsr/wsr/wsr-ws\\_service\\_group-1.2-spec-os.pdf](http://docs.oasis-open.org/wsr/wsr/wsr-ws_service_group-1.2-spec-os.pdf), 2006-9-2
- 22 Lily Liu, Sam Meder. Web Services Base Faults 1.2 (WS-BaseFaults) OASIS Standard, 1 April 2006[EB/OL]. [http://docs.oasis-open.org/wsr/wsr/wsr-ws\\_base\\_faults-1.2-spec-os.pdf](http://docs.oasis-open.org/wsr/wsr/wsr-ws_base_faults-1.2-spec-os.pdf), 2006-9-2
- 23 Steve Graham, David Hull, Bryan Murray. Web Services Base Notification 1.3 (WS-BaseNotification) OASIS Standard, 1 October 2006[EB/OL].  
[http://docs.oasis-open.org/wsn/wsn-ws\\_base\\_notification-1.3-spec-os.pdf](http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf), 2006-9-2
- 24 Dave Chappell, Lily Liu. Web Services Brokered Notification 1.3 (WS-BrokeredNotification) OASIS Standard, 1 October 2006[EB/OL].  
[http://docs.oasis-open.org/wsn/wsn-ws\\_brokered\\_notification-1.3-spec-os.pdf](http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf), 2006-9-2
- 25 William Vambenepe, Steve Graham, Peter Niblett. Web Services Topics 1.3 (WS-Topics) OASIS Standard, 1 October 2006[EB/OL]. [http://docs.oasis-open.org/wsn/wsn-ws\\_topics-1.3-spec-os.pdf](http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf), 2006-9-2
- 26 刘会斌,都志辉.网格与 Web 服务的融合—WSRF 和 WS-Notification[J].计算机科学,2005,32(2):76-79
- 27 Tim Banks. Web Services Resource Framework(WSRF)-Primer v1.2 Committee Draft 02 – 23 May 2006[EB/OL]. <http://docs.oasis-open.org/wsr/wsr/wsr-primer-1.2-primer-cd-02.pdf>, 2006-9-2
- 28 <http://www-128.ibm.com/developerworks/library/specification/ws-mex/>, 2006-9-20
- 29 <http://www-128.ibm.com/developerworks/library/specification/ws-polfram/>,2006-9-20
- 30 陈娟, 王汝传. 开放网格服务结构中 WSRF 与 OGSi 的比较分析[J].江苏技术师范学院学报,2005,11(4):38-45
- 31 D.Fensel, C.Bussler. Web Service Modeling Framework WSMF[EB/OL].  
<http://www.wsmo.org/papers/publications/wsmf.paper.pdf>, 2006-9-25
- 32 Business Process Management Initiative[EB/OL].<http://www.bpmi.org/> , 2006-9-25
- 33 Matthew Moran, Kashif Iqbal. Technical relationship between WSMX and Globus[EB/OL].  
[http://www.deri.ie/fileadmin/documents/20050114\\_WSMX\\_Globus\\_final.ppt](http://www.deri.ie/fileadmin/documents/20050114_WSMX_Globus_final.ppt), 2006-9-25
- 34 Ian Foster, Carl Kesselman. The Grid: Blueprint for a New Computing Infrastructure[M], published by Morgan Kaufmann, 1999
- 35 Axel Polleres, et al. A Conceptual Comparison of WSMO and OWL-S[EB/OL].  
<http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/>, 2006-9-25
- 36 Michael Stollberg, Armin Haller. Semantic Web Services Tutorial[EB/OL].  
<http://www.wsmo.org/TR/d17/resources/200507-ICWS/SWStutorial-iswc05.ppt>, 2006-9-25
- 37 Cristina Feier, John Domingue.D3.1 v0.1 WSMO Primer[EB/OL].  
<http://www.wsmo.org/TR/d3/d3.1/v0.1/wsmo-d3.1-v0.1.pdf>, 2006-9-25
- 38 Dirk Gorissen, et al. Integrating heterogeneous information services using JNDI[EB/OL].  
[http://dcl.mathcs.emory.edu/h2o/papers/h2o\\_hcw06.pdf](http://dcl.mathcs.emory.edu/h2o/papers/h2o_hcw06.pdf), 2006-9-25
- 39 Jacek Kopecky, et al. Relationship of WSMO to other relevant technologies[EB/OL].  
<http://www.w3.org/Submission/WSMO-related/>, 2006-9-25
- 40 Emilia Cimpian, et al. Web Service Execution Environment (WSMX) [EB/OL].  
<http://www.w3.org/Submission/WSMX/>, 2006-9-25
- 41 Omair Shafiq, et al. Using Triple Space computing for communication and coordination in Semantic Grid[EB/OL].  
[http://www.semanticgrid.org/GGF/ggf16/papers/TSC-semgrid\\_20060129.pdf](http://www.semanticgrid.org/GGF/ggf16/papers/TSC-semgrid_20060129.pdf), 2006-9-25
- 42 Ioan Toma, et al .Towards Semantic Web Services in Grid Environments[EB/OL].

<http://homepage.uibk.ac.at/~c703305/publications/swsg-skg2005.pdf>, 2006-9-25  
 43 Christoph Bussler, et al. Towards a Semantic GRID Service Operating System[EB/OL].  
[http://www.cetic.be/coregrid/NCOS/papers/PP\\_Matthew\\_Moran.pdf](http://www.cetic.be/coregrid/NCOS/papers/PP_Matthew_Moran.pdf),2006-9-25  
 44 Marko Niinimaki. Grid Resources Services and Data Towards a Semantic Grid System[EB/OL].  
<http://www.cs.uta.fi/reports/dsarja/D-2006-1.pdf>, 2006-9-25  
 45 Hongsuda Tangmunarunkit, Stefan Decker, Carl Kesselman.Ontology-based Resource Matching  
 in the Grid[EB/OL]. <http://epicenter.usc.edu/docs/iswc03.pdf>, 2006-9-25  
 46 John Brooke, et al. Semantic matching of Grid Resource Descriptions[EB/OL].  
<http://www.grid-interopability.org/semres.pdf>, 2006-9-25  
 47 <http://www.globus.org/toolkit/about.html>, 2006-10-26  
 48 [http://dev.globus.org/wiki/Reliable\\_File\\_Transfer](http://dev.globus.org/wiki/Reliable_File_Transfer), 2006-10-26  
 49 <http://www.globus.org/toolkit/docs/3.2/gram/>, 2006-10-26  
 50 <http://www.globus.org/mds/>,2006-10-26  
 51 <http://www.globus.org/toolkit/docs/4.0/security/prewsaa/>,2006-11-4  
 52 <http://www.globus.org/security/>,2006-11-4  
 53 <http://www.globus.org/toolkit/docs/4.0/common/javawscore/>, 2006-11-4  
 54 <http://java.sun.com/products/jndi/>,2006-11-5  
 55 <http://www.globus.org/toolkit/docs/4.0/common/cwscore/>,2006-11-5  
 56 <http://www.chinagrid.edu.cn/>,2006-11-11  
 57 CGSP 开发组. ChinaGrid & ChinaGrid Support Platform[EB/OL].  
[http://www.chinagrid.edu.cn/cgsp/download/cgspdocuments/tutorial\\_introduction.pdf](http://www.chinagrid.edu.cn/cgsp/download/cgspdocuments/tutorial_introduction.pdf), 2006-11-11  
 58 <http://www.globus.org/toolkit/docs/development/wsrif/3.9.1/index.html>, 2006-11-11  
 59 中国教育科研网格公共支撑平台工作组. 中国教育科研网格公共支撑平台功能说明  
 [EB/OL].  
[http://www.chinagrid.edu.cn/cgsp/download/cgspdocuments/CGSP\\_V1.0\\_FD\\_ch.pdf](http://www.chinagrid.edu.cn/cgsp/download/cgspdocuments/CGSP_V1.0_FD_ch.pdf),2006-11-11  
 60 CGSP Development Team. CGSP2 Introduction[EB/OL].  
<http://www.chinagrid.edu.cn/cgsp/doc/CGSP2-Intro/en/html/book.html>, 2006-11-11  
 61 Ian Springer. Apache WSRF, Pubscribe, and Muse, 26 September 2005[EB/OL].  
[http://www.byte.com/documents/s=9553/byt1127770733874/0926\\_springer.html](http://www.byte.com/documents/s=9553/byt1127770733874/0926_springer.html),2006-11-12  
 62 <http://ws.apache.org/wsrif/index.html>,2006-11-12  
 63 <http://ws.apache.org/pubscribe/index.html>,2006-11-12  
 64 <http://ws.apache.org/muse/>,2006-11-12  
 65 [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsdm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm), 2006-11-12  
 66 <http://xmlbeans.apache.org/>, 2006-11-12  
 67 <http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx>, 2006-11-12  
 68 <http://www.osgi.org/>,2006-11-12  
 69 <http://www.cs.virginia.edu/~gsw2c/wsrif.net.html>, 2006-11-12  
 70 <http://asp.net/>, 2006-11-12  
 71 Glenn Wasson. WSRF.NET 3.0 Programmer's Reference(13 January 2006)[EB/PL].  
[http://www.cs.virginia.edu/~gsw2c/WSRFdotNet/WSRFdotNet\\_programmers\\_reference.pdf](http://www.cs.virginia.edu/~gsw2c/WSRFdotNet/WSRFdotNet_programmers_reference.pdf),  
 2006-11-12  
 72 Marty Humphrey, Glenn Wasson. Architectural Foundations of WSRF.NET[J]. International  
 Journal of Web Services Research, 2005, 2(2):83-97  
 73 [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss), 2006-11-13

- 74 <http://www-128.ibm.com/developerworks/library/specification/ws-secon/>, 2006-11-13
- 75 Marty Humphrey, Glenn Wasson, Jarek Gawor, et al. State and Events for Web Services A Comparison of Five WS-Resource Framework and WS-Notification Implementations[A]. In: 14<sup>th</sup> IEEE International Symposium on High Performance Distributed Computing(HPDC-14), 24-27 July 2005
- 76 <http://www.deri.ie/>, 2006-9-25
- 77 Jacek Kopecky. D31v0.1 Semantic Web Services Resource Framework(WSRF-S) report[EB/OL]. [www.wsmo.org/TR/d31/v0.1/d31v01\\_20050808.pdf](http://www.wsmo.org/TR/d31/v0.1/d31v01_20050808.pdf), 2006-9-25