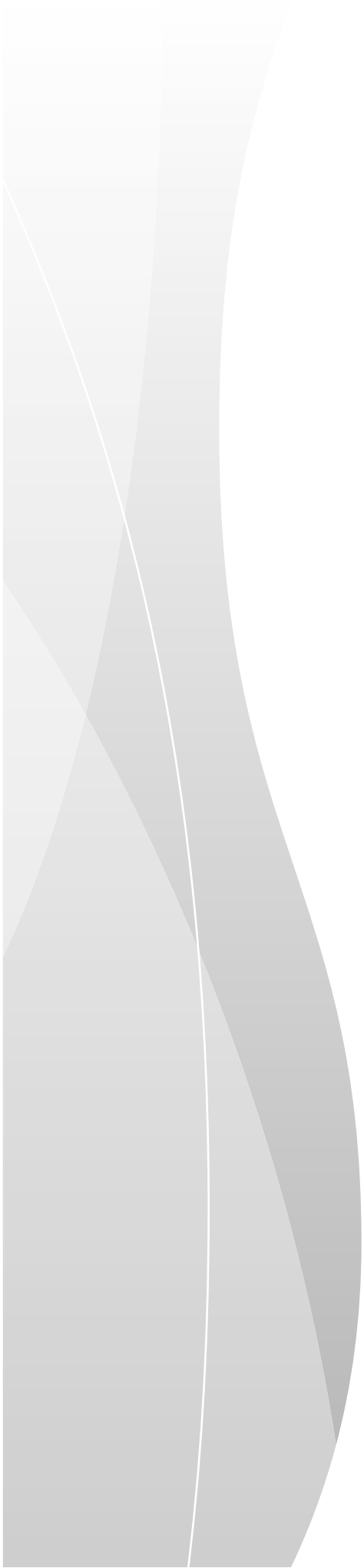

EFFECTIVE SOA DEPLOYMENT
USING AN SOA REGISTRY REPOSITORY

A Practical Guide
September 2005

Table of Contents

Today, Service Oriented Architecture [SOA] is being adopted by many IT organizations because it promises to make them more agile and efficient. This is possible, in large part, due to the loosely coupled nature of SOA, which enables service components to evolve without needing costly rework in existing deployments. Such agility and efficiency is also possible due to increased leverage and reuse of existing service components when building new service components.

Until recently, SOA deployments were done by early adopters, and consisted of limited pilots involving a few relatively simple services. The information about these services was equally simple and was managed, shared, and tracked informally using Web sites and e-mails.

As SOA deployments become mainstream, the complexity and scale of these deployments are growing steadily. It is no longer practical to use informal means such as Web sites and e-mails to manage, share, and track service-related information. At the same time, important new requirements are emerging, such as governance of services and related information artifacts.

An SOA registry-repository is increasingly becoming an important infrastructure middleware solution for managing this rising complexity and meeting new requirements. This paper explores some common challenges faced by large-scale SOA deployments, and offers practical advice on how an SOA registry-repository may be harnessed to manage these challenges.

Rising Complexity of Service Information

As service components become richer and more complex, it is no longer realistic to assume that all information about a service component can be captured in a single information artifact, such as a Web Services Description Language (WSDL) file.

Many different information artifacts describe a service component or relate it to other service components and information artifacts. Some examples follow:

- Multiple WSDL files may describe the various interfaces and protocol bindings of these interfaces for the service component.
- Extensible Markup Language (XML) schema files may describe the documents exchanged by messages in the service protocol.
- Business process orchestration for the service component may be described by artifacts such as Business Process Execution Language (BPEL) descriptions, and Electronic Business Extensible Markup Language (ebXML) business process specification schemas.
- Metadata may describe the assembly structure and subcomponents of a composite service.
- Extensible Stylesheet Language Transformations (XSLT) stylesheets may be used as adapters between service components to handle impedance mismatch due to service version differences.
- Web Services for Remote Portlets (WSRP) descriptions may describe how service components are used by portals.
- Organizational policies, business rules, and procedures may define how service components and service information artifacts may be defined and used.

The Need for Governance to Enforce Organizational Policies

Governance is defined as the policies, rules, and regulations under which an organization functions as well as the processes that are put in place to ensure compliance with those policies, rules, and regulations.

It is not enough to have organizational policies that stipulate how service components and service information artifacts may be defined and used. What is needed is a point of control within the SOA infrastructure that provides governance of service components and artifacts by enforcing the organizational policies that govern them. This ensures that the organizational policies are applied consistently and predictably across the SOA deployment and results in improved quality and integrity.

The Need for an Integrated SOA Registry-Repository

The need for a point of control and governance within the SOA deployment demands that service information artifacts be stored and managed in a consistent manner that allows enforcement of organizational policies. This is precisely the role served by a registry-repository service within an SOA deployment.

The following example describes a registry-repository using a common metaphor:

- A registry-repository is like your local library.
- It has a repository that contains all types of electronic assets, much like the library book shelves contain all types of published content including books, magazines, videos, and so on.
- It has a registry that contains metadata describing the electronic artifacts, much like the library's card catalog contains information describing the published content on its book shelves.
- The registry and repository are administered jointly. Within a library, the card catalog information and and books in the shelves are administered jointly.
- Any number of registry-repositories should be able to work together to offer a unified service, much like multiple libraries can participate in a cooperative network and offer a unified service.

Earlier SOA deployments recognized that the rising complexity and diverse nature of service information artifacts demanded more formal means of management, sharing, and tracking than simple Web sites. This led to the use of a registry such as a Universal Description, Discovery, and Integration (UDDI) registry to manage, share, and track service information artifacts. Though this is an improvement over the less formal Web site approach, it has a major limitation: a registry can only store links or pointers to service information artifacts. The actual artifacts must reside outside the registry, using informal and inconsistent means such as Web sites. This makes the actual artifacts ungovernable by the registry. What is missing is a repository that stores the artifacts. Using the library analogy, imagine a library that has only a card catalog but no books, and all the books were kept in people's homes.

A registry-repository provides an integrated solution able to store metadata such as links or pointers to artifacts, as well as the actual artifacts.

An SOA registry-repository should provide governance capabilities that enable organizations to define and enforce organizational policies governing the content and usage of the artifacts throughout their life cycles. Since organizational policies vary, an SOA registry-repository should enable organizations to enforce custom policies for the governance of any type of service information artifact throughout its life cycle. In particular, it should enforce conformance to such policies when a service information artifact is published or updated.

Validation: Key to Enforcing Organizational Policies

An important aspect of SOA governance is enforcing organizational and domain-specific business rules to *validate* information artifacts at the time they are published to the registry-repository. Validation improves the quality of published artifacts and ensures their conformance to established policies and standards. Such validation goes much further than syntax validation and, instead, performs semantic validation in an artifact-specific manner. For example, a rule may be enforced that WSDL artifacts MUST not contain bindings other than Simple Object Access Protocol (SOAP) bindings that use the *Document* style of communication.

A registry-repository should allow business rules to be enforced at the time of publishing. It should also allow such business rules to be defined by the organization and specialized for the types of artifacts. If an artifact fails publish-time validation checking, the registry-repository should either reject the artifact, or accept it as invalid and automatically notify responsible parties.

The Need for Standards-Based SOA Registry-Repository

As the need for an integrated, SOA registry-repository becomes broadly recognized, we will see a trend where vendors of registry-only products will begin to deliver new versions of their products with product-specific (proprietary) extensions to add repository capabilities. It is likely that these products will meet, either partially or fully, the governance requirements expected of an integrated registry-repository. Some IT architects may find this to be an acceptable solution that meets their needs. Indeed, this is a practical solution as long as the SOA deployment is under the control of a single organization that can ensure either that a single registry-repository is used or, if multiple registry-repository instances are required, they are based on the same vendor's product and interoperate.

In practice, SOA deployments tend to span organizational and governance boundaries. This is true even among different parts of the same enterprise. Organizations large and small prefer to have local autonomy over their SOA deployments, but also need to seamlessly integrate their services with those in SOA deployments of other organizations. In order to meet the requirement of local autonomy while providing seamless integration and interoperability globally, SOA deployments must federate with other SOA deployments using open standards.

It is therefore expected that the trend of vendor-specific, integrated registry-repositories will be ephemeral, and will be overshadowed by a new and growing trend towards integrated registry-repository products that are entirely based on standards facilitating federation and general interoperability.

Federated Information Management

The trend towards open standards-based, integrated registry-repositories is largely because they allow organizations to share and link information with other organizations in a secure manner. Federated information management allows multiple registry-repositories to federate together and appear as a single, virtual registry-repository, while allowing individual organizations to retain local control over their own registry-repositories. For example, a government may deploy a federated registry-repository that consists of several registry-repositories operated by various jurisdictions of government at the municipal, provincial, territorial, and federal levels. Such a federated registry-repository would give residents the ability to discover information seamlessly within any level or jurisdiction of the government. For example, they could search for services available for elder care across all levels of government.

The United Nations Centre for the Facilitation of the Administration, Commerce, and Transport (CEFACT) organization considered a federated information management requirement as the primary requirement for choosing an open standards-based, integrated registry-repository.

Federated Policy Management

Governance requires enforcement of organization policies that are described in a machine-processable syntax, typically XML. In an SOA deployment, these policy files must be published, managed, discovered, and governed like other service information artifacts. In a federated SOA deployment, policies need to be linked and composed across enterprise boundaries within the federation.

At present, policies are managed within proprietary policy stores in product-specific ways. Policy expression syntax standards have not yet fully matured, either. In any case, it is likely that an SOA registry-repository will be used to manage and govern policies in much the same manner as other SOA artifacts. It is also likely that policy standards will mature during the next year to provide a standard policy expression syntax that is capable of expressing all types of policies. The ebXML Registry standard supports federated policy management of access control policies expressed in the XML syntax defined by the OASIS Extensible Access Control Markup Language (XACML) standard.

Federated Identity Management

Open standards-based, integrated registry-repositories are necessary but not sufficient for federated SOA deployments across organizational boundaries. There needs to be a way to securely manage identities of clients (human and machine), and authenticate them when they access service components and artifacts within a federation. The challenge is to open up an enterprise's services and artifacts to clients in other enterprises while continuing to maintain air-tight security.

Federated identity management addresses this challenge by establishing a circle of trust across all services (including all registry-repositories) within the federation. Within this environment, it is possible to support single sign-on (SSO) where a client is authenticated *once* and then uses the authenticated session to access services throughout the federation multiple times without having to reauthenticate. It is also possible to map identity credentials across systems within the federation.

A registry-repository should support federated identity management features such as single sign-on, and integrate with identity and access management services using open standards such as Security Assertions Markup Language (SAML) and Liberty.

Service Discovery and Reuse

An important motivation behind SOA is its component-centric architecture. This architecture enables the building of complex service components from simpler, task-specific service components, much like building castles out of Lego blocks. Unlike Lego, however, a service component may be reused any number of times in other services without ever running out of blocks. An example of a task-specific service component is a credit rating service that may be used by any number of more specialized and complex services such as car and home loans, credit cards, or financial aid applications.

A registry-repository contains all service-related artifacts for service components available within an SOA deployment. This information should be available to developers during service design time, so they can discover existing service components in order to reuse and leverage them within a new service being developed. It should be possible for discovery of the service-related artifacts to be done using any search criteria relevant for the organization and its SOA deployment.

A registry-repository should provide discovery capabilities that are extensible and can accommodate the simplest to the most complex domain-specific discovery queries. Specifically, its discovery queries should not all be predefined. Instead, it should provide an ad hoc query syntax supporting complex predicates that can be combined using Boolean logic. The following are some examples of service artifact discovery queries expressed in plain English:

- Find all WSDL documents that use a specified namespace pattern
- Find all Service Bindings or Services that have a certain text pattern in their documentation
- Find all Service Bindings that are SOAP bindings AND use DOC Literal style AND do not use HTTP as transport
- Find all WSDLs with Service implementations that use specified implementation platform (for example, Java™ 2 Platform, Enterprise Edition or J2EE™, .Net, and so on)
- Find all WSDLs with Service implementations that use specified platform resources (such as database, Java Message Service or JMS, Java API for XML Registries or JAXR, and so on)

The flexibility and expressive power of an ad hoc query syntax can lead to problems if not properly governed and managed by the organization. For example, the query syntax may be too complex to be of any use to typical users. It may also lead to inefficient queries, placing an excessive load on the registry-repository. What is needed is a stored query capability similar to relational databases.

A registry-repository should allow an organization to define parameterized, ad hoc queries as stored queries within the registry-repository. It should allow such queries to be invoked by a client with some or all of its parameter values supplied as invocation parameters. This enables the organization to define approved, domain-specific discovery queries within the registry-repository, so that these discovery queries may be used by their user community. The queries may be exposed to the users as simple Web forms hiding all the underlying complexity within the registry-repository server.

Cataloging: Key to Artifact Discovery

Cataloging of artifacts improves their discoverablity and is essential in supporting the kind of artifact-specific queries mentioned as examples in the previous section.

Cataloging of artifacts is very similar in concept to indexing of tables in a relational database. In both cases, information is automatically converted to metadata at the time it is published, and the metadata is used to facilitate efficient discovery of the published information. For example, an organization may define cataloging policies for WSDL artifacts such that when a WSDL document is published, it is cataloged in a WSDL-specific manner to generate metadata that includes information on:

- The documents imported by the WSDL document (such as other WSDL documents and XML schema documents)
- The name spaces used by the WSDL document and documents imported by it
- The name and description of the bindings, interfaces, and types used by the WSDL document

Such metadata can then be used in WSDL-specific discovery queries, such as the queries mentioned as examples in the previous section.

Service Artifact Dependency Management

As more and more service components are reused within other service components, the task of tracking the network of dependencies between service components becomes more challenging and significant. This is another challenge that is made easier by an SOA registry-repository where interservice dependency information can easily be managed as relationships between service information artifacts. Examples of such relationships include Contains, Extends, Implements, Supersedes, Uses, Depends Upon, and more. As with discovery queries, SOA deployments vary, and a fixed set of relationship types may not be suitable for all deployments.

A registry-repository should provide a set of standard relationship types, but also allow an organization to define additional relationship types based on its specific requirements.

Phased Deployment of Services

Service components are deployed in phases. During each phase, organizational access control policies (ACPs) determine the controlled community of users which has access to the service. The controlled community is typically defined in terms of functional roles people play or groups they belong to. Here is a typical, phased-deployment scenario (with potential roles and groups in *italics*):

- Development phase – Service component is accessible to the *Development Engineers, QA Engineers, Document Writers,* and so on.
- Pilot phase – Service component is accessible to *Pilot Users*.
- Production phase – Service component is accessible to *Licensed Users*.
- Obsolescence phase – Organizational ACPs restrict the user community instead of broadening it (by preventing new clients from using the service, and grandfathering those who are already clients of the service).

A registry-repository should allow ACPs to be defined and enforced for service information artifacts. Since ACPs tend to be fairly specific to organizational needs, the registry-repository should allow for fine-grained access control policies that can accommodate specific needs.

For federated SOA deployments, the registry-repository should also allow ACPs to be defined anywhere in the federation, and be usable as building blocks for more complex ACPs.

Service Evolution and Versioning

Service components evolve over time for a variety of reasons, such as the need to fulfill new requirements. A service's evolution may involve changes in its implementation and/or public interface. Changes to a service interface need much more careful management because of the potential impact to existing clients of the service. These changes typically require a new version of the service to be deployed, while maintaining the older version until its clients have had time to migrate to the new version according to their own schedule. New versions of a service or a service component also typically require publication of corresponding new versions of its service information artifacts .

A registry-repository should provide versioning capabilities that enable automatic version control of any type of service information artifact. The versioning feature should allow publishers and organizational policies to determine when modifications to an existing information artifact should be treated as an update of the existing artifact, and when these modifications should result in a new version of the information artifact.

Service Change Notification

When a service evolves, it is important to notify its consumers of the changes to the service. For example, when a new version of a service becomes available, it is important to tell administrators responsible for clients of that service, so they can begin planning the migration of their clients to the new version.

A registry-repository should provide a change notification capability that allows interested parties, such as system administrators, to create subscriptions to events within the registry-repository that may be of interest to them. Such a capability should allow a subscription to be flexible enough to express precisely the types of events that are of interest to the subscriber. It should be possible for change notification to automatically invoke services that can automate governance workflow based upon the change notification.

Registry-Repository SDKs and Custom Applications

Many people focus on the role of a registry-repository as a design-time service to publish, manage, discover, and govern service information artifacts. This is an important role, but it is also possible for deployed services to use a registry-repository as a source of operational and configuration data during runtime. To facilitate this use case, a registry-repository should provide a software development toolkit (SDK) to develop custom registry client applications and services. A registry-repository SDK for the Java platform should include support for JAXR, the standard application programming interface (API) for registries and repositories within the Java platform.

Choosing a Registry-Repository

As IT organizations evaluate which registry-repository to deploy as part of their SOA infrastructures, the choices often fall into the following categories:

1. Proprietary registry-repository
2. UDDI registry without a repository
3. UDDI registry with a proprietary repository
4. ebXML registry-repository
5. Combination of UDDI registry and ebXML registry-repository

As mentioned earlier, federated SOA deployments require a standards-based registry-repository. This suggests eliminating options 1 and 3 above. The remaining standards-based choices involve two standards, UDDI and ebXML Registry.

A UDDI registry offers a subset of capabilities offered by an ebXML Registry. See Table 1: Registry Standards Comparison Matrix for details. In particular, it provides only a registry and no repository. What gets published in a UDDI registry are pointers to service artifacts such as WSDL. What gets published to the ebXML Registry are not just pointers to service artifacts, but also the actual artifact themselves. Thus, an ebXML registry-repository can be used for governance of any type of service artifacts throughout their life cycles.

An ebXML registry-repository is highly extensible. This extensibility has been a double-edged sword for the ebXML Registry standard. On the positive side, it enables organizations to enforce custom governance policies for managing their service artifacts. On the negative side, it also makes the standard harder to understand and deploy because of its generic and extensible nature. This problem is being addressed by the emergence of vertical or use case-specific profiles of ebXML Registry that define precisely how to make use of its generic and extensible features in a standard, interoperable manner within a particular domain. For example, in the Web services area of greatest interest to those doing SOA deployments, a Web Services Profile is being defined by the OASIS ebXML Registry Technical Committee (TC).

A UDDI registry may be adequate for simpler SOA deployments. However, experience has shown that as SOA deployments increase in complexity and scale — which they inevitably do — an ebXML registry is a much better fit. It is sometimes the case that a SOA deployment starting with a UDDI-based implementation will later migrate to an ebXML Registry-based one, as needs grow.

It is not necessary to choose between the two standards. Newer products are appearing that support both standards in a single engine. An example of this new class of registry-repository products is Sun's Service Registry. When deploying such a registry, be aware that the functionality offered by each interface is quite different. As discussed, the ebXML Registry interface provides the fuller set of capabilities. This means that a dual-interface registry like the ebXML Registry interface is employed more pervasively, while the UDDI interface is used more specifically to interoperate with clients restricted to the UDDI protocol.

Conclusions

In the past, enterprise integration was achieved via data integration using a common enterprise database as the integration point. SOA represents the latest approach to enterprise integration via loosely coupled service integration based on a component and document-centric architecture. The next logical step is a federated SOA deployment that achieves enterprise integration within and across enterprise boundaries via service integration enhanced with secure, federated information management.

Today's SOA deployments are becoming increasingly complex and require strong governance capabilities. A standards-based registry-repository is emerging as an important SOA infrastructure component. First-generation Web services registries based solely on UDDI lack many important capabilities, including repository functions, which are necessary for governing and managing complex SOA deployments. An ebXML registry-repository provides a much richer set of capabilities to meet the advanced governance and federated information management requirements of complex SOA deployments. A new class of registry-repository will support both UDDI and ebXML Registry standards in a single solution. Sun's Service Registry is a prime example, and offers features that meet the growing requirements of today's SOA deployments.

Registry Standards Comparison Matrix

Table 1 compares UDDI and ebXML Registry 3.0 standards in various categories.

Table 1: Registry Standards Comparison Matrix

Category/Feature	ebXML Registry 3.0	UDDI 3.0
Standards Leveraged		
Service Description Standards	WSDL 1.1	WSDL 1.1
Messaging Standards	SOAP 1.1 with Attachments	SOAP 1.1
Message Security Standards	OASIS Web Service Security	
Access Control Policy Standards	XACML 1.0	
Identity Management Standards	SAML 2.0	
Architecture		
Object-Oriented API	Yes • API offers a few task-oriented calls that may be used by any type of metadata object • Consistent and uniform actions supported via few API calls across entire information model	No • API offers type-oriented calls that keep growing as new types are added in each release of UDDI standard
Object-Oriented Information Model	Yes	No
Extensible API	Yes • New API calls may be defined using standards-based API extensibility features	No
Extensible Information Model	Yes • New information model types may be defined using standards-based type extensibility features	No

Category/Feature	ebXML Registry 3.0	UDDI 3.0
Core Features		
Registry	Yes • Rich set of ~25 standard metadata classes	Yes • Inadequate set of ~6 standard metadata classes
Repository	Yes • Integrated registry-repository • Any type of electronic content supported	No
Publish		
Can publish metadata describing any type of information artifact	Yes	Yes
Can publish any type of information artifact	Yes • Information artifacts subject to governance	No • Actual information artifact resides external to registry and therefore not subject to governance
Discovery		
Predefined queries	Yes	Yes
User-defined queries	Yes	No
Ad hoc queries	Yes • Supports predicate combination using logical operators	No
SQL query syntax	Yes • Ad hoc syntax supports unlimited number of queries	No
XML query syntax	Yes • Ad hoc syntax supports unlimited number of queries	Yes • Fixed syntax supports ~4 predefined queries
Stored parameterized queries	Yes	No

Category/Feature	ebXML Registry 3.0	UDDI 3.0
Life Cycle Management		
Approval	Yes	No
Update	Yes	Yes
Automatic Version Control	Yes • Versioning of metadata • Versioning of information artifacts	No
Deprecation	Yes • Prevents proliferation of obsolete information artifacts	No
Undeprecation	Yes	No
Deletion	Yes • Prevents accidental deletion of information artifacts that are in use	Yes
Taxonomy/Classification Support		
Predefined taxonomies	Yes	Yes
User-defined taxonomies	Yes	No
Taxonomy browsing and validation	Yes	No
Classification of artifacts	Yes	Yes
Classification of any metadata object	Yes	No
Relationship Support		
Predefined relationship types	Yes - Extensive	Yes - Very Limited
User-defined relationship types	Yes	No

Category/Feature	ebXML Registry 3.0	UDDI 3.0
Ability to relate any two objects in registry using any relationship type	Yes	No
Packaging/Grouping Support		
User-defined packages	Yes	No
Group any number of objects in same package	Yes	No
Group an object in multiple packages	Yes	No
Security Features		
Digital signature based authentication	Yes - Required	Yes - Optional. Most vendors do not support it.
Basic acess control based on predefined roles and predefined policies	Yes	Yes - Limited
User-defined, fined-grained acess control policies based on user-defined roles/groups	Yes • Based on XACML 1.0	No
Federated identity management and SSO	Yes • Based on SAML 2.0	No
Audit trail	Yes	Yes
Protocol Bindings Support		
HTTP binding (REST)	Yes • Allows any metadata or artifact to be addressable via an HTTP URL module access control	No
SOAP API binding	Yes	Yes

Category/Feature	ebXML Registry 3.0	UDDI 3.0
Advanced Features		
Information Management		
Metadata Validation	Yes • Unrestricted, may be used to validate any metadata type • XSLT-based, content-specific cataloging of XML artifacts • Extensible via custom validation services (requires programming)	No
Artifact Validation	Yes • Unrestricted, may be used to validate any artifact type • Extensible via custom validation services (requires programming)	No
Event Subscription and Notification		
Ability to select events using custom query	Yes • Can use any user-defined query (see Discovery features)	No
Content-based event notification	Yes • Can specify interest in specific types of content within specific types of artifacts	No
Delivery of notifications to registered Web service	Yes	Yes
Delivery of notifications to registered e-mail address	Yes	No
Federation Support		
Federated Queries	Yes	No
Object references between any object in one registry to any object in any other registry	Yes	No
Object replication from any registry to any other registry	Yes • Supports selective replication	Yes • All data replicated across all registries all the time

Category/Feature	ebXML Registry 3.0	UDDI 3.0
Client SDK Support		
JAXR API	Yes • JAXR level 0 and level 1 support	Yes • JAXR level 0 support only
Other Features		
Web Services Support	Yes • Several predefined, WSDL discovery-parameterized stored queries (such as "Find all WSDLs that use a specified namespace or name space pattern") • Automatic validation of WSDL upon publish to ensure compliance with WS-I Basic profile • Automatic cataloging of WSDL upon publish to support WSDL discovery	Yes • Limited to ~5 predefined discovery queries • No automatic validation of WSDL upon publish • No automatic cataloging, requires publisher to manually catalog the WSDL

Domain-Specific Profile Support	Yes
	• Extensibility features allow standard extensions to be defined for domain-specific use cases
	• Allows interoperability within and across domains
	• Examples of standard profile include:
	  • Web Services Profile
	  • WSRP Profile
	• Open **Geographic Information System** (GIS) Profile
	• Institute of Health Education (IHE)
	– XML Data Representation (XDR) Profile
	  • Health Level 7 (HL7) Conformance Profile
		No

Related information

[SOA] Service-Oriented Architecture
webservices.xml.com/pub/a/ws/2003/09/30/soa.html

[UDDI] UDDI
www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec

[ebRR] ebXML Registry Meta Links
ebxmlrr.sourceforge.net/tmp/ebXMLRegistryLinks.html

[SUNR] Sun's Service Registry
www.sun.com/products/soa/registry/© 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

© 2005 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California.

Sun, Sun Microsystems, the Sun logo, Java, and J2EE are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Mozilla and Netscape are trademarks or registered trademarks of Netscape Communications Corporation in the United States and other countries. AMD Opteron is a trademark or registered trademark of Advanced Micro Devices. Oracle is a xxx.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS HELD TO BE LEGALLY INVALID.

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN **Web** sun.com

©2005 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Java, and J2EE are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

09/05