OGSI, WSRF and Globus Toolkit 4

Thomas Sandholm <sandholm@pdc.kth.se>

KTH VETENSKAP OCH KONST

CENTER FOR PARALLEL COMPUTERS

Outline

- **Grid Middleware**
- Web Services
- Web Service based Grid Standards
- Globus Toolkit 4

Grid Middleware Features

- Middleware: Software designed to facilitate communication between clients and services offering capabilities such as compute resources
- Key features:
  - abstraction of resources (server->service)
  - interaction protocols (information and data retrieval)
  - information models (what is sent across the wire)
  - management capabilities (how can we control and track resource usage)
  - security models (how can we authenticate and authorize resource consumers and providers)

Grid Middleware Evolution

- Evolution of DCE, CORBA, DCOM, EJB
- Traditionally: main problem heterogeneity of
  - machines
  - operating systems
  - networks
  - programming languages
- Grid and large-scale distributed systems: heterogeneity of
  - interaction protocols
  - resource management policies
  - security models
  - programming models

Globus Middleware Evolution

- Globus 1 and Globus 2: de-facto approach, focus on portability
- Globus 3 and Globus 4: standard approach, focus on interoperability
- Introduction of service interfaces and a common protocol framework
- Leverage industry standardization efforts
- Multi-language support and open protocols
- Outsourcing of the core low-level interoperability layer
- Container technology from service provider community to reuse system-level components

Outline

- Grid Middleware
- **Web Services**
- Web Service based Grid Standards
- Globus Toolkit 4

Web Services in the Past

- Definition: A service on the web (WWW)
- Ubiquity Central
- Simple light-weight protocols that can be implemented easily
- Information search/retrieval focus
- Ad-hoc integration of services
- Security controlled with firewalls and end-to-end secure TCP-IP connections
- Very limited set of clients (web browser)
- Two-tier client-server model

Web Services Today

- Definition: "*A Web service is a software system designed to support interoperable* ***machine-to-machine interaction*** *over a network. It has an* ***interface described in a machine-processable format*** *(specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an* ***XML serialization*** *in conjunction with other Web-related standards" - W3C*
- Ubiquity and Interoperability Central
- Higher-level protocols to provide additional QoS
- Service composition and interaction focus
- Standard frameworks for integration of services
- Security model based on interaction context and payload contents and stakeholder policies
- A large variety of different clients, when a service is written the client is typically not known
- Multi-tier peer-to-peer model

Web Services Standards

- XML: basic markup and add on specs (W3C):
  - Namespaces
  - XSLT
  - XPATH, XQUERY
  - XML-Encryption, XML-Signature
- SOAP: XML based protocol framework (W3C)
  - Envelope with mandatory body (payload) and optional headers (out of band information)
  - Message exchange patterns (request-response, one-way etc)
- WSDL: Service interface and protocol specification (W3C)
  - Message based interactions
  - Service interfaces
  - Transport protocol definition
  - Endpoint access definition
- XML Schema (replaces DTD): Type Model Language
  - Information model

Web Services Standards Continued

- WS-SOAP-Security: Standard authentication
  - Ensuring integrity and privacy of SOAP payloads using XML-Signature and XML-Encryption
  - Extensible cryptography algorithm model
  - Message-level security (as opposed to end-to-end connection based to simplify asynchronous communication and intermediaries)

Web Services Industry Backing

- Industry leaders (such as IBM and Microsoft) keen on collaborating
- Large number of standard implementations and good tool support
- Extensibility and Community Specialization Key Drivers
- Economic/Business Driver: Industry is moving towards Open-Source solutions for the ubiquitous core infrastructure to boost market uptake and the added value of collaborating in outsourcing or peer-to-peer arrangements
- Strong industry know-how in economic markets, SLAs, QoS: great need in the next generation sustainable Grid

Outline

- Grid Middleware
- Web Services
- **Web Service based Grid Standards**
- Globus Toolkit 4

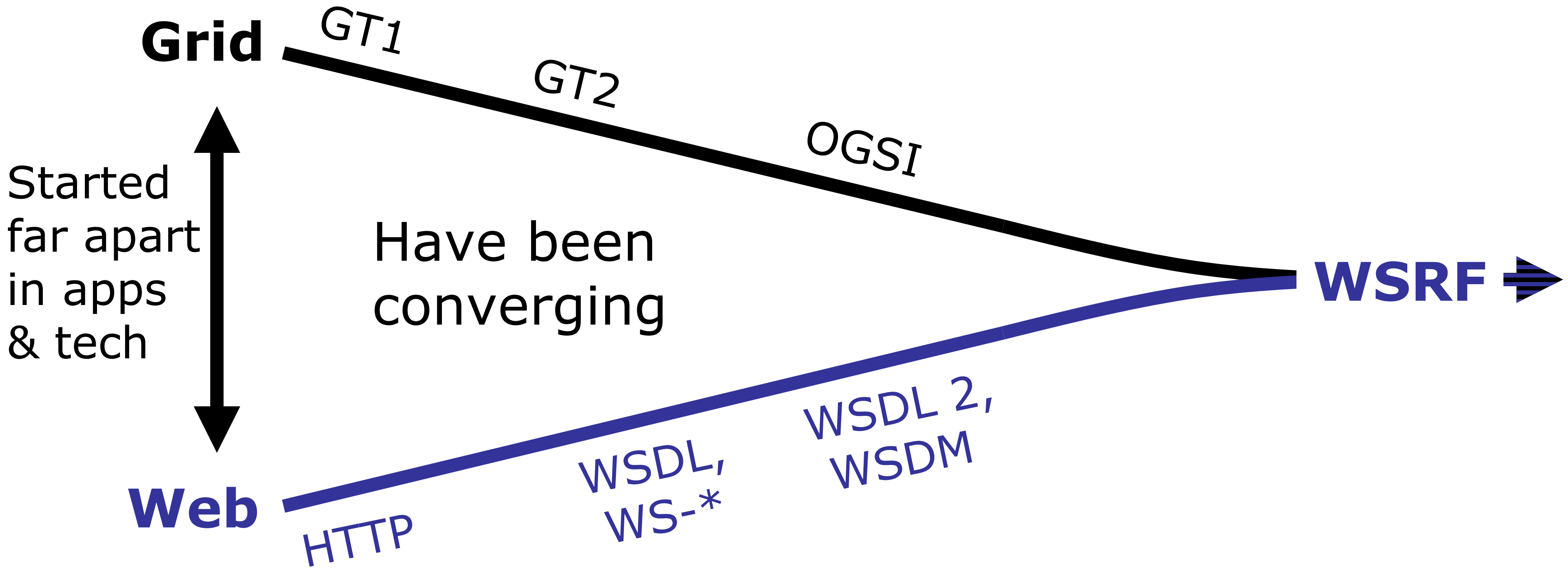
Web Services Gaps

- Light-weight flexible discovery and introspection
- Fine-grained control over stateful resources
- Dynamic remote deployment and sandboxing
- Standard notification (pub/sub) messaging model
- Base service capabilities to be used by meta-level tools
- Standard Faults hierarchy
- Arbitrary light-weight hosting environment
- Ability to dynamically negotiate WSDL binding such as transport mechanism
- Resource virtualization
- Security:
  - Cross security-domain communication
  - Single sign-on
  - Scalability of access control
  - Delegation of Privileges

Specification History

- From Open Grid Services Infrastructure to Web Services Resource Framework
  - 2/2002: IBM & Globus introduced OGSI draft
  - 6/2002: GGF OGSI WG first meeting
  - 6/2003: OGSI v1.0 completed
  - 1/2004: WSRF drafts introduced
    - Authored by a small set of interested parties
    - Based on concerns expressed about OGSI (OGSI v2.0)
    - Input to ongoing standardization effort of the functionality embodied in OGSI
  - 3/2004: Revised WSRF drafts released
  - 4/2004: OASIS TCs formed

Convergence of Grid and Web Services



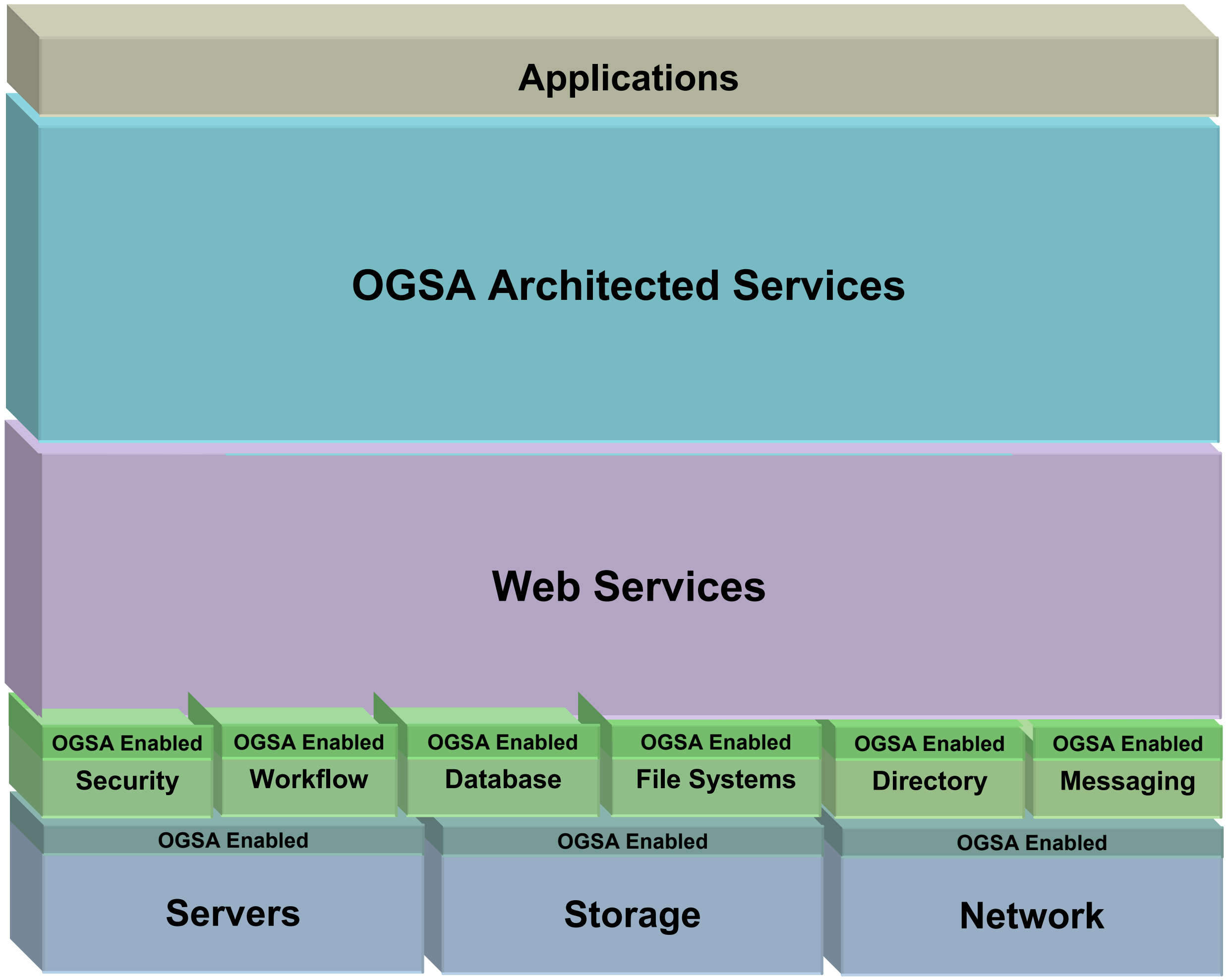
The definition of WSRF means that Grid and Web communities can move forward on a common base

Open Grid Services Architecture

- Define a service-oriented architecture …
  - the key to effective virtualization
- … to address vital “Grid” requirements
  - AKA utility, on-demand, system management, collaborative computing
- … building on Web services standards
  - extending those standards where needed

OGSA and Web Services

- OGSA Services can be defined and implemented as Web services
- OSGA can take advantage of other Web services standards
- OGSA can be implemented using standard Web services development tools
- Grid applications will NOT require special Web services infrastructure



OGSI Overview

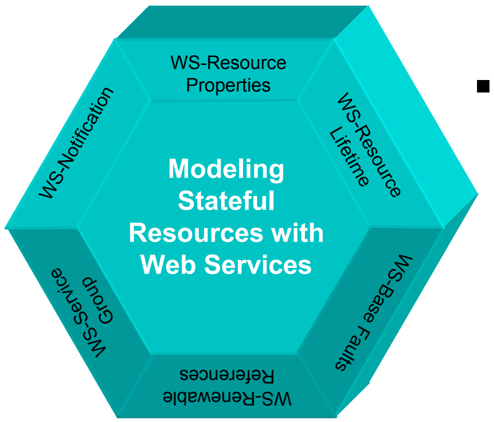
- Naming and bindings (basis for virtualization)
  - Every service instance has a <u>unique name</u>, from which can discover <u>supported bindings</u>
- Lifecycle (basis for fault resilient state management)
  - Service instances created by <u>factories</u>
  - Destroyed <u>explicitly</u> or via <u>soft state</u>
- Information model (basis for monitoring & discovery)
  - <u>Service data</u> (attributes) associated with GS instances
  - Operations for <u>querying</u> and <u>setting</u> this info
  - Asynchronous <u>notification</u> of changes to service date
- Service Groups (basis for registries & collective svcs)
  - Group membership rules & membership management
- Base Fault type

WSRF & WS-N Overview

- Naming and bindings (basis for virtualization)
  - Every resource can be uniquely referenced, and has one or more associated services for interacting with it
- Lifecycle (basis for fault resilient state management)
  - Resources created by services following factory pattern
  - Resources destroyed immediately or scheduled
- Information model (basis for monitoring & discovery)
  - Resource properties associated with resources
  - Operations for querying and setting this info
  - Asynchronous notification of changes to properties
- Service Groups (basis for registries & collective svcs)
  - Group membership rules & membership management
- Base Fault type

Web Services and Stateful Resources

- "State" appears in almost all applications
  - Data in a purchase order
  - Current usage agreement for resources
  - Metrics associated with work load on a server
- There are many possible ways Web services might model, access and manage state
  - OGSI v1.0 defined one approach
  - WS-Resource Framework proposes an evolution of that approach
  - Ad-hoc approaches can be used per-application



What is a WS-Resource?

- Web service: Operation execution component made available at an endpoint address
  - Implementation often stateless, but accesses state
- WS-Resource: Web service + associated resource
  - Equivalently: A resource with an associated WS
- A WS-Resource has:
  - Identity: Can be uniquely identified/referenced
  - Lifetime: Often created & destroyed by clients
  - State: Can be projected as an XML document
- WS-Resource type = Web service interface
- WS-Resources are not just for physical devices
  - Jobs, subscriptions, logical data sets, etc.

WS-ResourceProperties

- What (similar to OGSI v1.0 service data):
  - Portions of resource state are projected as a set of resource properties element
    - Modeled using standard XML Schema
  - WSDL portType attribute declares association between Web service and resource properties document
    - A WS-Resource’s type is determined by the interface of its Web service component
  - Standard operations for getting, setting, querying, and subscribing (via WS-Notification)
- Why:
  - Basis for standard resource inspection and monitoring

WS-ResourceProperties Operations

- Get

```
<wsrp:GetResourcePropertyRequest>
   QName
</wsrp:GetResourcePropertyRequest>
```

- Get Multiple

```
<wsrp:GetMultipleResourcePropertiesRequest>
   QName *
</wsrp:GetMultipleResourcePropertiesRequest>
```

- Query (XPath & extensible)

```
<wsrp:QueryResourcePropertiesRequest>
  <wsrp:QueryExpression dialect=”URI”>
     xsd:any
  </wsrp:QueryExpression>
</wsrp:QueryResourcePropertiesRequest>
```

WS-ResourceLifetime

- What (similar to OGSI v1.0):
  - Immediate, synchronous destruction operation
  - Time-based, scheduled destruction operation
    - "Soft-state" or "leased" lifetime management
    - Termination time not required to monotonically increase
    - Absolute time – idempotent (multiple client support)
  - Resource properties:
    - CurrentTime: Can be used to determine clock skew
    - TerminationTime: Current scheduled termination time
  - Notification of resource termination
- Why:
  - Commonality encourages tooling support

WS-Notification

- Subscriber indicates interest in a particular “Topic” by issuing a “subscribe” request
- Broker (intermediary) permits decoupling Publisher and Subscriber
- “Subscriptions” are WS-Resources
- Publisher need NOT be a Web Service
- Notification may be “triggered” by:
  - WS Resource Property value changes
  - Other “situations”
- Broker examines current subscriptions
- Brokers may
  - “Transform” or “interpret” topics
  - Federate to provide scalability

WS-ServiceGroup

- What (should be similar to OGSI v1.0):
  - Web service interfaces for representing and managing a by-reference collection of EPRs to Web services or WS-Resources
  - Each entry is member EPR + associated content
    - WS-RP used for representing the entries
    - Can have rules to membership and content
    - Members may be homogenous or heterogeneous, depending on the purpose and membership rules of the group
  - Has a registration interface for adding entries
    - Follows the WS-Resource factory pattern
    - The entry is represented as a WS-Resource
  - WS-RL used for removing entries
- Why:
  - Myriad of reasons for groups: E.g. Registries, collective operations, federated services, etc.

WS-BaseFaults Approach

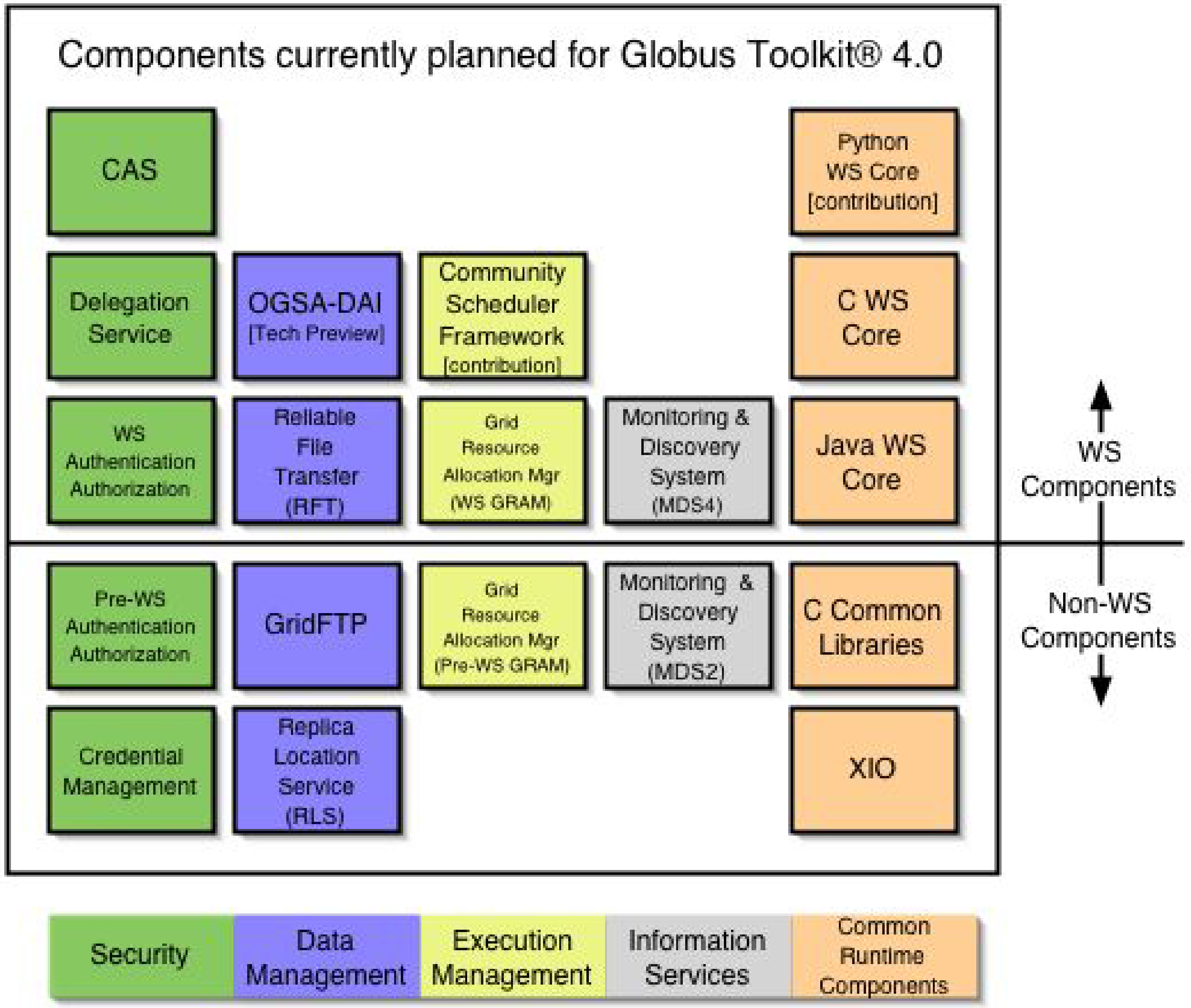
- Define base set of information that can appear in fault messages
- Convention for how to extend this base fault type for more specialized faults
  - Refine the type of the fault
  - Add information relevant to that refined fault type
- Convention for using these extended fault element as WSDL 1.1 fault messages

WS-Resource Framework Capabilities

- Clarifies how stateful resources are addressed
- Specifies how to use XML to describe and access a resource’s properties
- Defines how a resource is created and messages to destroy resources
- Provides a message subscription and notification mechanism for Web services
- Defines how to organize groups of resources and services
- Defines a standard, extensible format for Web services error messages
- In publicly released specifications

Outline

- Grid Middleware
- Web Services
- Web Service based Grid Standards
- **Globus Toolkit 4**


Components currently planned for Globus Toolkit® 4.0
CAS
Python WS Core [contribution]
Delegation Service
OGSA-DAI [Tech Preview]
Community Scheduler Framework [contribution]
C WS Core
WS Authentication Authorization
Reliable File Transfer (RFT)
Grid Resource Allocation Mgr (WS GRAM)
Monitoring & Discovery System (MDS4)
Java WS Core
WS Components
Non-WS Components
Pre-WS Authentication Authorization
GridFTP
Grid Resource Allocation Mgr (Pre-WS GRAM)
Monitoring & Discovery System (MDS2)
C Common Libraries
Credential Management
Replica Location Service (RLS)
XIO
Security
Data Management
Execution Management
Information Services
Common Runtime Components

WS Security

- Community Authorization Service
  - Manages access policies for VOs
  - Issues and verifies SAML based claims/assertions
  - Custom call-outs intercepting incoming message on the service side (E.g, in GridFTP server)
- Delegation Service
  - Transfer credentials to remote host
  - Separation from Handshake protocol
  - Delegation once per host as opposed to once per interaction/service
  - Credential renewal
- WS Authentication and Authorization
  - Message-Level Security (enc, sig)
  - SAML callout
  - Custom callouts
  - Grid-mapfile

WS Data Management

- OGSA DAI
  - University of Edinburgh contribution
  - Data Access and Integration
  - Querying and retrieving large amounts of data from various databases with a common interface
  - RDBMS, File system, XML DB
  - Streaming interface
  - GridFTP one possible transport mechanism
- Reliable File Transfer
  - Manage 3rd party GridFTP transfers
  - Submission of a set of transfer requests that are monitored and automatically restarted in case of failures

WS Execution Management

- Community Scheduler Framework
  - Platform Computing Contribution
  - Consistent interface into scheduling systems (LSF, PBS, SGE, LL)
  - Coordinates communication between schedulers
- GRAM
  - Grid Resource Allocation and Management
  - A set of WSRF services to locate, submit, monitor, manage remote jobs
  - Communication with a range of local schedulers
  - MPICH support
  - Multi-job and sub-job coordination
  - File staging

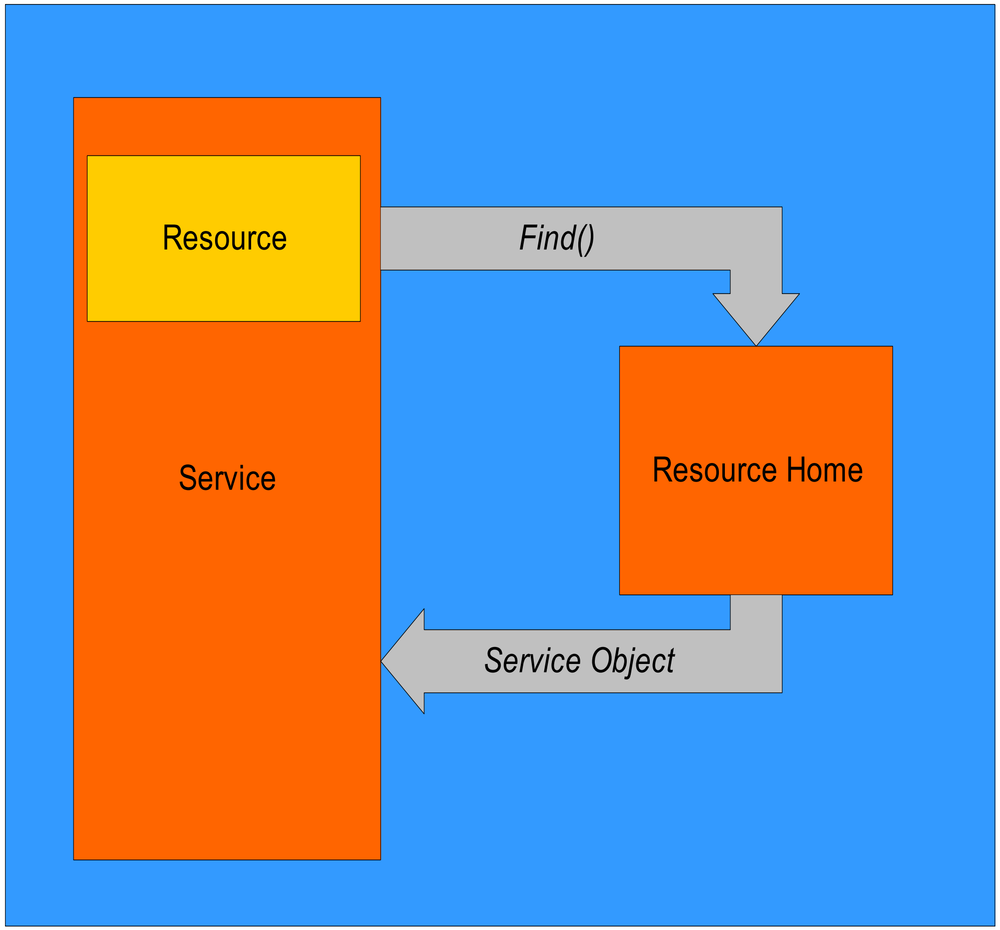
WS Information Management

- Monitoring and Discovery System (MDS4)
  - Index Service (GIIS replacement)
    - WSRF ServiceGroup implementation to discover Grid resource
    - Collect resource data on a VO basis
    - XPath query support
  - Trigger Service
    - Collects data from Grid resource and triggers events based on administrator defined rules
    - Email notifications when thresholds are reached
  - Aggregator Service
    - Plugin framework to aggregate data from multiple sources into a common sink
    - WSRF Service Group based
  - All services leverage WS Core WSRF Resource Properties (GRIS replacement)

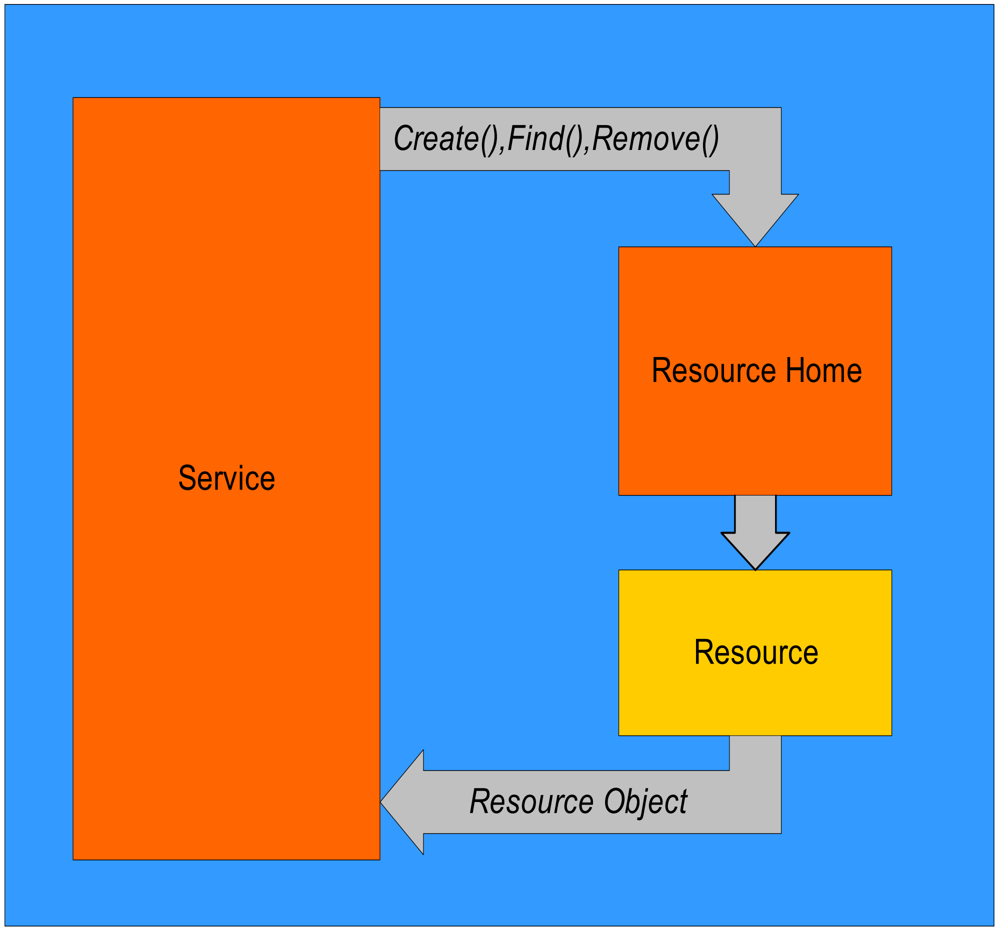
WS Core

- WSRF, WSN, and WS-Security implementations
- Python
  - LBNL contribution
  - Built on top of Globus C libraries
- C
  - Libxml2 based
  - Globus XIO plugin
  - WSDL Stub generator
- Java
  - Apache Axis (SOAP) and WSS4J (WS-Security) based
  - Light-weight standalone container and Tomcat container support

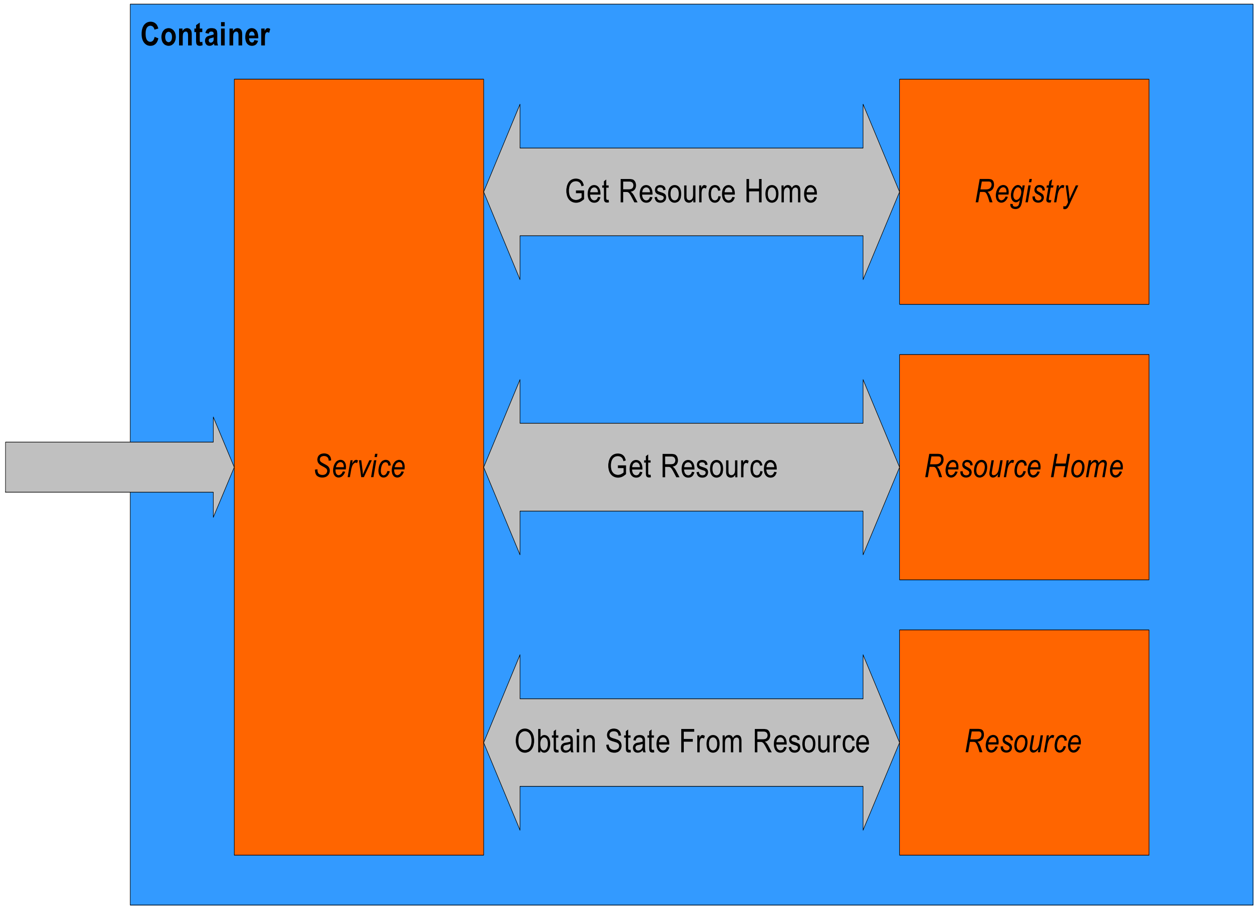
WS Core: Service as Resource



WS Core: Service + Resources



WS Core: Typical Resource Interaction



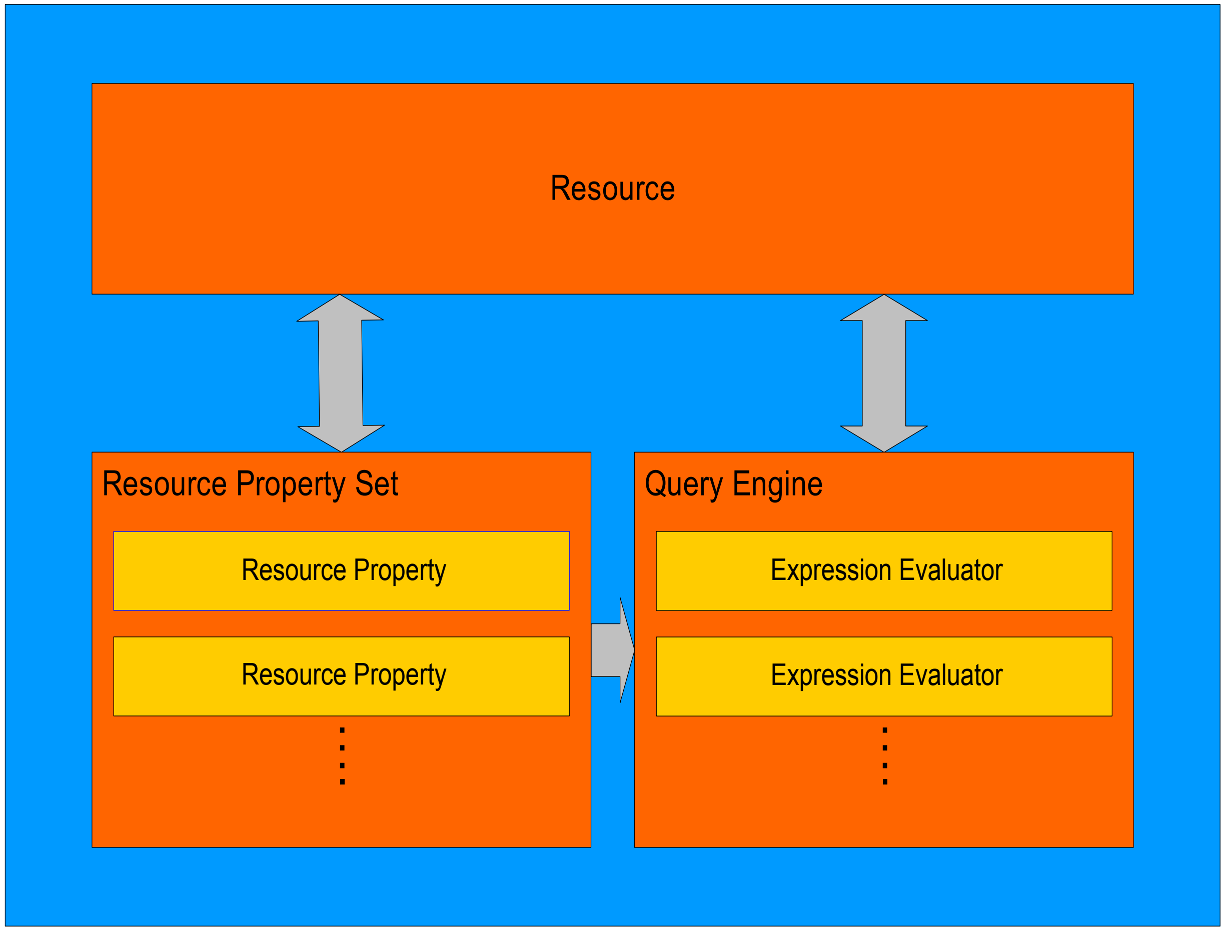
WS Core: Operation Provider Model

- Used in both GT3 and GT4
- Provides a web service composition framework
  - Enables reusable components
    - Currently implemented for GT4:
      - Destroy
      - Set Termination Time
      - Get Current Message
      - Notification Consumer
      - Pause/Resume Subscription
      - Subscribe
      - Get/Set Resource Property
      - Get Multiple Resource Properties
      - Query Resource Properties

WS Core: Resource Properties

- Resources implement the ResourceProperties interface
  - Accessor for the Resource Property Set
- The ResourcePropertySet manages properties
  - Add, remove, get, create, etc.
- Every ResourceProperty implements a interface
  - getName, add, remove, get, set, clear etc.
- Query Framework
  - Flexible dialect support

WS Core: Resource Overview



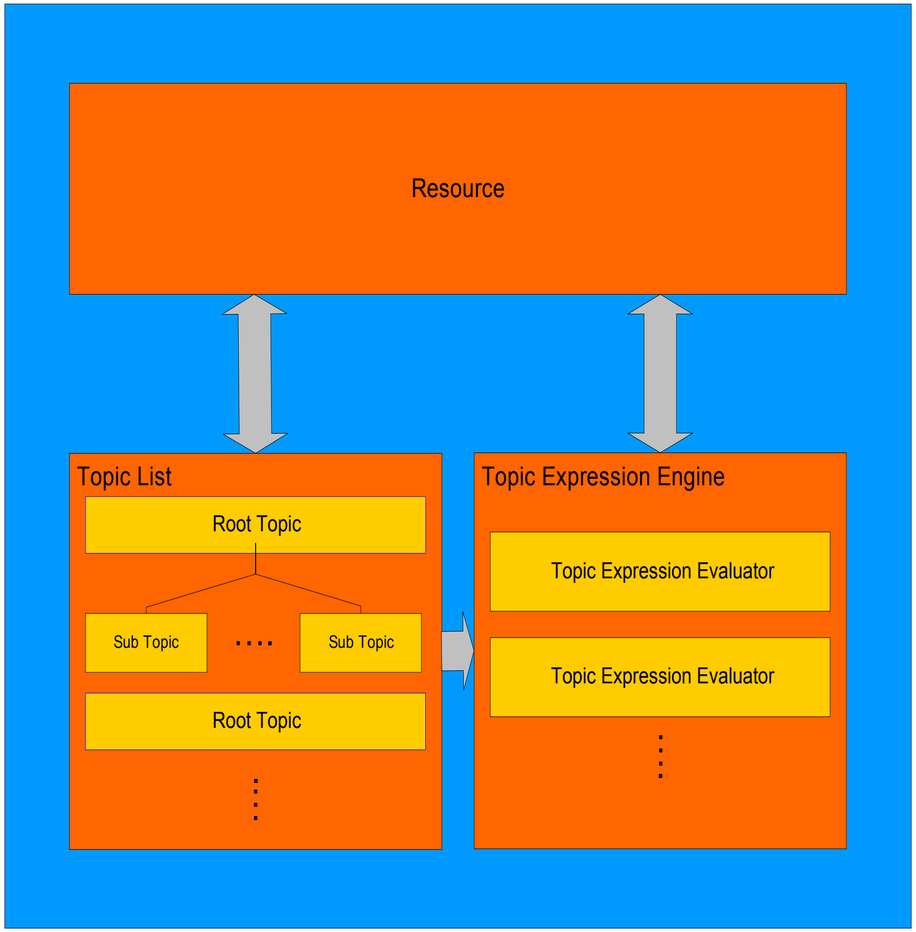
WS Core: Notification Service Interfaces

- TopicListAccessor
  - Allows for different TopicList implementations
  - Usually implemented by a Resource
- TopicList
  - List of root topics
- Topic
  - Represents a topic
  - May have child topics

WS Core: Notification Client Interfaces

- TopicListener
  - Interface for propagating Topic changes
  - Used to connect subscriptions to topics
  - Used for creating the topics Resource Property
- Subscription
  - Interface to subscription state

WS Core: Notification Overview



WS Core: Resource State Management

- Resource State Management
  - Will provide implementations for common patterns:
    - Soft references
      - Good when state is easily recreated
    - Resource as a data base entry
- Design your service with scalability in mind
  - Don't keep long lived references to your resources

WS Core: Security

- Model remains unchanged
  - Clients will have to set security properties on stub
  - Service/Resource security policy via deployment descriptor
    - Security settings will be per resource
- New Features in 4.0
  - Better GSI Secure Message support
    - Encryption
    - Replay Attack Prevention
  - Flexible Authorization Support
    - Based on Work in OGSA AuthZ WG and XACML TC
  - Rebase on Apache WSS4J code

WS Core: Directory/Registry

- Uses JNDI code from Apache Tomcat
  - Hierarchical
  - Object Factories
    - Resource Homes
    - DataSource
    - Etc.
  - Entries can be linked
  - For more information see http://jakarta.apache.org/tomcat/tomcat-5.0-doc/jndi-resources-howto.html

WS Core: Threads and Timers

- Based on J2EE APIs proposed by IBM & BEA
  - Royalty free
  - More information at http://www-106.ibm.com/developerworks/java/library/j-commonj-sdowmt/
- WorkManager & Timer interfaces
  - Using thread/timer pools
  - Container provides default WorkManager and Timer objects via JNDI lookup
- Replaces SweeperPool in GT3

GT4 Release Plan

- January 2005: Last Alpha Quality Release
- March 2005: Beta Release
- April 2005: Final Release