

Common Runtime Components: Key Concepts

Common Runtime Components: Key Concepts

Overview

The common runtime components provide GT4 web and pre-web services with a set of libraries and tools that allows these services to be platform independent, to build on various abstraction layers (threading, io) and to leverage functionality lower in the web services stack (WSRF, WSN, etc).

The components included in this category are the WS Core (Java, C and Python), C Common Libraries and XIO (extensible I/O). and pyglobus

Table of Contents

1. Conceptual Details	1
1. Containers	1
2. Web Services	1
2. Related Documents	6
1. Web Services	6

List of Figures

1.1. An abstract view of the various specifications that define the Web services architecture	2
1.2. WS Container. High-level picture of functional components commonly encountered in Web service implementations, showing the path taken by requests and responses.	3

Chapter 1. Conceptual Details

1. Containers

Users mostly refer to the WS Core as the container. To be more specific, when we use the term container, we mean the hosting environment for web services. In GT, there are web services containers available for Java, C, and Python (Python version is a contribution with little support at this time).

2. Web Services

We introduce basic concepts relating to Web services and their use and implementation within GT4, in particular within the "WS Core" (Java & C) components.

2.1. GT4, Distributed Systems, and Web Services

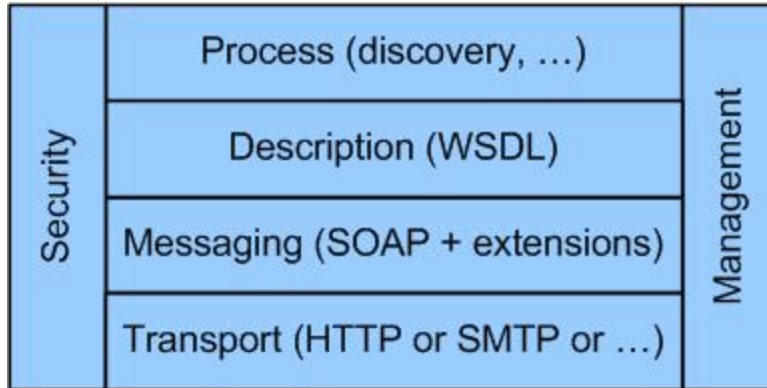
GT4 is a set of software components for building distributed systems: systems in which diverse and discrete software agents interact via message exchanges over a network to perform some tasks. Distributed systems face particular challenges relating to sometimes high and unpredictable network latencies, the possibility of partial failure, and issues of concurrency. In addition, system components may be located within distinct administrative domains, thus introducing issues of decentralized control and negotiation.

GT4 is, more specifically, a set of software components that (with some exceptions) implement Web services mechanisms for building distributed systems. Web services provide a standard means of interoperating between different software applications running on a variety of platforms and/or frameworks.

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.

Web services standardize the messages that entities in a distributed system must exchange in order to perform various operations. At the lowest level, this standardization concerns the protocol used to transport messages (typically HTTP), message encoding (SOAP), and interface description (WSDL). A client interacts with a Web service by sending it a SOAP message; the client may subsequently receive response message(s) in reply. At higher levels, other specifications define conventions for securing message exchanges (e.g., WS-Security), for management (e.g., WSDM), and for higher-level functions such as discovery and choreography. Figure 1 presents a view of these different component technologies; we discuss specific specifications below in [Section 2.4, "Web Services Specifications"](#).

Figure 1.1. An abstract view of the various specifications that define the Web services architecture



2.2. Service Oriented Applications and Infrastructure

Web services technologies, and GT4 in particular, can be used to build both service-oriented applications and service-oriented infrastructure. Deferring discussion of the sometimes controversial term "service-oriented" to later in [Section 2.9, "Service Oriented Architecture"](#), we note that a service-oriented application is constructed via the composition of components defined by service interfaces (in the current context, Web services): for example, a financial or biological database, an options pricing routine, or a biological sequence analyzer. Many descriptions of Web services and SOAP focus on the task of defining interfaces to such components, often illustrating their discussion with examples such as a "stock quote service" (the "hello world" of Web services).

Particularly when servicing many such requests from a distributed community, we face the related problem of orchestrating and managing numerous distributed hardware and software components. Web services can be used for this purpose also, and thus we introduce the term service-oriented infrastructure to denote the resource management and provisioning mechanisms used to meet quality of service goals for components and applications. Many GT4 features are concerned with enabling the construction of service-oriented infrastructure.

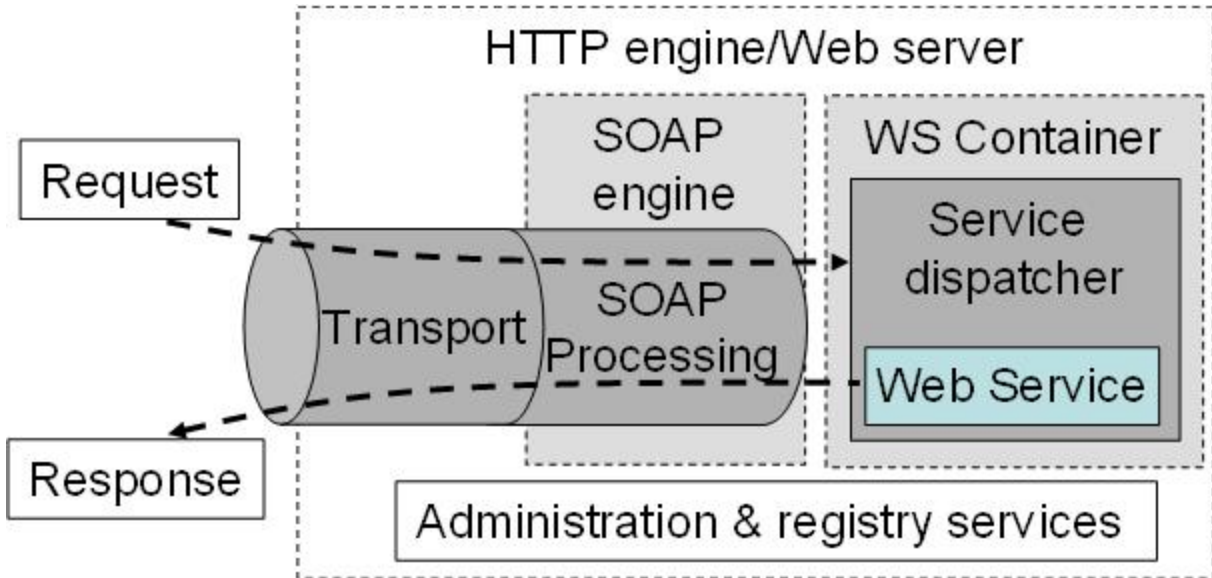
2.3. Web Services Implementation

From the client perspective, a Web service is simply a network-accessible entity that processes SOAP messages. Things are somewhat more complex under the covers. To simplify service implementation, it is common for a Web services implementation to distinguish between:

1. the hosting environment (or container), the (domain-independent) logic used to receive a SOAP message and identify and invoke the appropriate code to handle the message, and potentially also to provide related administration functions, and;
2. the Web service implementation, the (domain-specific) code that handles the message.

This separation of concerns means that the developer need only provide the domain-specific message handling code to implement a new service. It is also common to further partition the hosting environment logic into that concerned with transporting the SOAP message (typically via HTTP, thus an "HTTP engine" or "Web server"-sometimes termed an "application server") and that concerned with processing SOAP messages (the "SOAP engine" or "SOAP processor"). Figure 2 illustrates these various components.

Figure 1.2. WS Container. High-level picture of functional components commonly encountered in Web service implementations, showing the path taken by requests and responses.



Many different containers exist, with different performance properties, supported Web services implementation languages, security support, and so forth. We mention below those used in GT4.

2.4. Web Services Specifications

We provide pointers to the Web services specifications that underlie GT4. These comprise the core specifications that define the Web services architecture (XML, SOAP, WSDL); WS-Security and other specifications relating to security; and the WS-Addressing, WSRF, and WS-Notification specifications used to define, name, and interact with stateful resources. We also speak briefly to emerging specifications that are likely to be important in future GT evolution. An important source of information on the requirements that motivate the use and development of these specifications is the Open Grid Services Architecture.

2.5. XML, SOAP, WSDL

XML is used extensively within Web services as a standard, flexible, and extensible data format. In addition to XML syntax, other important specifications are XML Schema and XML Namespaces. Note that while current Web services tools typically adopt a textual serialization, a binary encoding is also possible and may provide higher efficiency.

SOAP 1.2 provides a standard, extensible, composable framework for packaging and exchanging XML messages between a service provider and a service requester. SOAP is independent of the underlying transport protocol, but is most commonly carried on HTTP.

WSDL 1.1 is an XML document for describing Web services. Standardized binding conventions define how to use WSDL in conjunction with SOAP and other messaging substrates. WSDL interfaces can be compiled to generate proxy code that constructs messages and manages communications on behalf of the client application. The proxy automatically maps the XML message structures into native language objects that can be directly manipulated by the application. The proxy frees the developer from having to understand and manipulate XML.

2.6. WS-Security and Friends

The WS-Security family of specifications addresses a range of issues relating to authentication, authorization, policy representation, and trust negotiation in a Web services context. GT4 uses a number of these specifications plus other related specifications, notably Security Authorization Markup Language (SAML), to address message protection, authentication, delegation, and authorization, as follows:

- TLS (transport-level) or WS-Security and WS-SecureConversation (message level) are used as message protection mechanisms in combination with SOAP.
- X.509 End Entity Certificates or Username and Password are used as authentication credentials.
- X.509 Proxy Certificates and WS-Trust are used for delegation.
- SAML assertions are used for authorization.

2.7. WS-Addressing, WSRF, and WS-Notification

A number of related specifications provide functionality important for service oriented infrastructure in which we need to be able to represent and manipulate stateful entities such as physical resources of various kinds, logical components such as software licenses, and transient activities such as tasks and workflows.

The WS-Addressing specification defines transport-neutral mechanisms to address Web services and messages. Specifically, this specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages.

The WS Resource Framework (WSRF) specifications define a generic and open framework for modeling and accessing stateful resources using Web services. This framework comprises mechanisms to describe views on the state (WS-ResourceProperties), to support management of the state through properties associated with the Web service (WS-ResourceLifetime), to describe how these mechanisms are extensible to groups of Web services (WS-ServiceGroup), and to deal with faults (WS-BaseFaults).

The WS-Notification family of specifications define a pattern-based approach to allowing Web services to disseminate information to one another. This framework comprises mechanisms for basic notification (WS-Notification), topic-based notification (WS-Topics), and brokered notification (WS-BrokeredNotification).

We note that the Web services standards space is in some turmoil due to competing proposed specifications. In particular, Microsoft and others recently proposed WS-Transfer, WS-Eventing, and WS-Management, which define similar functionality to WSRF, WS-Notification, and WSDM (discussed below), respectively, but using different syntax. We hope that these differences will be resolved in the future.

2.8. Other Relevant Specifications

The WS-Interoperability (WS-I) organization has produced a number of profiles that define ways in which existing Web services specifications can be used to promote interoperability among different implementations. The WS-I Basic Profile speaks to messaging and service description: primarily XML, SOAP, and WSDL. The WS-I Basic Security Profile speaks to basic security mechanisms. Other profiles are under development.

Web services distributed management (WSDM) specifications under development within OASIS are likely to play a role in future GT implementations as a means of managing GT components.

WS-CIM specifications under development within DMTF are likely to play a role in future GT implementations as a means of representing physical and virtual resources.

The Global Grid Forum's Open Grid Services Architecture (OGSA) working group has completed a document that provides a high-level description of the functionality required for future service-oriented infrastructure and applications, and a framework that suggests how this functionality can be factored into distinct specifications. The OGSA working group is now proceeding to define OGSA Profiles that, like WS-I profiles, will identify technical specifications that can be used to address specific Grid scenarios.

2.9. Service Oriented Architecture

We provide some additional discussion concerning the term service oriented architecture (SOA), which is used widely but not necessarily consistently within the Web services community. One common usage is simply to indicate the use of Web services technologies. However, the intention of those who coined the term seems to be rather to contrast two different styles of building distributed systems. Distributed object systems are distributed systems in which the semantics of object initialization and method invocation are exposed to remote systems by means of a proprietary or standardized mechanism to broker requests across system boundaries, marshal and unmarshal method argument data, etc. Distributed objects systems typically (albeit not necessarily) are characterized by objects maintaining a fairly complex internal state required to support their methods, a fine grained or "chatty" interaction between an object and a program using it, and a focus on a shared implementation type system and interface hierarchy between the object and the program that uses it.

In contrast, a Service Oriented Architecture (SOA) is a form of distributed systems architecture that is typically characterized by the following properties:

- **Logical view:** The service is an abstracted, logical view of actual programs, databases, business processes, etc., defined in terms of what it does, typically carrying out a business-level operation.
- **Message orientation:** The service is formally defined in terms of the messages exchanged between provider agents and requester agents, and not the properties of the agents themselves. The internal structure of an agent, including features such as its implementation language, process structure and even database structure, are deliberately abstracted away in the SOA: using the SOA discipline one does not and should not need to know how an agent implementing a service is constructed. A key benefit of this concerns so-called legacy systems. By avoiding any knowledge of the internal structure of an agent, one can incorporate any software component or application that can be "wrapped" in message handling code that allows it to adhere to the formal service definition.
- **Description orientation:** A service is described by machine-processable metadata. The description supports the public nature of the SOA: only those details that are exposed to the public and important for the use of the service should be included in the description. The semantics of a service should be documented, either directly or indirectly, by its description.
- **Granularity:** Services tend to use a small number of operations with relatively large and complex messages.
- **Network orientation:** Services tend to be oriented toward use over a network, though this is not an absolute requirement.
- **Platform neutral:** Messages are sent in a platform-neutral, standardized format delivered through the interfaces. XML is the most obvious format that meets this constraint.

It is argued that these features can allow service-oriented architectures to cope more effectively with issues that arise in distributed systems, such as problems introduced by latency and unreliability of the underlying transport, the lack of shared memory between the caller and object, problems introduced by partial failure scenarios, the challenges of concurrent access to remote resources, and the fragility of distributed systems if incompatible updates are introduced to any participant.

Web services technologies in general, and GT4 in particular, can be used to build both distributed object systems and service-oriented architectures. The specific design principles to be followed in a particular setting will depend on a variety of issues, including target environment, scale, platform heterogeneity, and expected future evolution.

Chapter 2. Related Documents

1. Web Services

- Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. Web Services Architecture. W3C, Working Draft¹

¹ <http://www.w3.org/TR/2003/WD-ws-arch-20030808/>