Web Services Agreement Specification (WS-Agreement)

Version 1.1

Draft 18

5/14/2004

Authors (alphabetically):
Alain Andrieux, (Globus Alliance / USC/ISI)
Karl Czajkowski, (Globus Alliance / USC/ISI)
Asit Dan (IBM)
Kate Keahey, (Globus Alliance / ANL)
Heiko Ludwig (IBM)
Jim Pruyne (HP)
John Rofrano (IBM)
Steve Tuecke (Globus Alliance / ANL)
Ming Xu (Platform Computing)

Abstract

This document describes Web Services Agreement Specification (WS-Agreement), an XML language for specifying an agreement between a resource/service provider and a consumer, and a protocol for creation of an agreement using agreement templates.

Status

This document is a draft of the WS-Agreement Specification from the Global Grid Forum (GGF). This is a public document being developed by the participants of the GRAAP Working Group (Grid Resource Allocation and Agreement Protocol WG) of the Scheduling and Resource Management (SRM) Area of the GGF.

Full Copyright Notice

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contact information at GGF website).

Table of Contents

1 Introduction

In a distributed service-oriented computing environment, service consumers like to obtain guarantees related to services they use, often related to quality of a service. Whether service providers can offer – and meet – guarantees usually depends on their resource situation at the requested time of service. Hence, quality of service and other guarantees that depend on actual resource usage cannot simply be advertised as an invariant property of a service and then bound to by a service consumer. Instead, the service consumer must request state-dependent guarantees to the provider, resulting in an agreement on the service and the associated guarantees. Additionally, the guarantees on service quality must be monitored and failure to meet these guarantees must to be notified to consumers. The objective of the WS-Agreement specification is to define a language and a protocol for advertising the capabilities of providers and creating agreements based on creational offers, and for monitoring agreement compliance at runtime.

TODO decide on naming for the responding party: “responder” or “agreement provider”?
signaling roles: “agreement initiator” / “agreement provider” (or “responder”?)
(service provisioning roles: “service consumer” / “service provider”)

An agreement between a service consumer and a service provider specifies one or more service level objectives both as expressions of requirements of the service consumer and assurances by the provider on the availability of resources and/or on service qualities. For example, an agreement may provide assurances on the bounds on service response time and service availability. Alternatively, it may provide assurances on the availability of minimum resources such as memory, CPU MIPS, storage, etc.

To obtain this assurance on service quality, the service consumer or an entity acting on its behalf must establish a service agreement with the service provider, or another entity acting on behalf of the service provider. Because the service objectives relate to the definition of the service, the service definition must be part of the terms of the agreement or be established prior to agreement creation. This specification provides a schema for defining overall structure for an agreement document. An agreement includes information on the agreement parties and references to prior agreements, referred to as agreement context, one or more discipline specific service definition terms, and one or more guarantee terms specifying service level objectives and business values associated with these objectives.

The agreement creation process typically starts with a pre-defined agreement template specifying customizable aspects of the documents, and rules that must be followed in creating an agreement, which we call agreement creation constraints. This specification defines a schema for an agreement template.

The creation of an agreement can be initiated by the consumer side or by the provider side, and the protocol provides hooks enabling such symmetry.

We use a coherent example of a hypothetical job submission to illustrate various aspects of the WS-Agreement specification, particularly relationship of service level objectives with service description, an agreement template specifying alternative service description terms and use of logical grouping operators, and agreement creation constraints in negotiating service level objectives. Details of the example scenario are described in section 2.

Section 3 introduces the layered model of WS-Agreement. Sections 4, 4.1, 4.2 specify the overall agreement structure, service description as agreement terms and guarantee terms, respectively. Section 5 specifies the schema for the agreement template and agreement creation constraints. Section 6 defines the compliance Section 7 introduces the port types and operations in the specification. Section 8 describes the process leading to the creation of an agreement.

1.1 Goals and Requirements

The goals of WS-Agreement are to standardize the terminology, concepts, overall agreement structure with types of agreement terms, agreement template with creation constraints and protocols for creation, negotiation and renegotiation of agreements, including WSDL needed to express the message exchanges and resources needed to express the state.

1.1.1 Requirements

In meeting these goals, the specification must address the following specific requirements:

- Must allow use of any service description term: It must be possible to create agreements for services defined by any domain specific service description terms, such as job specification, data service specification, network topology specification and web service description language (WSDL). Service objective description will reference the elements defined in service description.
- Must allow creation of agreements for existing and new services: It must be possible to create agreements for predefined services and resources modeling service state. Additionally, service description can be passed as agreement terms for coordinated creation of agreements and new service specific resources.
- Must allow use of any condition specification language: It must be possible to use any domain specific or other standard condition expression language in defining service level objectives and negotiability constraints.
- Must enable symmetry of protocol: A large number of scenarios are possible depending on whether a provider or consumer initiates agreement creation, and also where the agreement state is maintained. The basic messages defined in this document can be applied for modeling various usage specific scenarios.
- Must be composable with various negotiation models: it must be possible to design negotiation protocols based on WS-Agreement.
- Must be usable by itself: simple agreement creation must be supported in the WS-Agreement specification, independent of any negotiation model.
- Relationship to other WS-* specifications: WS-Agreement must be composable with other Web services specifications, in particular WS-Security, WS-Policy, WS-Federation, WS-Addressing, WS-Coordination, WS-

ResourceProperties, WS-ResourceLifetime, WS-Notification, Web Services for Remote Portals, and WS-ReliableMessaging and the WS-Resource framework [WS-Resource].

1.1.2 Non-Goals

The following topics are outside the scope of this specification:

- Defining domain-specific expressions for service descriptions.
- Defining specific condition expression language for use in specifying guarantee terms and certain negotiability constraints. We assume standards will emerge elsewhere for a common expression definition language. Alternatively, different expression language may be used in different usage domain.
- Defining specific service level objective terms for a specific usage domain such as network, server, applications, etc.
- Defining specification of metrics associated with agreement parameters, i.e., how and where these are measured.
- Defining a protocol and conventions for claiming domain-specific services according to agreements. For example, agreement identification in SOAP headers might suit a Web service, another mechanism is required for networking services, etc.
- Defining a general protocol for negotiating agreements.

1.2 Notational Conventions

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [RFC 2119].

When describing abstract data models, this specification uses the notational convention used by the [XML Infoset]. Specifically, abstract property names always appear in square brackets (e.g., [some property]). When describing concrete XML schemas, this specification uses the notational convention of [WS-Security]. Specifically, each member of an element’s [children] or [attributes] property is described using an XPath-like notation (e.g., /x:MyHeader/x:SomeProperty/@value1). The use of {any} indicates the presence of an element wildcard (<xsd:any/>). The use of @{any} indicates the presence of an attribute wildcard (<xsd:anyAttribute/>).

1.3 Namespace

This is an XML or other code example:

```
http://www.ggf.org/namespaces/ws-agreement (Code)
```

The following namespaces are used in this document:

Prefix	Namespace

wsag	http://www.ggf.org/namespaces/ws-agreement (temporary)
wsa	http://schemas.xmlsoap.org/ws/2004/03/addressing
wsbf	http://www.ibm.com/xmlns/stdwip/web-services/WS-BaseFaults
wssg	http://www.ibm.com/xmlns/stdwip/web-services/WS-ServiceGroup
wsrp	http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties
xs/xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
wsdl	http://schemas.xmlsoap.org/wsdl/

TODO: we need a “Terminology and Concepts” section (just like in WSRF) to define terms such as: signaling roles: agreement initiator / agreement provider (instead of "responder"?), application service-related roles: service consumer / service provider, “template”, “guarantee”, etc…

2 Example Scenarios

WS-Agreement covers a wide scope of application scenarios relating to the establishment of an agreement between a service provider and a service consumer. This is achieved by using a single document format and a protocol comprising few states. Two examples are chosen here to illustrate the range of applications that this specification covers. These examples are referred to throughout the specification.

2.1 Job submission

A typical application scenario is the request for executing a computing job. A service provider may post an agreement template available to interested requesters. In this scenario, the agreement template defines the list applications to be executed, and the software execution environment typically specified in a job submission. Service consumers are given a quality of service guarantee in terms of number of nodes and/or per node memory and storage for a specific time period. Alternatively, the guarantees can be on the completion time. A service consumer requesting a submitted job must fill in the name of the application to be executed, input and output files. In addition, a service consumer chooses the number of nodes (or any other resource requirements) that the application is guaranteed to execute on.

To submit a job, a service consumer retrieves the template from the provider, selects the application name, and provides URL of the input and output files as well as the details of resource guarantees. The filled template is sent as an offer to the provider. The provider decides whether to accept or reject the requested job. This may depend on the queue of jobs waiting to be processed and the current allocation of resources. The service provider answers the offer with a confirmation or a fault. In due time, the service provider processes the job and writes the output file to the URL defined in the agreement.

2.2 Service Parameterization

In the second scenario, the service contracted is an authentication and access control service. The service exposes an interface to register a new user, set an

access control policy, manage a user's passwords, authenticate a user and check a requested user action against the corresponding access control policy. In an access control environment, quality of service aspects such as response to for access verification and service availability is critical. Depending on particular needs, service consumers require different service quality levels and are prepared to pay differently for their quality of service requirements.

The service is very convenient for event organizers or other temporary projects. For example, sports events such as an athletics meeting or a soccer tournament require access control services for a limited amount of time to a large and diverse group of constituents such as athletes, journalists, jurors, and spectators who access the event's Web site or applications.

A service provider offers an agreement template describing the service and its guarantees, including the options available to the customer. The service description includes the WSDL of the service interface. Customer can choose among a service using Kerberos-based authentication or a proprietary authentication system. Furthermore, customers can choose how many users ID should be managed. Customers can add availability and response time guarantees to individual operations of the interface, e.g., to distinguish quality requirements for management and access control operations. For operation availability, customers choose between 95%, 98%, 99%, and 99.9%, defined as receiving a reply in 15 seconds. For average response time guarantees, customers choose between 0.5, 1 or 2 seconds, and set the number of operations per minute for which the response time goal must hold. Also, customers can set the time when the service will be available.

This template offers many options to service consumers. Service consumers send a completed offer to the service provider. Based on capacity limitations, the provider may accept the agreement creation offer or reject it. For example, if a service consumer asks for 1 sec response time for up to 1000 requests per minute, the provider might only have capacity for up to 500 requests.

If the agreement offer is accepted by the provider, the provider provisions the service and exposes status information on guarantee compliance to the user.

3 Layered Model

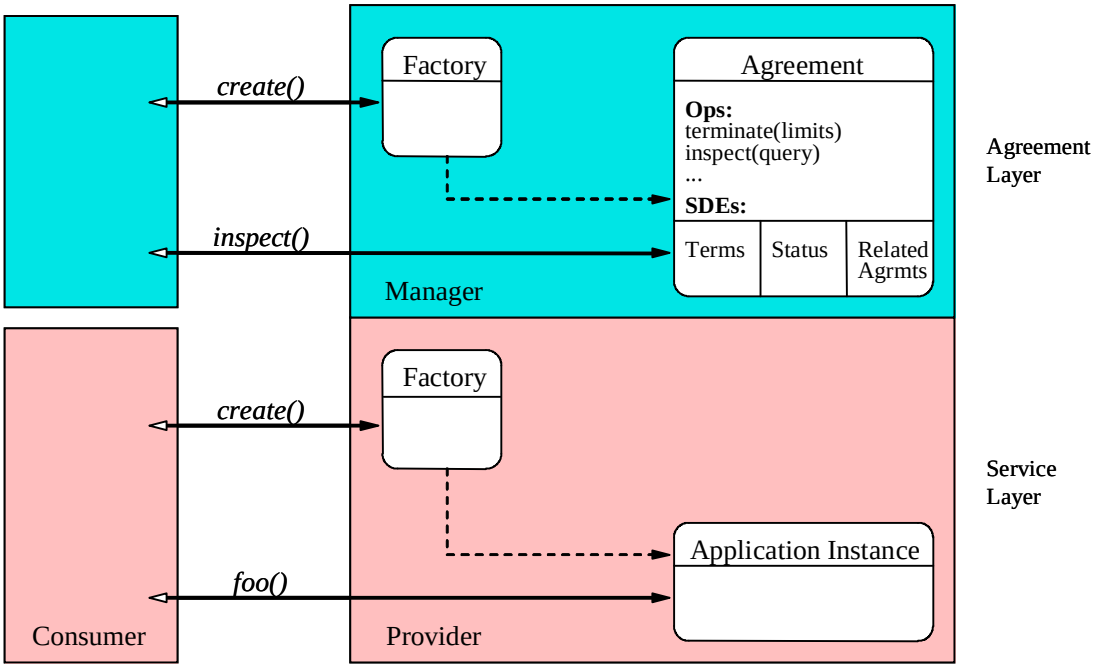


Figure 1: WS-Agreement Conceptual Layered Service Model.
Note: The names of the different operations and "attributes" are not normative.

The conceptual model for the architecture of WS-Agreement-based system interfaces has two layers (see figure 1), which are from bottom to top:

1. The service layer represents the application-specific layer of business service being provided. The class of provided service MAY or MAY NOT be exposed as a Web service interface. For instance, computational jobs may be virtualized as Web service instances, but other legacy services may not be referable as separate instances, let alone be exposed as Web services. Network availability can be seen as a class of service with no Web service representation, but it can be useful to manage its controllable Quality of Service (QoS) characteristics via agreements defined at layers above the service layer.

   The interface to this layer is domain-specific. This layer MAY be exposed as Web services. If it is, it SHOULD expose port types such as:

   - An application domain-specific service port type virtualizes the concrete service(s) being performed by the provider. It exposes domain-specific operations. For instance the virtualization of a file transfer service into a FileTransfer port type could expose operations such as "suspend", "resume", etc. In addition is can expose domain-specific state that the client (which can be a different actor than the initiator) can query or monitor. For instance a FileTransfer port type could expose a "bytesTransfered" resource property.

- A service is created by a service factory which creation operation takes a set of domain-specific parameters as arguments. For instance: createFileTransferService(sourceURL, destinationURL, ...).

2. The agreement layer provides a Web service-based interface that can be used to represent and monitor agreements with respect to provisioning of services implemented in the service layer.
   The agreement layer has the following port types, as detailed in the [WS-Agreement] specification:
   - An agreement port type, without any operation other than getters for state and metadata of the agreement such as the terms, the context, etc….
   - An agreement factory exposes an operation for creating an agreement out of an input set of terms. It returns an EPR to an Agreement service. The agreement factory also exposes resource properties such as the templates of offers acceptable for creation of an agreement.

     The creation parameters can be defined independently of the domain-specific agreement terms defined at the agreement layer. What is merely needed is an unambiguous mapping between the two. The binding between the agreement and the domain-specific service(s) it manages MUST be described in the agreement, and can take alternative forms:
     a. Existing services MAY be referenced by the agreement as part of its terms (thus, these references can be negotiated if it makes sense).
     b. Services MAY be created as per agreement, i.e. the agreement implementation has control over service (instance) creation with the agreement describing the behavior of the newly created service.
     c. Services MAY be created externally but bear domain-specific identifiers enabling the binding of a particular agreement. For instance an agreement on the bandwidth of a computer network can refer to network-specific metadata (such as fields in message headers) as a way to state QoS guarantees on specific network traffic.

The Agreement port type MAY also virtualize the domain-specific service being provided, although the decision to design it as such would depend on the desired strength of the coupling between the agreement and the service.

Because of the multiple possibilities in terms of design of a WS-Agreement system, domain-specific and application-specific decisions SHOULD be made in terms of composition of operation and port type design that cannot be mandated by this specification. This document specifies canonical factories and port types for the agreement layer. Designers of WS-Agreement services MAY reuse WSDL port types, operations, messages, and input/output types specified here although they will always have to define the binding between the agreement and service layer, which is domain-specific.

4 Agreement Structure

An agreement is conceptually composed of several distinct parts. We summarize the structure in the following diagram:

Figure 3: Structure of an agreement.

The section after the (optional) name is the context, which contains the meta-data describing the agreement as a whole. It names the participants in the agreement, the agreement's lifetime and links to other agreements related to this agreement. The next section contains the terms that describe the agreement itself.

The XML representation of an agreement or an agreement creation offer has the following structure:

```
<wsag:Agreement>
   <wsag:Name>
       xs:NCName
   </wsag:Name> ?
   <wsag:AgreementContext>
      wsag:AgreementContextType
   </wsag:AgreementContext>
   <wsag:Terms>
       wsag:TermCompositorType
   </wsag:Terms>
</wsag:Agreement>
```

The following describes the attributes and tags listed in the schema outlined above:

/wsag:Agreement

This is the outermost document tag which encapsulates the entire agreement. An agreement contains and agreement context and a collection of agreement terms.

/wsag:Agreement/Name

This is an OPTIONAL name that can be given to an agreement

/wsag:Agreement/AgreementContext

This is a REQUIRED element in the agreement and provides information about the agreement that is not specified in the terms such as who the involved parties are, what the services is that is being agree to, the length of the agreement, and references to any related agreements.

/wsag:Terms

The terms of an agreement comprises one or more service definition terms, and zero or more guarantee terms grouped using logical grouping operators.

Agreement Context

An agreement is scoped by its associated context that SHOULD include parties to an agreement, and additionally, SHOULD include reference to the service(s) provided in support of the agreement. The context MAY also include other prior and/or related agreements. The new agreement thus augments prior related agreements, between the service consumer and the service provider.

The wsag:AgreementContext element is used to describe the involved parties and to identify the service that the agreement is about. It can also optionally contain references to other related agreements.

```
<wsag:AgreementContext xsd:anyAttribute>

  <wsag:AgreementInitiator>xs:anyType</wsag:AgreementInitiator> +

  <wsag:AgreementProvider>xs:anyType</wsag:AgreementProvider> +

  <wsag:AgreementInitiatorIsServiceConsumer>
      xsd:boolean
  </wsag:AgreementInitiatorIsServiceConsumer> +

  <wsag:TerminationTime>xs:DateTime</wsag:TerminationTime> +

  <wsag:RelatedAgreements>
      <wsag:RelatedAgreement wsag:RelationshipType=”wsag:dependency”>
          <wsag:RelatedAgreementEPR>
              wsa:EndpointReferenceType
          </wsag:AgreementEPR>
      </wsag:RelatedAgreement> *
  </wsag:RelatedAgreements> +
```

```
  <xsd:any/> *
</wsag:AgreementContext>
```

The following describes the attributes and tags listed in the schema outlined above:

/wsag:AgreementContext

This is the outermost tag which encapsulates the entire agreement context

/wsag:AgreementContext/AgreementInitiator

This optional element identifies the initiator of the agreement creation request. It MAY be a URI or a wsa:EndpointReference from WS-Addressing or MAY identify the initiator by a more abstract type of naming, e.g. by security identity of the owner or operator.

/wsag:AgreementContext/AgreementProvider

This optional element identifies the provider of the agreement, i.e. the entity that responds to the agreement creation request. It MAY be a URI or a wsa:EndpointReference from WS-Addressing or MAY instead identify the provider by a more abstract type of naming, e.g. by security identity of the owner or operator.

/wsag:AgreementContext/AgreementInitiatorIsServiceConsumer

This element of type xsd:boolean MAY appear. If it is absent or empty, its default value is “true”.

- If it is “true”, the agreement initiator MUST be viewed as the consumer of the service and the agreement provider MUST be viewed as the provider of the service when interpreting the agreement terms.
- If it is “false”, the mapping of the signaling roles to the service provisioning roles are reversed, i.e. the initiator MUST be viewed as the service provider and the agreement provider MUST be viewed as the service consumer.

/wsag:AgreementContext/TerminationTime

This optional element specifies the time at which this agreement is no longer valid. Agreement initiators MAY use this mechanism to specify an Agreement service lifetime. Extended negotiation languages MAY define other mechanisms to negotiate lifetime integrated with other negotiation terms. The resulting negotiated lifetime MUST be exposed as wsag:TerminationTime and further negotiation MUST be possible through the basic [WS-ResourceLifetime]mechanisms.

/wsag:AgreementContext/RelatedAgreements

This element defines a list of any number of related agreements. The related agreements are represented in the agreement service as related agreement services (see the port type section of this document). This element MUST appear; however it MAY be empty.

We need discussion about the kind of meta-information that each related agreement should be associated with. In particular: do we need to define canonical types of relationships in the spec like we did before, such as dependency, composition... I am referring to the different types of related agreements that started to be identified in OGSI-Agreement (back in the day...). So we may need to restart a discussion that we dropped on those types of related agreements.

We may want to have a placeholder for domain-specific relationship type such as "advance reservation"...? Which makes me think that such as thing as "reservationID" could be a term more than a part of the context....

Also, OGSI-Agreement says relationship can be modified at run-time: that is not in favor of specifying the related agreements in the context. Or maybe those are only for relationships which cannot change during the lifecycle of the agreement?

/wsp:AgreementContext/{any}

Additional child elements MAY be specified to make additional agreement contexts but MUST NOT contradict the semantics of the parent element; if an element is not recognized, it SHOULD be ignored.

/wsp:AgreementContext/@{anyAttribute}

Additional attributes MAY be specified but MUST NOT contradict the semantics of the owner element; if an attribute is not recognized, it SHOULD be ignored.

A wsag:Context element of type wsag:AgreementContextType MAY be used in an agreement to define an agreement context. Alternatively, the agreement context MAY be specialized, through derivation of the wsag:AgreementContextType Schema type in order to define other attributes of the parties or services engaged in an agreement.

4.2 Agreement Terms

The terms of an agreement are wrapped by a wsag:Terms term compositor.

We define two types of terms: service description terms and guarantee terms.

- The service description terms provide information needed to instantiate or otherwise identify a service to which this agreement pertains.
- The guarantee terms specify the service levels that the parties are agreeing to. Management systems may use the guarantee terms to monitor the service and enforce the agreement.

The specification defines schema for service description and agreement terms as abstract types that must be extended for specific usage domain.

4.2.1 Term Compositor Structure

Within the wsag:Terms compositor, special compositor elements can be used as logical AND/OR/XOR operators to combine terms. This enables the specification of alternative branches with potentially complex nesting within the terms of agreement.

TODO: discussion on this. Alternative only for an agreement offer and agreement itself should bear no alternatives?

The terms consist of one or more service definition terms and zero or more guarantee terms grouped using the logical grouping compositors.

The recursive structure of a term compositor, of type wag:TermCompositorType, is as follows:

```
<wsag:Terms>
    <wsag:All>
        wsag:TermCompositorType
    </wsag:All>       |
    <wsag:OneOrMore>
        wsag:TermCompositorType
    </wsag:OneOrMore> |
    <wsag:ExactlyOne>
```

```
        wsag:TermCompositorType
    </wsag:ExactlyOne> |
    {
        <wsag:ServiceDescriptionTerm>
            wsag:ServiceDescriptionTermType
        </wsag:ServiceDescriptionTerm> |
        <wsag:GuaranteeTerm>
            wsag:GuaranteeTermType
        </wsag:GuaranteeTerm>
    } *
</wsag:Terms>
```

The contents of a term compositor are described as follows:

/wsag:Terms/wsag:All (or wsag:OneOrMore, or wsag:ExactlyOne)

This is a logical AND (or OR, or XOR) operator of type wsag:TermCompositorType which is used to logically group terms and/or other compositors underneath it. This provides a recursive structure to the logical composition of terms.

/wsag:Terms/wsag:ServiceDescriptionTerm

These terms are OPTIONAL and MAY specify the parameters used to instantiate a service which will fulfill this agreement or to describe a service to be used by the agreement.

/wsag:Terms/wsag:GuaranteeTerm

These terms are OPTIONAL and MAY specify the guarantees (both promises and penalties) that are associated with the other terms in the agreement.

Service Description Terms

Service description terms (SDTs) are a fundamental component of an agreement: the agreement is about the service(s) - existing or not - described by the service description terms. The provisioning of this service may be conditional to specific run-time constraints, and additional service level objectives on how the service is performed may be imposed by the service guarantee; service terms define the functionality that will be delivered under an agreement. The service description content itself is dependent on the particular domain. A ServiceDescriptionTerm consists of three parts,

- The name of the ServiceDescriptionTerm.
- The name of the service being described partially or fully by the domain-specific part of this service description term. This allows for semantic grouping of service description terms that may not be structurally grouped together in the agreement.
- A domain-specific description of the offered or required functionality. This element MAY completely describe the service it is about, or it MAY do so only partially.

An Agreement MAY contain any number of SDTs, as an agreement can refer to multiple components of functionality within one service, and can manage several services.

4.2.2.1 Service Description Term Structure

The following definition describes the simple generic content of this type:

```
<wsag:ServiceDescriptionTerm
   wsag:Name="xs:NCName" wsag:ServiceName="xs:NCName">
   <xsd:any> … </xsd:any>
</wsag:ServiceDescriptionTerm>
```

The following describes the elements of the schema above:

/wsag:ServiceDescriptionTerm

ServiceDescriptionTerm encloses a description of a service or part of a service.

/wsag:ServiceDescriptionTerm/@Name

The name attribute (of type xs:NCName) represents the name given to a term. Since an Agreement MAY encompass multiple ServiceDescriptionTerms related to the same service each term SHOULD be given a unique name in order to make structural referencing of service description terms (for instance via XPATH) more convenient (see guarantee term section).

/wsag:ServiceDescriptionTerm/@ServiceName

This attribute identifies a service across multiple service description terms. The service description term is defined as “being about” the service identified by the wsag:ServiceName attribute. This identifier is scoped within the agreement i.e. it is not meant to identify the service outside of the agreement. This element is optional but SHOULD appear if several services are described by this agreement.

/wsag:ServiceDescriptionTerm/{xsd:any}

This element is a placeholder for a partial or full description of, and/or a reference to, the domain-specific service this service description term is about.

- This element is expressed using a domain-specific language that MAY be independent of WS-Agreement. Service description languages from different domains or specifications MAY be embedded inside distinct service description terms.
- This element MUST be defined as a global element in the XML schema where it comes from. WS-Agreement does not mandate any restriction on the name or type (which can be simple or complex) of this element.
- This element MAY refer to one or more aspects of functionality of the described service, as granularity of that functionality is a domain-specific concern.

4.2.3 Guarantee Terms

The primary motivation for creating a service agreement between a service provider and a service consumer is to provide assurance to the service consumer on the service quality and/or resource availability offered by the service provider. Guarantee terms define this assurance on service quality, associated with the service described by the service definition terms. In the job submission example, an agreement may provide assurance on the bounds (e.g., minimum) on the availability of resources such as memory, type of central processing unit (CPU), storage and/or job execution

beginning or completion time. These bounds are referred to as the service level objectives (SLO).

An expression of assurance also includes qualifying conditions on external factors such as time of the day as well as the conditions that a service consumer must meet. For example, a bound on the average response time of the authorization service (as per the second example) is assured only if the request rate is below a specified threshold during weekdays.

An assurance also includes specification of one more forms of business values associated with an SLO. For example, a business value may represent the strength of this commitment by the provider. Another example of business value is the importance of this assurance to the consumer and/or to the provider.

An agreement MAY contain zero or more Guarantee terms, where each GuaranteeTerm element consists of the following parts:

- ServiceScope: the list of services this guarantee applies to.
- Variables: aliases to concepts understood in the context of the agreement or to parts of it, used in qualifying conditions and service level objectives.
- QualifyingCondition: an optional condition that must be met (when specified) for a guarantee to be enforced.
- ServiceLevelObjective: an assertion expressed over service descriptions.
- BusinessValueList: one or more business values associated with this objective.

Note that a single ServiceLevelObjective MAY be a complex of objectives expressed as a complex condition expressing bounds over many service attributes. Meeting the overall objective MAY imply meeting all the individual objectives. However, if the business values associated with individual objectives are different, (for example, if not all objectives are equally important), then each objective SHOULD be expressed as a separate GuaranteeTerm. Similarly, a QualifyingCondition MAY be a complex condition if multiple qualifying conditions need to be met for a guarantee to be honored.

4.2.3.1 Guarantee Term Structure

A GuaranteeTerm has the following form:

```
<wsag:GuaranteeTerm wsag:ServiceScope="wsag:ListOfServiceNames">
    <wsag:Variables>…</wsag:Variables>
    <wsag:QualityingCondition>…</wsag:QualifyingCondition>?
    <wsag:ServiceLevelObjective>…</wsag:ServiceLevelObjective>
    <wsag:BusinessValueList>…</wsag:BusinessValueList>
</wsag:GuaranteeTerm>
```

/wsag:GuaranteeTerm

This element, of type GuaranteeTermType, represents an individual guarantee related to the service described in service description terms.

/wsag:GuaranteeTerm/@wsag:ServiceScope

This is a list of service names referring to the respective wsag:ServiceName attributes of one or more of the service description terms in this agreement. The guarantee applies to every service in the list.

Question: Should we define a special value ALL to refer to all services in the agreement or is that implied by an empty value?

/wsag:GuaranteeTerm/wsag:Variables

This element is a list of variables representing domain-specific concepts, for instance aspects of the service(s) this guarantee refers to. Variables are used in domain-specific assertions about the provisioning of the service(s), such as qualifying conditions and/or service level objectives.

/wsag:GuaranteeTerm/wsag:QualifyingCondition

This element MAY appear to express a precondition under which a guarantee holds.

/wsag:GuaranteeTerm/wsag:ServiceLevelObjective

This element, of type xsd:anyType, expresses the condition that must be met to satisfy the guarantee.

/wsag:GuaranteeTerm/wsag:BusinessValueList

This is the higher level element that contains a list of business value elements associated with a service level objective. Two standard business value types are defined later. Customized business value types can be expressed extending an abstract business value type, defined here.

The detailed description of the types associated with a GuaranteeTerm follows in the subsections.

4.2.3.2 Variables

Guarantees contain logical expressions that refer to aspects of the service(s) subject to the guarantee. For instance, metrics for availability and response time must refer to named concepts (availability, response time) and must be declared as named variables that can be used in assertions. The semantics of those variables must be defined to interpret the condition expression. Each individual variable has the following form:

```
<wsag:Variable wsag:Name="xsd:NCName" wsag:Metric="xsd:QName">
    <wsag:Reference>xsd:anyType<wsag:Reference>
</wsag:Variable>
```

/wsag:Variable/wsag:Location

The value of this element is a structural reference to a field of arbitrary granularity in the service description terms - including fields within the domain-specific service descriptions.

- This reference gives scope to the concept represented by the variable, i.e. the concept applies at the nesting level of the structural item that is referred.
- This reference MAY be an XPATH expression for instance to use with domain-specific service description languages that are based on XML. If this reference is an XPATH, it MAY be relative to the wsag:Terms section of the agreement document.

/wsag:Variable/@wsag:Name

This element, of type xsd:NCName, is the name of the variable and allows the concept represented by this variable to be used in assertions. The name of each variable MUST be unique within the list of variables declared in the containing guarantee description term.

/wsag:Variable/@wsag:Metric

This element, of type xsd:QName, is an identification of a domain-specific metric. This element is optional and intended for cases where the structural reference of the variable does not sufficiently explain the semantics and typing of a variable. The domain specification where the metric is defined MUST define a namespace and a local name for the metric, as well as its type in logical expressions.

Note: If an XML particle definition exists for the metric, and when a fixed value makes sense for the concept, a wsag:Guarantee is not necessary and the XML particle MAY instead be used inside a wsag: ServiceDescriptionTerm element in order to specify a fixed value.

Issue: should we make wsag:Metric required so that the variable value always has the same unambiguous Schema type in assertions?

Examples:

```
<wsag:Variable name="CPUcount" metric="job:numberOfCPUs">
    <wsag:Location>
//wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:ServiceDefinitionTerm/job:executable]
    <wsag:Location/>
</wsag:Variable>
```

In this example, we assume a computational job is specified in an agreement offer (or agreement template, or agreement). A variable “CPUcount” refers to the concept of “number of CPUs to be used for the job at execution time “, represented as a typed, globally-defined Schema particle “numberOfCPUs” in the namespace assigned to the prefix “job” (domain of computational jobs). “CPUCount” can be used in assertions that express limits, ranges or more complex relationships. Its scope of application is the ‘job:executable’ unique domain-specific term so as to distinguish it from the overall job specification, which may includes other directives such as file transfers.

```
<wsag:Variable wsag:Name="bandwidth"
wsag:Metric="job:networkBandwidth">
    <wsag:Location/>
//wsag:Agreement/wsag:Terms/wsag:All/wsag:ServiceDefinitionTerm[@wsag:Name='fileStageIn1']
    <wsag:Location/>
</wsag:Variable>
```

In this example, the variable “bandwidth” could be used in the qualifying condition of the guarantee term to express a precondition on the file transfer it refers to.

```
<wsag:Variable wsag:Name="duration" wsag:Metric="time:duration">
    <wsag:Location/>
//wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@
wsag:Name='fileStageIn1']
    <wsag:Location/>
</wsag:Variable>
```

In this example, the variable “transferTime” could be used to express a quality of service requirement (as a service level objective) on the file transfer it refers to. Note that the XPath expression enables to distinguish between several domain-specific terms of the same name (for instance to specify several file stage-in directives) as long as the wsag:Name given to the wrapping ServiceDescriptionTerm is unique.

Variables are grouped into a set:

```
<wsag:Variables>
   <wsag:Variable> … </wsag:Variable> *
</wsag:Variables>
```

/wsag:Variables

This element, of type VariableSetType, contains one or more variables.

/wsag:VariableSet/wsag:Variable

Variables are specified above.

4.2.3.3 Qualifying Condition and Service Level Objective

QualifyingCondition and ServiceLevelObjective are expressed as assertions over service attributes and/or external factors such as date and time. The type of both elements is xsd:anyType as a completely open content that can be extended with assertion languages which MAY designed independently of the WS-Agreement specification but which MUST address the requirements of the particular domain of application of the agreement.

An example of a generic assertion language can be found in [XQUERYX].

4.2.3.4 Business Value List

Associated with each ServiceLevelObjective is a BusinessValueList that contains multiple business values, each expressing a different value aspect of the objective. The values may express relative importance of this objective to a consumer or penalty to be assessed upon failure to meet this objective. Other customized domain specific business values can be defined and associated with a service level objective.

Expression of business value in meeting certain assurances and flexible specification of service consumer requirements may free a provider from fixed allocation of resources. A provider can dynamically allocate resources based on actual measured or estimated service consumer requirements, and evaluation of business values. For

example, a new arrival of a high priority job may result in reduction of allocated resources or suspension of an existing low priority job.

```
<wsag:BusinessValueList>
  <wsag:Importance> xsd:integer </wsag:Importance>?
  <wsag:Penalty> </wsag:Penalty>?
  <wsag:Reward> </wsag:Reward>?
  <wsag:BusinessValue> … </wsag:BusinessValue>*
</wsag:BusinessValue>
```

/wsag:BusinessValueList

This element comprises the set of business value expressions.

/wsag:BusinessValueList/wsag:Importance

This element when present expresses relative importance (defined below) of meeting an objective.

/wsag:BusinessValueList/wsag:Penalty

This element (defined below) when present expresses penalty to be assessed for not meeting an objective.

/wsag:BusinessValueList/wsag:Reward

This element (defined below) when present expresses reward to be assessed for meeting an objective.

/wsag:BusinessValueList/wsag:BusinessValue

Zero or more domain specific customized business values can be defined.

4.2.3.4.1 Importance

In many cases, all service level objectives (SLO) will not carry the same level of importance. It is necessary therefore, to be able to assign a “business value” in terms of relative importance to an objective so that its importance can be understood, and so tradeoffs can be made by the provider amongst various guarantees when sufficient resources are available. Absolute value of a guarantee on the other hand specifies business impact of meeting or violating an individual SLO, expressed via Reward and Penalty. Relative importance can be though of as a measure of importance with a default measurement unit.

Relative terms, such as high, low, medium, etc. can be used to prioritize across many guarantees. However, to provide stronger semantics and easier comparison of this value, this is expressed using an integer.

4.2.3.4.2 Penalty and Rewards

In business Service Level Agreements (SLAs), this importance is indirectly expressed by specifying the consequences of not meeting this assurance. Here, each violation of a guarantee term during an assessment window will incur a certain penalty. The penalty assessment is measured in a specified unit and defined by a value expression.

```
<wsag:Penalty>
    <wsag:AssesmentInterval>
```

```
        <wsag:TimeInterval>xsd:duration</wsag:TimeInterval> |
        <wsag:Count>xsd:positiveInteger</wsag:Count>
    </wsag:AssesmentInterval>
    <wsag:ValueUnit>xsd:string</wsag:ValueUnit>?
    <wsag:ValueExpr>xsd:any</wsag:ValueExpr>
</wsag:Penalty>
```

/wsag:Penalty

This element defines a business value expression for not meeting an associated objective.

/wsag:Penalty/wsag:AssesmentInterval

This element defines the interval over which a penalty is assessed.

/wsag:Penalty/wsag:AssesmentInterval/wsag:TimeInterval

This element when present defines the assessment interval as a duration.

/wsag:Penalty/wsag:AssesmentInterval/wsag:Count

This element when present defines the assessment interval as a service specific count, such as number of invocation.

/wsag:Penalty/wsag:ValueUnit

This element defines the unit for assessing penalty, such as USD. This is an optional element since in some cases a default unit MAY be assumed.

/wsag:Penalty/wsag:ValueExpr

This element defines the assessment amount, which can be an integer, a float or an arbitrary domain-specific expression.

Alternatively, meeting each objective generates a reward for a provider. The value expression for reward is similar to that of penalty.

5 Agreement Template and Creation Constraints

In order to create an agreement, a client makes an agreement creation offer to an agreement factory. An agreement creation offer has the same structure as an agreement. The agreement factory advertises the types of offers it is willing to accept by means of agreement templates.

An agreement template is composed of three distinct parts. We summarize the structure in the following diagram:

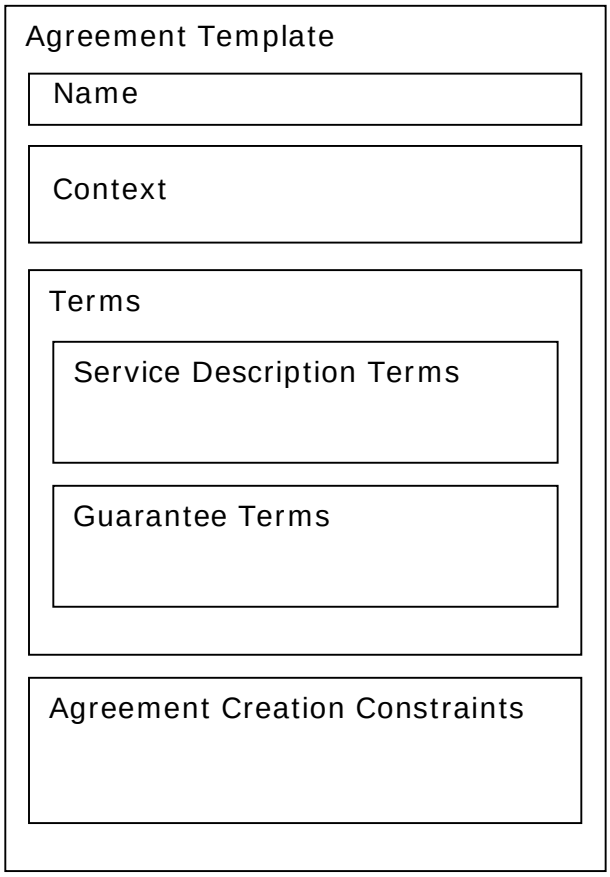


Figure 4: Structure of an agreement template.

The structure of an agreement template is the same as that of an agreement, but an Agreement template MAY also contain a creation constraint section, i.e. a section with constraints on possible values of terms for creating an agreement. The constraints make it possible to specify the valid ranges or distinct values that the terms may take. The constraints refer back to individual terms they apply to using XPATH.

The contents of an agreement template are of the form:

```
<wsag:template>
   <wsag:Name>
       xs:NCName
```

```
   </wsag:Name> ?
   <wsag:AgreementContext>
      wsag:AgreementContextType
   </wsag:AgreementContext>
   <wsag:Terms>
       wsag:TermCompositorType
   </wsag:Terms>
   <wsag:CreationConstraints>
       …
   </wsag:CreationConstraints> ?
</wsag:template>
```

The following describes the contents of the agreement template:

/wsag:template

This is the outermost document tag which encapsulates the entire agreement template. An agreement template contains and agreement context template and a collection of possible agreement terms.

/wsag:template/Name

This is an OPTIONAL name that can be given to an agreement matching this template.
Question: what is the name of an agreement template
in relationship to the name of the created agreement
(matching that template)?

What is the nature of the relationship between the two:
1) none, the template name is just an identifier for the template (versus others in the factory),
2) template name is given as a name to the created agreement. If so, then the template name could and maybe be part of the template context (and the resulting agreement name would be part of the agreement context). In that case the agreement should not expect to be identified by its name (which would just be the name of the origin template).

/wsag:template/AgreementContext

This is a REQUIRED element in the agreement template (does it need to be required in the template?). This is the template for the context of the agreements matching the containing agreement template.

/wsag:template/wsag:Terms

This section specifies the possible terms in the agreements matching this template. The description of this section has been made previously in this document (see “Agreement Structure”) and is not repeated here.

/wsag:template/wsag:CreationConstraints

These are OPTIONAL elements that MAY provide constraints on the values that the various terms may take in a concrete agreement.

The specification of a creation constraint section in a template does not state a promise that an agreement creation offer fulfilling the constraints will be accepted. Typically, a provider MAY publish an agreement template containing a creation

constraint section, outlining agreements it is generally willing to accept. Whether the provider accepts a given offer might depend on its current resource situation.

5.1 Creation Constraints

The element CreationConstraints is of type wsag:ConstraintSectionType. It has the following form inside the template:

```
<wsag:template>
      …
   <wsag:CreationConstraints> ?
      <wsag:Item>…</wsag:Item> *
      <wsag:Constraint>…</wsag:Constraint> *
   </wsag:CreationConstraints>

</wsag:template>
```

/wsag:template/wsag:CreationConstraints

This optional element of an Agreement, of type wsag:ConstraintSectionType, expresses the constraints for creating/negotiating an agreement. It contains any number of offer items and constraints in any order.

/wsag:template/wsag:CreationConstraints/wsag:Item

This element specifies that a particular field of the agreement must be present with a value in the agreement offer, and which values are possible.

/wsag:template/wsag:CreationConstraints/wsag:Constraint

A constraint, of type wsag:ConstraintType, defines any constraint involving the values of one or more terms.

The wsag:ConstraintSectionType MAY be used by other specifications in order to define constraints that must apply when creating or modifying agreements, for instance in agreement negotiations.

5.1.1 Offer Item

An offer item specifies the requirement for the presence in the agreement offer terms of a field and a value for that field. It contains a label, a pointer to the position of the field in the terms of the offer and a definition of its acceptable values in the form of a restriction of its value space.

```
<wsag:Item Name="xsd:NCName">
    <wsag:Location>
        xsd:anyType
    <wsag:location/>
    <xsd:restriction>
        xsd:simpleRestrictionModel
    <xsd:restriction> ?
</wsag:Item>
```

/wsag:Item

A simple restriction represents a simple value constraint on a term of an offer.

/wsag:Item/@Name

The name is a label of the field that uniquely identifies the field in the offer and can be used to refer to the restriction item in a convenient way.

/wsag:Item/@Location

The location is a structural reference, for instance an XPATH expression, which points to the location in the terms of the Agreement that can be changed and filled in. The value currently set at the location referred to is the default value of the item.

/wsag:Item/restriction

The restriction element, which is a reference to the group xs:simpleRestrictionModel from the XML Schema namespace, is a constraint that restricts the domain beyond the type definition of the particular term syntax of the item, which can be domain-specific. The restriction syntax is taken from the corresponding XML Schema definition of the group. It is the responsibility of the author of the template to make sure that the restriction defined in the Item is a valid restriction of the type of the field that the item location attribute points to.

Should a template specify the valid cardinality of a given domain-specific service description belonging to the same service?

Shouldn't it be possible to specify creation constraints on guarantees? I can foresee ways to do it but a discussion may be needed…

5.1.2 Free–form Constraints

Free-form constraints make it possible to restrict the possible values of the term set of an offer beyond restrictions of individual terms. For example, an offered response time may only be valid for a given range of throughput values of a service. This specification does not define a constraint language but proposes to choose a suitable existing one. Hence, the Constraint is an empty top-level element that must be extended by a specific, suitable constraint language:

```
<wsag:Constraint/>
```

A general purpose constraint language has been proposed as part of the XQuery and XPATH language. The XML rendering of this expression language, XQueryX, contains a suitable constraint language that can be used to phrase constraints involving multiple items.

```
<wsag:XQueryXConstraint>
   <wsag:Expression> … </wsag:Expression>
</wsag:XQueryXConstraint>
```

/wsag:XQueryXContraint

This element, of type XQueryXConstraintType, substitutes the Constraint element to contain XQueryX expressions.

/wsag:XQueryXContraint/wsag:Expression

This element, of type operatorExpr, taken from the XQueryX schema, contains an operator expression according to this syntax. However, the syntax design of XQueryX is very liberal and, hence, expressions can be phrased that are not semantically valid.
In XQueryX expressions, Item names are mapped to variable names.

Any other constraint language MAY be equally or better suited for particular purposes.

6 Compliance of Offers with Templates

In order for an agreement offer to be accepted, it MUST comply with at least one template advertised by the agreement provider to which the offer is submitted. In this section we define the concept of agreement template compliance.

Definition: An agreement template offer is compliant with a template advertised by an agreement provider if and only if each term of service described in the Terms section of the agreement offer complies with the term constraints expressed in the CreationConstraints section of the agreement template.

TODO: describe precisely what “term of service” means (currently an abstract denomination until we check and better our current service/term model)

Issue: Does template name has influence on agreement name (see comment/question about template name in chapter about template structure)?

Issue: does the context in the offer has obligations of values with respect to the context in the templates? In the template that the offer is compliant with respect to constraints?

In addition, certain options of the Context section of the offer can override equivalent characteristics of an agreement that are understood by default or specified in the template with which the offer is compliant:

- TerminationTime: (or should that be a term i.e. submitted and potentially rejected by the factory?)
- Mapping of signaling roles to service provisioning roles:
    - By default the initiator of the agreement creation request is also the consumer of the service that the agreement is bound to, and the agreement provider is the service provider (see Agreement Context section in this document). By default every agreement created will have this role mapping, unless specified otherwise by the template(s) or the agreement offer itself.
    - A template can specify a different mapping in its Context section. Every agreement based on that template will inherit the mapping specified in the template.
    - An agreement offer can specify a different mapping in its Context section. Every agreement based on this offer will inherit the mapping specified in the offer.

If the mapping of an agreement is not the default one it MUST be reflected in the agreement context (as explained by the chapter of this document about the Agreement Context)

Issue: what if the offer complies with several templates? Does the current model create indecision in terms of which template the factory must choose when creating the agreement? Should the offer be able to specify an identifier i.e. the name of the template?

7 Port Types and Operations

In this section we detail the AgreementFactory and Agreement port types.

Per the reuse principles of the WS-Resource Framework [WS-Resource] on which the Web service expression of this specification is based, interface reuse can be achieved by copying and pasting operation and resource definitions specified here. Designers can reuse the messages and resource properties defined in the AgreementFactory and Agreement port types and compose them in their own specialized, domain-specific port types. They can also compose agreement state-related resource properties as defined in the AgreementState placeholder port type into their own Agreement port type.

Every port type exposes a wsag:GetResourceProperty operation based on the operation of the same NCName as defined in [WS-ResourceProperties]. This operation enables the port types to expose read-only resource properties. Its definition is identical to the one in [WS-ResourceProperties] and has not been repeated here.

The wsrp:GetMultipleResourceProperties operation from [WS-ResourceProperties] MAY be composed as well in order to enable retrieval of several resource properties in one request/response message exchange, for instance in order to obtain a complete agreement in one round-trip invocation. Similarly, other operations from [WS-ResourceProperties] (and other specifications) such as wsrp:QueryResourceProperty MAY be composed into domain-specific agreement and agreement factory port types.

Full WSDL definition of the port types can be found in Appendix.

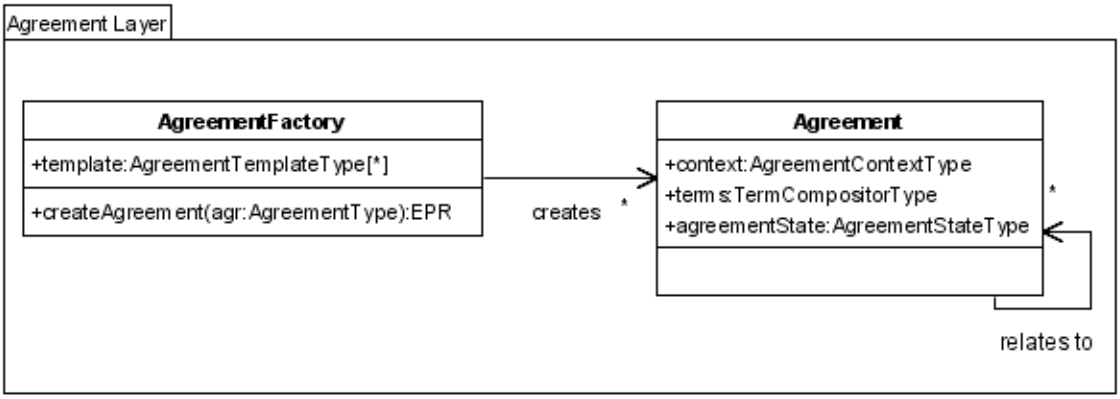


Figure 11: WS-Agreement Services. The '+' sign indicates resource properties that are read-only accessible.

7.1 Port Type wsag:AgreementFactory

7.1.1 Operation wsag:CreateAgreement

The wsag:createAgreement operation is used to generate an Agreement.

7.1.1.1 Input

The form of the wsag:createAgreement input message is:

```
<wsag:createAgreementInput>
    <initiatorAgreementEPR>
        EPR1
    </initiatorAgreementEPR> ?
    <offer>
        ...
    </AgreementOffer>
</wsag:createAgreementInput>
```

The contents of the input message are further described as follows:

/wsag:createAgreementInput/initiatorAgreementEPR

This optional element is an endpoint reference (EPR) providing a contact point EPR1 where the invoked party can send messages pertaining to this Agreement. The invoked party MUST NOT invoke operations on EPR1 after returning a fault on this operation.

/wsag:createAgreementInput/AgreementOffer

The agreement offer made by the sending party. It MUST satisfy the agreement creation constraints expressed in one or more of the templates advertised by the AgreementFactory.

7.1.1.2 Result

The successful result of wsag:createAgreement is a combination of the optional EPR of a newly created Agreement and the acceptance of the initiator's offer:

```
<wsag:createAgreementResponse>
    <createdAgreementEPR>
        EPR2
    </createdAgreementEPR>
</wsag:createAgreementResponse>
```

The contents of the response message are further described as follows:

/wsag:createAgreementResponse/createdAgreementEPR

This is the EPR to a newly created Agreement bearing the same terms as the input agreement offer. This element MUST appear. Could the resulting agreement be more refined/narrowed than to the offer?

/wsag:createAgreementResponse/agreement

The response offer MUST be textually equivalent to the input offer except that the offer type MUST follow the rules of the protocol state machine.

7.1.1.3 Faults

A fault response indicates that the offer was rejected and may also indicate domain-specific reasons.

7.1.2 Resource Property wsag:Template

The templates resource property represents 0 or more templates of offers that can be accepted by the wsag:AgreementFactory operations in order to create an Agreement. A template defines a grouping of certain agreement terms along with negotiability constraints.

7.1.3 Resource Property wssg:Entry

The wsag:AgreementFactory port type can create new resource-qualified endpoint references to services (with associated resources) of port types wsag:Agreement. It may be desirable to expose in the interface the created Agreements, for instance for monitoring clients to use. The wsag:AgreementFactory port type is therefore modeled as a service group with respect to the [WS-ServiceGroup] specification, and records information about each service-resource pair it creates in a new wssg:Entry resource property instance. The entry typically includes the EPR of the new qualified service and MAY contain optional information (see the WS-ServiceGroup specification for more information) that this specification does not define.

7.1.4 Resource Property wssg:MembershipContentRule

This resource property is defined so as to assert the specific content of the wsag:entry resource property and is mandated by [WS-ServiceGroup].

The set of wssg:MembershipContentRule elements specify the intentional constraints on each member service of the service group (see resource property wsag:entry). Each wsgg:membershipContentRule specifies at least a port type that every member service in the service group must implement.

In the context of the wsag:AgreementFactory, there MUST be one wsgg:MembershipContentRule specifying wsag:Agreement as the member port type.

The form of the set of wssg:MembershipContentRule resource properties is:

```
<wsgg:MembershipContentRule
    MemberInterface="port type"
    ContentElements="qnames"/> *
<wsgg:MembershipContentRule
    MemberInterface="wsag:Agreement
    ContentElements="qnames"/> +
<wsgg:MembershipContentRule
    MemberInterface="port type"
    ContentElements="qnames"/> *
```

See the [WS-ServiceGroup] specification for more information on the wsgg:MembershipContentRuleType.

7.2 Port Type wsag:Agreement

The wsag:Agreement port type does not expose any WS-Agreement-specific operations.

7.2.1 Resource Property wsag:Context

The wsag:context resource property is of type wsag:AgreementContextType. The context is static information about the agreement such as the parties involved in the agreement. See the section in this document about the agreement context.

7.2.2 Resource Property wsag:Terms

This property specifies the terms of the agreement.

Note: In some application cases it might be worthwhile to decorate a specialized Agreement port types with a QueryResourceProperty operation as defined in [WS-ResourceProperties], in order to expose the terms of the agreement in a more granular way.

7.2.3 Resource Property wssg:Entry

A wsag:Agreement can be related to others wsag:Agreement for chaining or composition. (how much do we want on this topic in the spec?). This one-to-many relationship is modeled as a service group (see [WS-ServiceGroup]), and records information about each service-resource pair in a wsag:entry resource property instance. An entry includes the EPR of a related wsag:Agreement and MAY contain optional information that this specification does not define.

7.2.4 Resource Property wssg:MembershipContentRule

This resource property is defined so as to assert the specific content of the wsag:entry resource property (see [WS-ServiceGroup]).

The set of wssg:MembershipContentRule elements specify the intentional constraints on each member service of the service group (see resource property wsag:entry). Each wsgg:membershipContentRule specifies at least a port type that every member service in the service group must implement.

In the context of the wsag:Agreement, there must be one wsgg:MembershipContentRule specifying wsag:Agreement as the member port type.

The form of the set of wssg:MembershipContentRule resource property is:

```
<wsgg:MembershipContentRule
    MemberInterface="port type"
    ContentElements="qnames"/> *
<wsgg:MembershipContentRule
    MemberInterface="wsag:Agreement
    ContentElements="qnames"/> +
<wsgg:MembershipContentRule
    MemberInterface="port type"
    ContentElements="qnames"/> *
```

See the [WS-ServiceGroup] specification for more information on the wsgg:MembershipContentRuleType.

7.3 Port Type wsag:AgreementState

The purpose of this port type is to define a resource document type for monitoring the state of the agreement. This port type is not meant to be used as is but instead, its resource properties MAY be composed into a domain-specific Agreement port type.

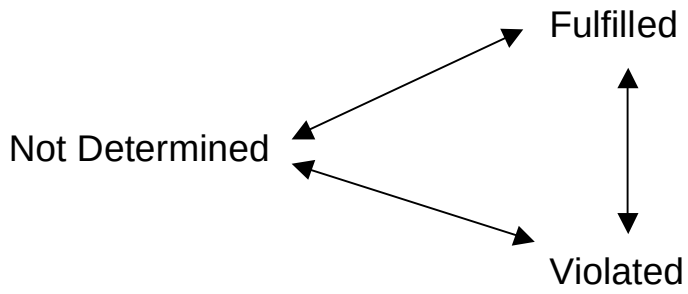
7.3.1 Resource Property wsag:AgreementState

Issue: What is the semantics of the aggregate agreement state? Observed etc.?

7.3.2 Resource Property wsag:GuaranteeTermStateList

This property represents a state of fulfillment for each guarantee term of the agreement. Each list element is a tuple (term ID, guarantee term state).

The guarantee states follow a simple state model:



The semantics of the states is as follows:

- Fulfilled – Currently the guarantee is fulfilled.
- Violated – Currently the guarantee is violated.
- NotDetermined – No activity regarding this guarantee has happened yet or is currently happening that allows evaluating whether the guarantee is met.

7.3.3 Resource Property wsag:ServiceTermStateList

The property exposes a service state for each service description term that abstractly describes the state of a service, independent of its domain. Each list element is a tuple (term ID, service term state).

The service term state observes the following state model:

The semantics of the states is as follows:

- Not Ready – The service cannot be used yet.
- Ready – The service can start now to be used by a client or to be executed by the provider.

- Processing – The service is currently being processed or in use.
- Completed – The service cannot used any more and any provider activity performing a job is finished. This state does not express whether an execution of a job or service was successful.

Based on the service term state, agreement states can be determined. If a service is not ready or ready, the state of a guarantee relating to this service term is not determined. If the service description term is processing or completed, the guarantee term can expose the states fulfilled or violated.

8 Agreement Creation Use Case

Note: since the binding between the agreement layer and the layer of the service being provided is out of the scope of this specification, we omit the steps and operations that expose service layer services or application functionality. Suggestions include using the [WS-ServiceGroup] idiom to have the Agreement service expose the list of services it binds to.

The agreement Factory MAY be a domain-specific specialization of the AgreementFactory described in the port types section of this document. In particular it MAY choose to replicate/reuse the wsag:createAgreement operation.

Process:

1. The initiator is interested in obtaining an agreement for service provisioning with the party implementing the factory. In order to create an agreement in one shot, the initiator calls the createAgreement operation on the Factory service, passing in offer terms that satisfy the creation constraints of one the templates exposed by the Factory as resource properties. If it is not accepted by the Factory, the createAgreement operation will throw a fault message.
2. Assuming the factory accepts the terms, it returns an endpoint reference (EPR) to an observed Agreement service.

9 Acknowledgements

This document is the work of the GRAAP Working Group GRAAP Working Group (Grid Resource Allocation and Agreement Protocol WG) of the Scheduling and Resource Management (SRM) Area of the GGF.

Members of the Working Group are (at the time of writing, and by alphabetical order): Alain Andrieux, (Globus Alliance / USC/ISI), Takuya Araki (ANL), Carl Czajkowski, (Globus Alliance / USC/ISI), Asit Dan (IBM), Kate Keahey (Globus Alliance / ANL), Chris Kurowski (PSNC), Heiko Ludwig (IBM), Jon MacLaren (University of Manchester), Steven Newhouse (London e-Science Centre), Steven Pickles (University of Manchester), Jim Pruyne (HP), John Rofrano (IBM), Volker Sander (*Forschungszentrum Jülich *), Chris Smith (Platform Computing), Steve Tuecke (Globus Alliance / ANL), Alan Weissberger (NEC), Ming XU (Platform Computing), Wolfgang Ziegler (*Fraunhofer–Institute*).

Contributions of the following people are also acknowledged (alphabetically): Ian Foster (ANL), Robert Kearney (IBM), David Kaminsky (IBM), Carl Kesselman

(ANL/USC/ISI), Miron Livny (University of Wisconsin), Jeff Nick (IBM), Ellen Stokes (IBM), John Sweitzer (IBM).

10 References

[SOAP 1.2]
http://www.w3.org/TR/soap12-part1/

[URI]
T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax," RFC 2396, MIT/LCS, U.C. Irvine, Xerox Corporation, August 1998.
http://www.ietf.org/rfc/rfc2396.txt

[WS-Agreement-old]
http://forge.gridforum.org/docman2/ViewProperties.php?group_id=71&document_content_id=358

[SNAP]
K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Tuecke:
"SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems"
http://www.isi.edu/~karlcz/papers/snap-lncs-25370153.pdf

[WS-Addressing]
http://www.ibm.com/developerworks/webservices/library/ws-add/

[WS-Resource]
http://www.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf

[WS-ResourceLifetime]
http://www.ibm.com/developerworks/library/ws-resource/ws-resourcelifetime.pdf

[WS-ResourceProperties]
http://www.ibm.com/developerworks/library/ws-resource/ws-resourceproperties.pdf

[WS-BaseFaults]
http://www.ibm.com/developerworks/library/ws-resource/ws-basefaults.pdf

[WS-ServiceGroup]
http://www.ibm.com/developerworks/library/ws-resource/ws-servicegroup.pdf

[WS-Notification]
http://www.ibm.com/developerworks/library/ws-resource/ws-notification.pdf

[WS-Security]
http://www.ibm.com/developerworks/webservices/library/ws-secure/

[XML-Infoset]
http://www.w3.org/TR/xml-infoset/

[XML]
http://www.w3.org/TR/REC-xml

[XML-ns]

http://www.w3.org/TR/1999/REC-xml-names-19990114

[XPath]
http://www.w3.org/TR/xpath

[ComputeJobs]
A. Andrieux, K. Czajkowski, J. Lam, C. Smith, M. Xu:
"Standard Terms for Specifying Computational Jobs (Proposal to JSDL-WG)"
http://www.epcc.ed.ac.uk/~ali/WORK/GGF/JSDL-WG/DOCS/WS-Agreement_job_terms_for_JSDL_print.pdf

Appendix 1 - WSDL

Factory Port Type WSDL

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-
services/WS-ResourceProperties" xmlns:wsbf="http://www.ibm.com/xmlns/stdwip/web-services/WS-BaseFaults"
targetNamespace="http://www.ggf.org/namespaces/ws-agreement">
    <wsdl:import namespace="http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
location="http://www.ibm.com/developerworks/library/ws-resource/WS-ResourceProperties.wsdl"/>
    <wsdl:types>
        <xs:schema targetNamespace="http://www.ggf.org/namespaces/ws-agreement"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:wssg="http://www.ibm.com/xmlns/stdwip/web-
services/WS-ServiceGroup" xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
elementFormDefault="qualified" attributeFormDefault="qualified">
            <xs:import namespace="http://www.ibm.com/xmlns/stdwip/web-services/WS-ServiceGroup"
schemaLocation="http://www-106.ibm.com/developerworks/library/ws-resource/WS-ServiceGroup.xsd"/>
            <xs:import namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"/>
            <xs:include schemaLocation="agreement_types.xsd"/>
            <!--Resource property element declarations-->
            <!--global elements are defined in the included schema-->
            <!--Resource property document declaration-->
            <xs:element name="AgreementFactoryProperties" type="wsag:AgreementFactoryPropertiesType"/>
            <xs:complexType name="AgreementFactoryPropertiesType">
                <xs:sequence>
                    <xs:element ref="wsag:Template" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="wssg:MembershipContentRule" minOccurs="1" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>
             Contains at least one membershipContentRule1 element such that
             membershipContentRule1/@memberInterface="wsag:Agreement"</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element ref="wssg:Entry" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <!--======================-->
            <!-- Operational input/output type declarations -->
            <xs:element name="createAgreementInput" type="wsag:CreateAgreementInputType"/>
            <xs:element name="createAgreementResponse" type="wsag:CreateAgreementOutputType"/>
            <xs:complexType name="CreateAgreementInputType">
                <xs:sequence>
                    <xs:element name="initiatorAgreementEPR" type="wsa:EndpointReferenceType"
minOccurs="0"/>
                    <xs:element ref="wsag:AgreementOffer"/>
                </xs:sequence>
```

```
            </xs:complexType>
            <xs:complexType name="CreateAgreementOutputType">
                <xs:sequence>
                    <xs:element name="createdAgreementEPR" type="wsa:EndpointReferenceType"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="createAgreementInputMessage">
        <wsdl:part name="parameters" element="wsag:createAgreementInput"/>
    </wsdl:message>
    <wsdl:message name="createAgreementOuputMessage">
        <wsdl:part name="parameters" element="wsag:createAgreementResponse"/>
    </wsdl:message>
    <wsdl:message name="createAgreementFaultMessage">
        <wsdl:part name="fault" element="wsag:ContinuingFault"/>
    </wsdl:message>
    <wsdl:portType name="AgreementFactory" wsrp:ResourceProperties="wsag:AgreementFactoryProperties">
        <wsdl:operation name="createAgreement">
            <wsdl:input message="wsag:createAgreementInputMessage"/>
            <wsdl:output message="wsag:createAgreementOuputMessage"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsrp:ResourceUnknownFault"/>
            <wsdl:fault name="ContinuingFault" message="wsag:createAgreementFaultMessage"/>
            <!-- or message="wsbf:baseFaultMessage " name="baseFault"
        with terminal = false-->
        </wsdl:operation>
        <!-- pasting resource property accessor definitions from WSRP -->
        <wsdl:operation name="GetResourceProperty">
            <wsdl:input name="GetResourcePropertyRequest" message="wsrp:GetResourcePropertyRequest"/>
            <wsdl:output name="GetResourcePropertyResponse"
message="wsrp:GetResourcePropertyResponse"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsrp:ResourceUnknownFault"/>
            <wsdl:fault name="InvalidResourcePropertyQNameFault"
message="wsrp:InvalidResourcePropertyQNameFault"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

Agreement Port Type WSDL

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-
services/WS-ResourceProperties" xmlns:wsbf="http://www.ibm.com/xmlns/stdwip/web-services/WS-BaseFaults"
targetNamespace="http://www.ggf.org/namespaces/ws-agreement">
    <wsdl:import namespace="http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
location="http://www.ibm.com/developerworks/library/ws-resource/WS-ResourceProperties.wsdl"/>
    <wsdl:types>
        <xs:schema targetNamespace="http://www.ggf.org/namespaces/ws-agreement"
xmlns:wssg="http://www.ibm.com/xmlns/stdwip/web-services/WS-ServiceGroup"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing" elementFormDefault="qualified"
attributeFormDefault="qualified">
            <xs:import namespace="http://www.ibm.com/xmlns/stdwip/web-services/WS-ServiceGroup"
schemaLocation="http://www-106.ibm.com/developerworks/library/ws-resource/WS-ServiceGroup.xsd"/>
            <xs:include schemaLocation="agreement_types.xsd"/>
            <!--Resource property element declarations-->
            <!--global elements are defined in the included schema-->
            <!--Resource property document declaration-->
            <xs:element name="agreementProperties" type="wsag:AgreementPropertiesType"/>
            <xs:complexType name="AgreementPropertiesType">
                <xs:sequence>
                    <xs:element ref="wsag:Name" minOccurs="0"/>
                    <xs:element ref="wsag:Context"/>
                    <xs:element ref="wsag:Terms"/>
```

```
                    <xs:element ref="wssg:MembershipContentRule" minOccurs="1" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>
                Contains at least one membershipContentRule1 element such that
                membershipContentRule1/@memberInterface="wsag:Agreement"</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element ref="wssg:Entry" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>
    <wsdl:portType name="Agreement" wsrp:ResourceProperties="wsag:agreementProperties">
        <!-- pasting resource property accessor definitions from WSRP -->
        <wsdl:operation name="GetResourceProperty">
            <wsdl:input name="GetResourcePropertyRequest" message="wsrp:GetResourcePropertyRequest"/>
            <wsdl:output name="GetResourcePropertyResponse"
message="wsrp:GetResourcePropertyResponse"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsrp:ResourceUnknownFault"/>
            <wsdl:fault name="InvalidResourcePropertyQNameFault"
message="wsrp:InvalidResourcePropertyQNameFault"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

Agreement Types Schema

```
<xs:schema targetNamespace="http://www.ggf.org/namespaces/ws-agreement"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:wsbf="http://www.ibm.com/xmlns/stdwip/web-
services/WS-BaseFaults" xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="qualified">
    <xs:import namespace="http://www.w3.org/2001/XMLSchema"
schemaLocation="http://www.w3.org/2001/XMLSchema.xsd"/>
    <xs:import namespace="http://schemas.xmlsoap.org/ws/2003/03/addressing"
schemaLocation="http://schemas.xmlsoap.org/ws/2003/03/addressing"/>
    <xs:import namespace="http://www.ibm.com/xmlns/stdwip/web-services/WS-BaseFaults"
schemaLocation="http://www-106.ibm.com/developerworks/library/ws-resource/WS-BaseFaults.xsd"/>
    <xs:element name="Template" type="wsag:AgreementTemplateType"/>
    <xs:element name="AgreementOffer" type="wsag:AgreementType"/>
    <xs:element name="Name" type="xs:NCName"/>
    <xs:element name="Context" type="wsag:AgreementContextType"/>
    <xs:element name="Terms" type="wsag:TermCompositorType"/>
    <xs:complexType name="AgreementContextType">
        <xs:sequence>
            <xs:element name="AgreementInitiator" type="xs:anyType" minOccurs="0"/>
            <xs:element name="AgreementProvider" type="xs:anyType" minOccurs="0"/>
            <xs:element name="AgreementInitiatorIsServiceConsumer" type="xs:boolean" default="true"
minOccurs="0"/>
            <xs:element name="TerminationTime" type="xs:dateTime" minOccurs="0"/>
            <xs:element name="RelatedAgreements" type="wsag:RelatedAgreementListType" minOccurs="0"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
    <xs:complexType name="TermCompositorType">
        <xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element name="ExactlyOne" type="wsag:TermCompositorType"/>
                <xs:element name="OneOrMore" type="wsag:TermCompositorType"/>
                <xs:element name="All" type="wsag:TermCompositorType"/>
                <xs:element ref="wsag:ServiceDescriptionTerm" minOccurs="0"/>
                <xs:element ref="wsag:GuaranteeTerm" minOccurs="0"/>
            </xs:choice>
```

```
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AgreementTemplateType">
        <xs:sequence>
            <xs:element ref="wsag:Name" minOccurs="0"/>
            <xs:element ref="wsag:Context"/>
            <xs:element ref="wsag:Terms"/>
            <xs:element name="CreationConstraints" type="wsag:ConstraintSectionType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AgreementType">
        <xs:sequence>
            <xs:element ref="wsag:Name" minOccurs="0"/>
            <xs:element ref="wsag:Context"/>
            <xs:element ref="wsag:Terms"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="AgreementInitiatorIdentifierType">
        <xs:sequence>
            <xs:element name="Reference" type="xs:anyType"/>
        </xs:sequence>
        <xs:attribute name="isServiceConsumer" type="xs:boolean" use="optional" default="true"/>
    </xs:complexType>
    <xs:complexType name="AgreementProviderIdentifierType">
        <xs:sequence>
            <xs:element name="Reference" type="xs:anyType"/>
        </xs:sequence>
        <xs:attribute name="isServiceProvider" type="xs:boolean" use="optional" default="true"/>
    </xs:complexType>
    <xs:complexType name="RelatedAgreementListType">
        <xs:sequence>
            <xs:element name="RelatedAgreement" type="wsag:RelatedAgreementType"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="RelatedAgreementType">
        <xs:sequence>
            <xs:element name="AgreementEPR" type="wsa:EndpointReferenceType"/>
        </xs:sequence>
        <xs:attribute name="RelationshipType" type="xs:QName" use="optional"/>
    </xs:complexType>
    <xs:complexType name="TermType" abstract="true">
        <xs:attribute name="Name" type="xs:NCName"/>
    </xs:complexType>
    <xs:complexType name="GuaranteeTermType">
        <xs:complexContent>
            <xs:extension base="wsag:TermType">
                <xs:sequence>
                    <xs:element name="Variables" type="wsag:VariableSetType"/>
                    <xs:element ref="wsag:QualifyingCondition" minOccurs="0"/>
                    <xs:element ref="wsag:ServiceLevelObjective"/>
                    <xs:element name="BusinessValueList" type="wsag:BusinessValueListType"/>
                </xs:sequence>
                <xs:attribute name="ServiceScope" type="wsag:ListOfServiceNames" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="GuaranteeTerm" type="wsag:GuaranteeTermType"/>
    <xs:element name="QualifyingCondition" type="xs:anyType"/>
    <xs:element name="ServiceLevelObjective" type="xs:anyType"/>
    <xs:complexType name="BusinessValueListType">
        <xs:sequence>
            <xs:element name="Importance" type="xs:integer" minOccurs="0"/>
            <xs:element name="Penalty" type="wsag:CompensationType" minOccurs="0"/>
            <xs:element name="Reward" type="wsag:CompensationType" minOccurs="0"/>
            <xs:element ref="wsag:BusinessValue" minOccurs="0" maxOccurs="unbounded"/>
```

```
    </xs:sequence>
</xs:complexType>
<xs:element name="BusinessValue" type="xs:anyType"/>
<xs:complexType name="CompensationType">
    <xs:sequence>
        <xs:element name="AssessmentInterval">
            <xs:complexType>
                <xs:sequence>
                    <xs:choice>
                        <xs:element name="TimeInterval" type="xs:duration"/>
                        <xs:element name="Count" type="xs:positiveInteger"/>
                    </xs:choice>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="ValueUnit" type="xs:string" minOccurs="0"/>
        <xs:element name="ValueExpression" type="xs:anyType"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceDescriptionTermType">
    <xs:complexContent>
        <xs:extension base="wsag:TermType">
            <xs:sequence>
                <xs:any namespace="##other" processContents="strict"/>
            </xs:sequence>
            <xs:attribute name="ServiceName" type="xs:NCName" use="optional"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:element name="ServiceDescriptionTerm" type="wsag:ServiceDescriptionTermType"/>
<xs:simpleType name="ListOfServiceNames">
    <xs:list itemType="xs:NCName"/>
</xs:simpleType>
<xs:element name="Location" type="xs:anyType"/>
<xs:complexType name="VariableType">
    <xs:sequence>
        <xs:element ref="wsag:Location"/>
    </xs:sequence>
    <xs:attribute name="Name" type="xs:NCName"/>
    <xs:attribute name="Metric" type="xs:QName"/>
</xs:complexType>
<xs:complexType name="VariableSetType">
    <xs:sequence>
        <xs:element name="Variable" type="wsag:VariableType" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ConstraintSectionType">
    <xs:sequence>
        <xs:element name="Item" type="wsag:OfferItemType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="wsag:Constraint" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="OfferItemType">
    <xs:sequence>
        <xs:element ref="wsag:Location"/>
        <xs:group ref="xs:simpleRestrictionModel" minOccurs="0"/>
        <!--//AA REMOVE COMMENTS -->
    </xs:sequence>
    <xs:attribute name="name" type="xs:string"/>
</xs:complexType>
<xs:element name="Constraint" type="xs:anyType"/>
<!-- ////// fault section -->
<xs:complexType name="TerminalFaultType">
    <xs:complexContent>
        <xs:extension base="wsbf:BaseFaultType"/>
    </xs:complexContent>
```

```
    </xs:complexType>
    <xs:complexType name="ContinuingFaultType">
        <xs:complexContent>
            <xs:extension base="wsbf:BaseFaultType"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="ContinuingFault" type="wsag:ContinuingFaultType"/>
    <xs:element name="TerminalFault" type="wsag:TerminalFaultType"/>
</xs:schema>
```

AgreementState Port Type WSDL

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-
services/WS-ResourceProperties" xmlns:wsbf="http://www.ibm.com/xmlns/stdwip/web-services/WS-BaseFaults"
targetNamespace="http://www.ggf.org/namespaces/ws-agreement">
    <wsdl:import namespace="http://www.ibm.com/xmlns/stdwip/web-services/WS-ResourceProperties"
location="http://www.ibm.com/developerworks/library/ws-resource/WS-ResourceProperties.wsdl"/>
    <wsdl:types>
        <xs:schema targetNamespace="http://www.ggf.org/namespaces/ws-agreement"
xmlns:wssg="http://www.ibm.com/xmlns/stdwip/web-services/WS-ServiceGroup"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing" elementFormDefault="qualified"
attributeFormDefault="qualified">
            <xs:include schemaLocation="agreement_state_types.xsd"/>
            <!--Resource property element declarations-->
            <!--global elements are defined in the included schema-->
            <!--Resource property document declaration-->
            <xs:element name="AgreementStateProperties" type="wsag:AgreementStatePropertiesType"/>
            <xs:complexType name="AgreementStatePropertiesType">
                <xs:sequence>
                    <xs:element ref="wsag:AgreementState"/>
                    <xs:element ref="wsag:GuaranteeTermStateList"/>
                    <xs:element ref="wsag:ServiceTermStateList"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>
    <wsdl:portType name="AgreementState" wsrp:ResourceProperties="wsag:AgreementStateProperties">
        <!-- pasting resource property accessor definitions from WSRP -->
        <wsdl:operation name="GetResourceProperty">
            <wsdl:input name="GetResourcePropertyRequest" message="wsrp:GetResourcePropertyRequest"/>
            <wsdl:output name="GetResourcePropertyResponse"
message="wsrp:GetResourcePropertyResponse"/>
            <wsdl:fault name="ResourceUnknownFault" message="wsrp:ResourceUnknownFault"/>
            <wsdl:fault name="InvalidResourcePropertyQNameFault"
message="wsrp:InvalidResourcePropertyQNameFault"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

Agreement State Types Schema

```
<xs:schema targetNamespace="http://www.ggf.org/namespaces/ws-agreement"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
xmlns:wsbf="http://www.ibm.com/xmlns/stdwip/web-services/WS-BaseFaults"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" elementFormDefault="qualified"
attributeFormDefault="qualified">
    <xs:simpleType name="AgreementStateType">
        <xs:restriction base="xs:QName">
            <xs:enumeration value="wsag:beforeObserved"/>
            <xs:enumeration value="wsag:observed"/>
            <xs:enumeration value="wsag:afterObserved"/>
```

```
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="NamedGuaranteeTermStateType">
        <xs:simpleContent>
            <xs:extension base="wsag:GuaranteeTermStateType">
                <xs:attribute name="guaranteeTermName" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="GuaranteeTermStateType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NotDetermined"/>
            <xs:enumeration value="Fulfilled"/>
            <xs:enumeration value="Violated"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="GuaranteeTermStateListType">
        <xs:sequence>
            <xs:element name="GuaranteeTermState" type="wsag:NamedGuaranteeTermStateType"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NamedServiceTermStateType">
        <xs:simpleContent>
            <xs:extension base="wsag:ServiceTermStateType">
                <xs:attribute name="guaranteeTermName" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:simpleType name="ServiceTermStateType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NotReady"/>
            <xs:enumeration value="Ready"/>
            <xs:enumeration value="Processing"/>
            <xs:enumeration value="Completed"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ServiceTermStateListType">
        <xs:sequence>
            <xs:element name="ServiceTermState" type="wsag:NamedServiceTermStateType" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <!--global elements are defined in the imported type library-->
    <xs:element name="AgreementState" type="wsag:AgreementStateType"/>
    <xs:element name="GuaranteeTermStateList" type="wsag:GuaranteeTermStateListType"/>
    <xs:element name="ServiceTermStateList" type="wsag:ServiceTermStateListType"/>
</xs:schema>
```

Appendix 2 - Example

Domain-specific Service Description Languages Used in these Examples

The service description elements encountered in the following examples are fictitious but their semantics are inspired from [ComputeJobs], in which they are referred to them as “terms” in the domain of computational jobs. The paper use a deprecated grammar for expressing terms of agreement, therefore the XML expression of computational jobs it describes should be ignored. Domain-specific service description languages can now be totally agnostic of WS-Agreement. The schema

below is an example of such a language. The elements it defines are used in the following examples.

```
<xsd:schema targetNamespace="http://www.gridforum.org/namespaces/job"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:job="http://www.gridforum.org/namespaces/job"
elementFormDefault="qualified" attributeFormDefault="qualified">
    <xsd:complexType name="JobType">
        <xsd:sequence>
            <xsd:element ref="job:executable"/>
            <xsd:element ref="job:arguments"/>
            <xsd:element ref="job:posixStandardInput" minOccurs="0"/>
            <xsd:element ref="job:posixStandardOutput" minOccurs="0"/>
            <xsd:element ref="job:fileStageIn" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="job:fileStageOut" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="job:numberOfCPUs" minOccurs="0"/>
            <xsd:element ref="job:endTime" minOccurs="0"/>
            <xsd:element ref="job:realMemorySize" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="job" type="job:JobType"/>
    <xsd:complexType name="TermType">
        <xsd:sequence>
            <xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute processContents="lax"/>
    </xsd:complexType>
    <xsd:element name="executable" type="xsd:anyType"/>
    <xsd:element name="arguments" type="job:ArgumentsType"/>
    <xsd:complexType name="ArgumentsType">
        <xsd:sequence>
            <xsd:element name="argument" type="xsd:string" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FileStageInTermType">
        <xsd:sequence>
            <xsd:element name="remoteSource" type="xsd:anyURI"/>
            <xsd:element name="localDestination" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="FileStageOutTermType">
        <xsd:sequence>
            <xsd:element name="localSource" type="xsd:string"/>
            <xsd:element name="remoteDestination" type="xsd:anyURI"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="fileStageIn" type="job:FileStageInTermType"/>
    <xsd:element name="fileStageOut" type="job:FileStageOutTermType"/>
    <xsd:element name="beginTime" type="xsd:dateTime"/>
    <xsd:element name="endTime" type="xsd:dateTime"/>
    <xsd:element name="realMemorySize" type="xsd:positiveInteger"/>
    <xsd:element name="numberOfCPUs" type="xsd:positiveInteger"/>
    <xsd:element name="posixStandardInput" type="xsd:string"/>
    <xsd:element name="posixStandardOutput" type="xsd:string"/>
    <xsd:element name="posixStandardError" type="xsd:string"/>
</xsd:schema>
```

Template

This example template enumerates the domain-specific service description elements that are allowed by the factory which advertises it. Note that while most service description elements bear no creational constraints, some of them are restricted in terms of value space. There is no constraint in this example that spans several items.

```
<wsag:Template xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:job="http://www.gridforum.org/namespaces/job"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xsi:schemaLocation="http://www.ggf.org/namespaces/ws-agreement
agreement_types.xsd http://www.gridforum.org/namespaces/job job_terms.xsd">
    <wsag:Name>Template1</wsag:Name>
    <wsag:Context/>
    <wsag:Terms/>
    <wsag:CreationConstraints>
        <wsag:Item wsag:name="executableTerm">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:executable</wsag:Location>
            <!-- for each domain-specific service description <job:executable>,
                 constrain the value of that element (i.e. reduce list of possible executables) -->
            <xs:enumeration xs:value="/bin/processData"/>
            <xs:enumeration xs:value="/bin/doStuff"/>
        </wsag:Item>
        <wsag:Item wsag:name="argumentsTerm">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:arguments</wsag:Location>
            <!--<job:arguments> is allowed; no constraint on its value, whichever the executable may be.-->
        </wsag:Item>
        <wsag:Item wsag:name="stdin">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:posixStandardInput</wsag:Location>
            <!--<job:posixStandardInput> is allowed; no constraint on its value-->
        </wsag:Item>
        <wsag:Item wsag:name="stdin">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:fileStageIn</wsag:Location>
            <!--<job:fileStageIn> is allowed; no constraint on its value-->
        </wsag:Item>
        <wsag:Item wsag:name="stdin">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:numberOfCPUs</wsag:Location>
            <!--<job:numberOfCPUs> is allowed; but must not be greater than 64-->
            <xs:maxInclusive xs:value="64"/>
        </wsag:Item>
        <wsag:Item wsag:name="stdin">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:realMemorySize</wsag:Location>
            <!--<job:realMemorySize> is allowed; but must be within a range-->
            <xs:minInclusive xs:value="128"/>
            <xs:maxInclusive xs:value="1024"/>
        </wsag:Item>
        <wsag:Item wsag:name="fullJob">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:job</wsag:Location>
            <!--A complete <job:job> description is also allowed (maybe this is not a good example...)-->
        </wsag:Item>
    </wsag:CreationConstraints>
</wsag:Template>
```

Offer

This is an example of an agreement offer that is compliant with the template above. Note the various structural complexities of the different domain-specific service description elements (job:executable, job:fileStageIn, job:job, etc…).

This example shows alternate branches using logical grouping compositors: the requested number of CPUs to allocate for the job “ComputeJob1” and the requested memory size used per CPU for the same service are packaged together in two flavors. In one of them, the number of CPUs is relatively high while the memory is relatively low and vice-versa for the other flavor.

Concepts for which it makes sense to specify single fixed values are expressed as domain-specific service descriptions inside wsag:ServiceDescriptionTerm elements. For instance, job:executable.

There are guarantees which express the following requests:

- A constraint on the sum of the respective durations of the two file stage-in transfers described within the context of the service “ComputeJob1”. The XPATH expressions in the variables point to the respective service description elements. The total duration must not exceed 50 seconds (the duration refers to the fictitious metric “time:duration” which in this example is assumed to be imported from another namespace and is of type xsd:duration).
- A constraint on the total time by which both services designated as “ComputeJob1” and “ComputeJob2” must be finished. The metric it refers to is job:endTime which is of type xsd:dateTime, thus the format of the time limit in the constraint expression.

The constraint language used within the guarantees is assumed as well. It could have been XML-based.

```
<wsag:AgreementOffer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xmlns:job="http://www.gridforum.org/namespaces/job" xsi:schemaLocation="http://www.ggf.org/namespaces/ws-
agreement
agreement_types.xsd http://www.gridforum.org/namespaces/job job_terms.xsd">
    <wsag:Name>Offer1</wsag:Name>
    <wsag:Context/>
    <wsag:Terms>
        <wsag:All>
            <wsag:ServiceDescriptionTerm wsag:Name="executable" wsag:ServiceName="ComputeJob1">
                <job:executable>/bin/processData</job:executable>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="arguments" wsag:ServiceName="ComputeJob1">
                <job:arguments>
                    <job:argument>-d</job:argument>
                    <job:argument>-c</job:argument>
                    <job:argument>myFile</job:argument>
                </job:arguments>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="fileStageIn1" wsag:ServiceName="ComputeJob1">
                <job:fileStageIn>
                    <job:remoteSource>protocol://submachine:3456/data/file1</job:remoteSource>
                    <job:localDestination>job/input/type1_data</job:localDestination>
                </job:fileStageIn>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="fileStageIn2" wsag:ServiceName="ComputeJob1">
                <job:fileStageIn>
                    <job:remoteSource>protocol://submachine:3456/data/file2</job:remoteSource>
                    <job:localDestination>job/input/type2_data</job:localDestination>
                </job:fileStageIn>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="FullComputeJob2"
wsag:ServiceName="ComputeJob2">
                <job:job>
                    <job:executable>/bin/doStuff</job:executable>
                    <job:arguments>
                        <job:argument>-u</job:argument>
                    </job:arguments>
                    <job:posixStandardInput>job/input/type1_data</job:posixStandardInput>
                </job:job>
```

```
            </wsag:ServiceDescriptionTerm>
            <wsag:ExactlyOne>
                <wsag:All>
                    <wsag:ServiceDescriptionTerm wsag:Name="numberOfCPUs"
wsag:ServiceName="ComputeJob1">
                        <job:numberOfCPUs>32</job:numberOfCPUs>
                    </wsag:ServiceDescriptionTerm>
                    <wsag:ServiceDescriptionTerm wsag:Name="memoryPerCPU"
wsag:ServiceName="ComputeJob1">
                        <job:realMemorySize>200</job:realMemorySize>
                    </wsag:ServiceDescriptionTerm>
                </wsag:All>
                <wsag:All>
                    <wsag:ServiceDescriptionTerm wsag:Name="numberOfCPUs"
wsag:ServiceName="ComputeJob1">
                        <job:numberOfCPUs>8</job:numberOfCPUs>
                    </wsag:ServiceDescriptionTerm>
                    <wsag:ServiceDescriptionTerm wsag:Name="memoryPerCPU"
wsag:ServiceName="ComputeJob1">
                        <job:realMemorySize>1000</job:realMemorySize>
                    </wsag:ServiceDescriptionTerm>
                </wsag:All>
            </wsag:ExactlyOne>
            <wsag:GuaranteeTerm wsag:Name="MaxTransferDurationForJob1"
wsag:ServiceScope="ComputeJob1">
                <wsag:Variables>
                    <wsag:Variable wsag:Name="duration1" wsag:Metric="time:duration">

    <wsag:Location>/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name='fil
eStageIn1']</wsag:Location>
                    </wsag:Variable>
                    <wsag:Variable wsag:Name="duration2" wsag:Metric="time:duration">

    <wsag:Location>/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name='fil
eStageIn2']</wsag:Location>
                    </wsag:Variable>
                </wsag:Variables>
                <wsag:ServiceLevelObjective>(duration1 + duration2) IS_LESS_INCLUSIVE
50S</wsag:ServiceLevelObjective>
                <wsag:BusinessValueList>
                    <wsag:Penalty>
                        <wsag:AssessmentInterval>
                            <wsag:Count>1</wsag:Count>
                        </wsag:AssessmentInterval>
                        <wsag:ValueExpression>2</wsag:ValueExpression>
                    </wsag:Penalty>
                </wsag:BusinessValueList>
            </wsag:GuaranteeTerm>
            <wsag:GuaranteeTerm wsag:Name="MaxEndTime" wsag:ServiceScope="ComputeJob1
ComputeJob2">
                <wsag:Variables>
                    <wsag:Variable wsag:Name="endTime" wsag:Metric="job:endTime">
                        <wsag:Location>/wsag:AgreementOffer/wsag:Terms/wsag:All</wsag:Location>
                    </wsag:Variable>
                </wsag:Variables>
                <wsag:ServiceLevelObjective>endTime IS_BEFORE 2004-05-
16T00:00:00</wsag:ServiceLevelObjective>
                <wsag:BusinessValueList>
                    <wsag:Penalty>
                        <wsag:AssessmentInterval>
                            <wsag:Count>1</wsag:Count>
                        </wsag:AssessmentInterval>
                        <wsag:ValueExpression>5</wsag:ValueExpression>
                    </wsag:Penalty>
                </wsag:BusinessValueList>
            </wsag:GuaranteeTerm>
```

```
        </wsag:All>
    </wsag:Terms>
</wsag:AgreementOffer>
```

Agreement

This is an example of an agreement after acceptance of the offer. Notice that in this example, the only difference with the offer is that the alternate branches have been reduced to only one, corresponding to the choice made by the factory (based on resource availability). The service provider could have inserted qualifying conditions on certain terms of service, depending on factors such as resource availability.

This agreement document is the response of a GetResourceProperty request with the QName of the wsag:Terms resource property as the input parameter.

```
<wsrp:GetResourcePropertyResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:wsrp="http://www.ibm.com/xmlns/stdwip/web-
services/WS-ResourceProperties" xmlns:job="http://www.gridforum.org/namespaces/job">
    <wsag:Terms>
        <wsag:All>
            <wsag:ServiceDescriptionTerm wsag:Name="executable" wsag:ServiceName="ComputeJob1">
                <job:executable>/bin/processData</job:executable>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="arguments" wsag:ServiceName="ComputeJob1">
                <job:arguments>
                    <job:argument>-d</job:argument>
                    <job:argument>-c</job:argument>
                    <job:argument>myFile</job:argument>
                </job:arguments>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="fileStageIn1" wsag:ServiceName="ComputeJob1">
                <job:fileStageIn>
                    <job:remoteSource>protocol://submachine:3456/data/file1</job:remoteSource>
                    <job:localDestination>job/input/type1_data</job:localDestination>
                </job:fileStageIn>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="fileStageIn2" wsag:ServiceName="ComputeJob1">
                <job:fileStageIn>
                    <job:remoteSource>protocol://submachine:3456/data/file2</job:remoteSource>
                    <job:localDestination>job/input/type2_data</job:localDestination>
                </job:fileStageIn>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm wsag:Name="FullComputeJob2"
wsag:ServiceName="ComputeJob2">
                <job:job>
                    <job:executable>/bin/doStuff</job:executable>
                    <job:arguments>
                        <job:argument>-u</job:argument>
                    </job:arguments>
                    <job:posixStandardInput>job/input/type1_data</job:posixStandardInput>
                </job:job>
            </wsag:ServiceDescriptionTerm>
            <wsag:All>
                <wsag:ServiceDescriptionTerm wsag:Name="numberOfCPUs"
wsag:ServiceName="ComputeJob1">
                    <job:numberOfCPUs>32</job:numberOfCPUs>
                </wsag:ServiceDescriptionTerm>
                <wsag:ServiceDescriptionTerm wsag:Name="memoryPerCPU"
wsag:ServiceName="ComputeJob1">
                    <job:realMemorySize>200</job:realMemorySize>
                </wsag:ServiceDescriptionTerm>
```

```
            </wsag:All>
            <wsag:GuaranteeTerm wsag:Name="MaxTransferDurationForJob1"
wsag:ServiceScope="ComputeJob1">
                <wsag:Variables>
                    <wsag:Variable wsag:Name="duration1" wsag:Metric="time:duration">

    <wsag:Location>/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name='fil
eStageIn1']</wsag:Location>
                    </wsag:Variable>
                    <wsag:Variable wsag:Name="duration2" wsag:Metric="time:duration">

    <wsag:Location>/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name='fil
eStageIn2']</wsag:Location>
                    </wsag:Variable>
                </wsag:Variables>
                <wsag:ServiceLevelObjective>(duration1 + duration2) IS_LESS_INCLUSIVE
50S</wsag:ServiceLevelObjective>
                <wsag:BusinessValueList>
                    <wsag:Penalty>
                        <wsag:AssessmentInterval>
                            <wsag:Count>1</wsag:Count>
                        </wsag:AssessmentInterval>
                        <wsag:ValueExpression>2</wsag:ValueExpression>
                    </wsag:Penalty>
                </wsag:BusinessValueList>
            </wsag:GuaranteeTerm>
            <wsag:GuaranteeTerm wsag:Name="MaxEndTime" wsag:ServiceScope="ComputeJob1
ComputeJob2">
                <wsag:Variables>
                    <wsag:Variable wsag:Name="endTime" wsag:Metric="job:endTime">
                        <wsag:Location>/wsag:AgreementOffer/wsag:Terms/wsag:All</wsag:Location>
                    </wsag:Variable>
                </wsag:Variables>
                <wsag:ServiceLevelObjective>endTime IS_BEFORE 2004-05-
16T00:00:00</wsag:ServiceLevelObjective>
                <wsag:BusinessValueList>
                    <wsag:Penalty>
                        <wsag:AssessmentInterval>
                            <wsag:Count>1</wsag:Count>
                        </wsag:AssessmentInterval>
                        <wsag:ValueExpression>5</wsag:ValueExpression>
                    </wsag:Penalty>
                </wsag:BusinessValueList>
            </wsag:GuaranteeTerm>
        </wsag:All>
    </wsag:Terms>
</wsrp:GetResourcePropertyResponse>
```