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

June 29, 2005

Web Services Agreement Specification (WS-Agreement)

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.

Abstract

This document describes Web Services Agreement Specification (WS-Agreement), a Web Services protocol for establishing agreement between two parties, such as between a service provider and consumer, using an extensible XML language for specifying the nature of the agreement, and agreement templates to facilitate discovery of compatible agreement parties. The specification consists of three parts which may be used in a composable manner: a schema for specifying an agreement, a schema for specifying an agreement template, and a set of port types and operations for managing agreement life-cycle, including creation, expiration, and monitoring of agreement states.

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 obtain state-dependent guarantees from the service provider, represented as an agreement on the service and the associated guarantees. Additionally, the guarantees on service quality should be monitored and service consumers may be notified of failure to meet these guarantees. The objective of the WS-Agreement specification is to define a language and a protocol for advertising the capabilities of service providers and creating agreements based on creational offers, and for monitoring agreement compliance at runtime.

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 service provider on the availability of resources and/or on service qualities. For example, an agreement may provide assurances on the bounds of 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, zero or more discipline-specific service terms, and zero 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 service consumer side or by the service 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. Section 4 provides 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 compliance and section 7 introduces the port types and operations in the specification. Section 10 describes the process leading to the creation of an agreement.

1.1 Goals and Requirements

The goal of WS-Agreement is to standardize the terminology, concepts, overall agreement structure with types of agreement terms, agreement template with creation constraints and a set of port types and operations for creation, expiration and monitoring 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 term: It must be possible to create agreements for services defined by any domain specific service 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 service 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 which compose with schemas defined by WS-Agreement.
- Must Standalone: simple agreement creation must be supported in the WS-Agreement specification, independent of any negotiation model.

Relationship to other WS-* specifications: WS-Agreement is dependent on WS-Addressing and WS-ResourceProperties. WS-Agreement is also meant to be composable with other Web services specifications.

External Specification	Spec Version Standards Body	Status	Is used for
WS-ResourceProperties (WS-RF RP)	Being Discussed in OASIS: Web Services Resource Framework (WSRF) TC Group: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf	Evolving Institutional	Resource properties on port types
WS-Addressing	Being Discussed in Web Services Addressing Working Group http://www.w3.org/2002/ws/addr/	Evolving Institutional	End point references to resource-qualified services

Evolving Institutional Standard: – A specification that is evolving toward an Institutional Standard. An active community within a recognized standards development organization is working on the specification. A specification of this status type must have an externally stable reference to the specification.

(cf. "OGSA profile definition 1.0."

https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-profile-definition/en/10)

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 languages may be used in different usage domains.
- 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 protocol for negotiating agreements.

1.2 Notational Conventions and Terminology

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/>).

Furthermore, this specification defines and uses the following terms:

Agreement. An agreement defines a dynamically-established and dynamically-managed relationship between parties. The object of this relationship is the delivery of a service by one of the parties within the context of the agreement. The management of this delivery is achieved by agreeing on the respective roles, rights and obligations of the parties. The agreement may specify not only functional properties for identification or creation of the service, but also non-functional properties of the service such as performance or availability. Entities can dynamically establish and manage agreements via Web service interfaces.

Business value. The business value is intended to represent the strength of an agreement in domain-specific terms. In general, business value is an assertion representing a value aspect of a service level objective attached to the service that is the subject of the agreement. The value may be specified in terms of domain-specific qualities such as importance, cost and others. Each service level objective may have a list of business values attached to it, representing different value aspects of this objective. Both agreement consumer and agreement provider may specify business values.

Consumer (Service Consumer). A service consumer is an entity entering into an agreement with the intent of obtaining guarantees on the availability of certain services from the service provider. The agreement is negotiated by an agreement initiator on behalf of the service consumer. A service consumer and agreement initiator may, but need not, represent the same entity.

Constraints (Agreement Creation Constraints). Agreement creation constraints define a set of possible values for the agreement terms. They are represented in a separate and optional element of the agreement template and refer back to individual terms they apply to using XPATH. Agreement constraints do not represent a promise on the part of the agreement provider that an agreement creation request will be accepted; they lay down rules which must be followed in the creation of an agreement but the acceptance of the individual term values is dependent on the state of the provider.

Context (Agreement Context). Agreement context represents the immutable part of the agreement (?). It contains information about agreement parties, the agreement's lifetime, and (optionally) a pointer to the template from which the agreement is created.

Expiration (Time). Expiration time defines a time when an agreement is no longer valid, and the parties are no-longer obligated by the terms of the agreement.

Guarantee (Guarantee Terms). Guarantee terms define the assurance on service quality (or availability) associated with the service described by the service definition terms. They refer to the service description that is the subject of the agreement and define service level objectives (describing for example the quality of service on execution that needs to be met), qualifying conditions (defining for example when those objectives have to be met) and business value expressing the importance of the service level objectives.

Initiator (agreement initiator). An agreement initiator is a party to an agreement. The initiator creates and manages an agreement on the availability of a service on behalf of the service consumer. Agreement initiator and service consumer may, but need not, represent the same entity.

Negotiation. The negotiation is an iterative message exchange intended to produce an agreement. The process of negotiation is initiated by retrieving the agreement template and ends in commitment.

Parties (Agreement Parties). Agreement parties consist of the agreement initiator and agreement provider.

Provider (Service Provider). A service provider is an entity entering into an agreement with the intent of providing a service according to conditions described by the agreement.

Provider (Agreement Provider). The agreement provider is a party to an agreement. It creates and manages an agreement on behalf of the service provider.

Service Description Terms. Service Description Terms describe the functionality that will be delivered under the agreement. The agreement description may include also other non-functional items referring to the service description terms.

Service level objective (SLO). Service Level Objective represents the quality of service aspect of the agreement. Syntactically, it is an assertion over the terms of the agreement as well as such qualities as date and time.

Template (Agreement Template). An agreement template is an XML document by the means of which the agreement factory advertises the types of offers it is willing to accept. Like an agreement document, the template is composed of a template name, a context element, and agreement terms, but additionally also includes information on agreement creation constraints.

Termination (Time). Termination time defines the time when a resource or service representing an agreement will be terminated. Termination type SHOULD be greater than or equal to Expiration time so that the service can be accessed during all times that parties are obliged by the agreement.

Terms (Agreement Terms). Agreement terms define the content of an agreement. It is expected that most terms will be domain-specific defining qualities such as for example service description, termination clauses, transferability options and others.

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/

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.

Note: in the examples we will assume that the service provider acts as the agreement provider, and the service consumer as the agreement initiator.

2.1 Job submission

A typical application scenario is the request for executing a computing job. A service provider may, as an agreement provider, post an agreement template available to interested requesters. In this scenario, the agreement template defines a resource preference and commitment profile that can be used to establish an agreement with the resource provider for multiple subsequent job submissions. The template may include limits on available resources, or in some cases, fixed allocation of resource types and quantity. A resource preference and commitment profile may include a quality of service guarantee in terms of number of nodes and/or per node memory and storage for a specific time period. It may also include an expression of preferences over resource amounts, such as a node with twice the memory size is preferred three times more over a basic node. Alternatively, the guarantees can be on the completion time.

All subsequent job submission under this resource agreement, (that further specify the name of an executable, input and output files, and additional dependency on software environment,) will be used by the resource provider to allocate resources for execution of these jobs. Resource preference and commitment agreements associated with multiple waiting jobs are used in matching available resources and improving overall business value of the provider.

2.2 Service Parameterization

In this scenario, the service contracted is an application service provided by a financial company. The service consists of online banking and investment, where online banking service operations are accessible via a web browser and investment operations as web services. Online banking operations, such as UpdateUserProfile, GetAcctBalance, SchedulePayment, GetTransactionHistory and GetPaymentHistory are exposed via a portal. Investment operations, such as StockQuote, BuyShares and SellShares are exposed as web services using the Web Service Description Language (WSDL).

The financial company offers several service levels, such as Gold, Silver, Bronze etc, where each service level requires minimum investment amount and/or account balance, as well as offers different banking and investment fee structure. Additionally, each service level provides a certain Quality of Service (QoS), described via an agreement template, specifying the service and its guarantees, including the QoS options available to the customer. When new customers open accounts with the financial institution, they select a service level by customizing the options specified in the template. Customers can add availability and response time guarantees to individual operations of the interface. For availability, customers may 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 (especially for web service operations) for which the response time goal must hold. Also, customers can set the time when the service will be available such as 8AM to midnight daily.

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. If not, the offer is rejected, and the customer may create a new offer with different desired service levels.

3 Layered Model

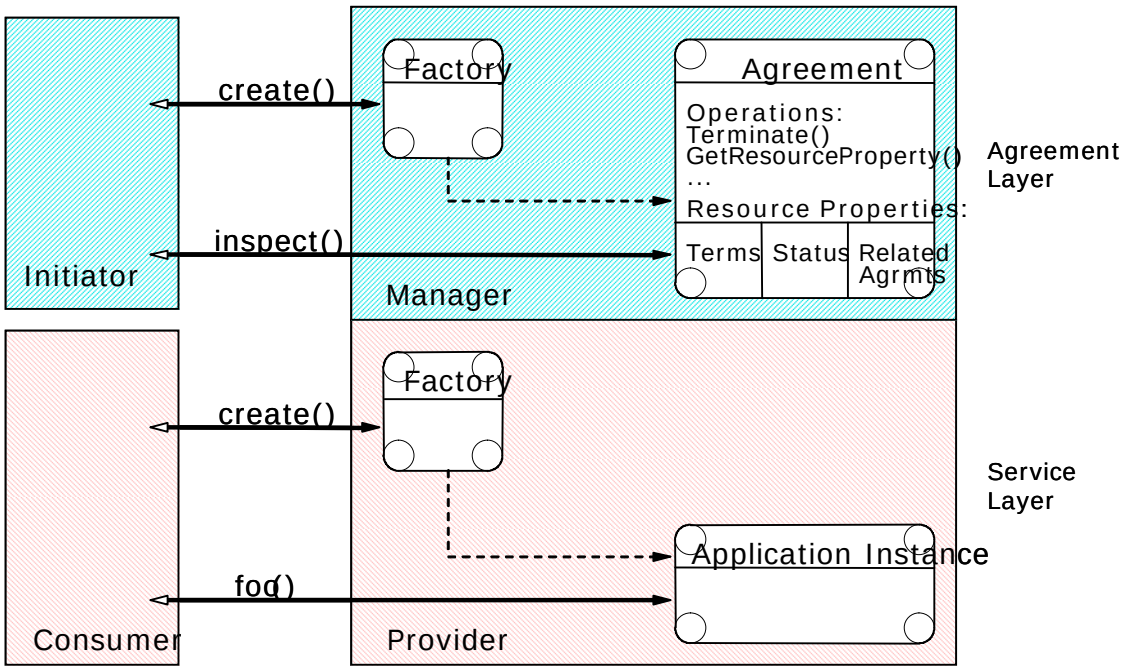


Figure 1: WS-Agreement Conceptual Layered Service Model.
Note: The names of the different operations and "attributes" are not normative.

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 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 later in this specification:
   - An agreement factory exposes an operation for creating an agreement out of an initial set of terms. It returns an Endpoint Reference (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 are be defined independently of the domain-specific agreement terms defined at the agreement layer. 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 desired).
     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.

- An agreement port type, without any operation other than getters for runtime state and metadata of the agreement.

2. The service layer represents the application-specific layer of the service being provided. The class of provided service MAY or MAY NOT be a Web service interface. For instance, computational jobs may be virtualized as Web service instances with additional, domain-specific port-types. Other services may not have a service oriented representation. 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 interfaces in this layer are domain-specific, and need not be altered when the agreement layer is introduced.

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.

Once an agreement is defined with the initiator (i.e., a party who acts on behalf of the consumer), the service behavior is managed by the provider as per the terms of the agreements. When different agreements are established on behalf of different consumers for a shared service instance, each service invocation needs to identify the agreement under which the invocation is to be managed. Details of service invocation (or resource usage) is specific to a service domain, and hence, outside the scope of this specification. Alternatively, once a consumer is identified, a provider specific mapping may be used to identify an agreement.

4 Agreement Structure

An agreement is conceptually composed of several distinct parts. We summarize the structure in Figure 2:

Figure 2: Structure of an agreement.

The section after the (optional) name is the context, which contains the meta-data for the entire agreement. It names the participants in the agreement, and the agreement's lifetime. 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 an agreement context and a collection of agreement terms.

/wsag:Agreement/wsag:Name

This is an OPTIONAL name that can be given to an agreement. The name of an agreement is independent from the name(s) of the template(s) it is based on (see wsag:Context/wsag:TemplateName below). The Name element is NOT a unique identifier. It MAY be used to provide a human-understandable name to an agreement beyond the Endpoint Reference of the Agreement Resource that will be created eventually.

/wsag:Agreement/wsag: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, and the duration of the agreement.

/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.

4.1 Agreement Context

An agreement is scoped by its associated context that SHOULD include parties to an agreement. Additionally, the agreement context contains various metadata about the agreement such as the duration of the agreement, and optionally, the template name from which the agreement is created.

```
<wsag:Context xsd:anyAttribute>

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

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

  <wsag:ServiceProvider>wsag:AgreementRoleType</wsag:ServiceProvider> ?

  <wsag:ExpirationTime>xs:DateTime</wsag:ExpirationTime> +
  <wsag:TemplateName>xs:string </wsag:TemplateName> +

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

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

/wsag:Context

This is the outermost tag which encapsulates the entire agreement context

/wsag:Context/wsag: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:Context/wsag:AgreementProvider

This optional element identifies the agreement provider, 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:Context/wsag:ServiceProvider

This optional element identifies the service provider and is either AgreementInitiator or AgreementProvider. The default is ServiceProvider.

/wsag:Context/wsag:ExpirationTime

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 MAY be exposed as wsag:ExpirationTime. One should note that ExpirationTime is included in the agreement context because it refers to the whole of the agreement.

/wsag:Context/wsag:TemplateName

This optional element specifies the name of the template from which this agreement is created. The reference to template is useful both for future modification of the agreement as well as provisioning of the service environment by the service provider. The template name MUST be included in an offer if the offer is based on a template (if no template is published by the agreement provider, this element MUST NOT be present in offers). A provider MAY check for this when doing an offer/template compliance check.

/wsag:Context/{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.

/wsag:Context/@{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 main body of an agreement offer, and the consequent agreement, consists of terms. The terms of an agreement are wrapped by a wsag:Terms term compositor.

4.2.1 Term Types

A term expresses definitorial consensus or obligations of a party. Terms are typed. Each term in an agreement has a type that is a subtype of the abstract wsag:TermType:

```
<wsag:Term Name="xsd:NCName?"/>
```

This specification defines two types of terms: service terms and guarantee terms.

- The service terms provide information needed to instantiate or otherwise identify a service to which this agreement pertains and to which guarantee terms can apply. These are further refined as service description, service reference and service property terms.
- 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.

Additional term types – subtypes of wsag:TermType – MAY be defined for specific domains or types of obligations. To introduce additional term types, the abstract type wsag:TermType as defined in the AgreementTypes XML Schema MUST be extended.

4.2.2 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.

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

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

```
<wsag:Terms>
    <wsag:All>
        <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:ServiceReference>
                wsag:ServiceReferenceType
            </wsag:ServiceReference> |
            <wsag:ServiceProperties>
                wsag:ServicePropertiesType
            </wsag:ServiceProperties> |
            <wsag:GuaranteeTerm>
                wsag:GuaranteeTermType
            </wsag:GuaranteeTerm>
```

```
        } *
    </wsag:All>
</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

One or more ServiceDescriptionTerms, and/or ServiceReferences and/or ServiceProperties specify different aspects of a service. A ServiceDescriptionTerm provides an inline functional full or partial description of a new service, i.e. the information necessary to “instantiate” the service.

/wsag:Terms/wsag:ServiceReference

These terms are OPTIONAL. A service reference contains a domain-specific reference to an existing business service.

/wsag:Terms/wsag:ServiceProperties

These terms are OPTIONAL. Service properties specify domain-specific aspects of a service that can be used to express the non-functional requirements (guarantees) of the service.

/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.

4.2.3 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 description terms define the functionality that will be delivered under an agreement. The service description content itself is dependent on the particular domain. An SDT consists of three parts,

- The name of the SDT.
- 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.

4.2.3.1 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/@wsag: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/@wsag: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.

There are two scenarios for which multiple service terms may be used to specify a single service, i.e., the same ServiceName is associated with multiple service terms. First, an agreement may define a packaged service where multiple ServiceDescriptionTerms may specify different service components. Alternatively, different ServiceDescriptionTerms may describe different facets of a service, e.g., interface using WSDL, and additional service properties and associated metrics using ServiceProperties.

/wsag:ServiceDescriptionTerm/{xsd:any}

This element is a placeholder for a partial or full description of 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.

Example: the description of a computational job to execute.

4.2.4 Service Reference

A Service Reference points to a service, e.g., by providing an Endpoint Reference. Both parties understand the semantics of the service that is referred to or know how to query the service about its properties. The following definition describes the simple generic content of this type:

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

The following describes the elements of the schema above:

/wsag:ServiceReference/@wsag:Name

This is the name given to this set of service properties.

/wsag:ServiceReference/@wsag:ServiceName

This attribute identifies a service across multiple service description terms. The purpose of this attribute has been described previously.

/wsag:ServiceReference/{xsd:any}

This element is a domain-specific representation of a reference to a service.

Examples:

- An EPR in an agreement on the performance of an existing Web service
- Metadata identifying a class of packet headers in an agreement on network Quality of Service).

4.2.5 Service Properties

ServiceProperties are used to define measurable and exposed properties associated with a service, such as response time and throughput. The properties are used in expressing service level objectives. The following definition describes the simple generic content of this type:

```
<wsag:ServiceProperties
   wsag:Name="xs:NCName" wsag:ServiceName="xs:NCName">
   <wsag:VariableSet>wsag:VariableSetType</wsag:VariableSet>
</wsag:ServiceProperties>
```

The following describes the elements of the schema above:

/wsag:ServiceProperties/@wsag:Name

This is the name given to this set of service properties.

/wsag:ServiceProperties/@wsag:ServiceName

This attribute identifies a service across multiple service description terms. The purpose of this attribute has been described previously.

/wsag:ServiceProperties/wsag:VariableSet

This element is a variable set (see definition below).

4.2.5.1 Variable Set

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:Location>xsd:anyType</wsag:Location>
</wsag:Variable>
```

/wsag:Variable/wsag:Location

The value of this element is a structural reference to a field of arbitrary granularity in the service 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 variable set.

/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.

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:Terms/wsag:All/wsag:ServiceDefinitionTerm[@wsag:Name=’fileStageIn
1’]
    </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:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name='fileStageI
n1']
    </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.

4.2.6 Guarantee Terms

One 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 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 banking 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 service provider. Another example of business value is the importance of this assurance to the service consumer and/or to the service provider.

An agreement MAY also require a service consumer to give guarantees if the provider's service depends on it. For example, a service consumer of a compute job agreement may be required to provide a stage-in file in time such that the service provider can timely provide the results. To enable consumer-side guarantees, guarantee terms annotate the party that is obligated.

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

- Obligated: The obligated party.
- 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 set 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.

4.2.6.1 Guarantee Term Structure

A Guarantee Term has the following form:

```
<wsag:GuaranteeTerm Obligated="wsag:ServiceRoleType">
    <wsag:ServiceScope ServiceName="xsd:NCName">
        xsd:any
    </wsag:ServiceScope>*
    <wsag:QualifyingCondition>…</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:Obligated

This attribute defines, which party enters the obligation to the guarantee term. The wsag:partTypes can be either ServiceConsumer or ServiceProvider.

/wsag:GuaranteeTerm/wsag:ServiceScope

A guarantee term can have one or more service scopes. A service scope describes to what service element specifically a service applies. It contains a ServiceName attribute and any other XML structure describing a sub-structure of a service to which the scope applies. For example, a performance guarantee might only apply to one operation of a Web service at a particular end point.

If a guarantee term applies to multiple services, a set of service scopes MUST be defined. There are two scenarios under which a single guarantee term may refer to multiple services. First, a single SLO as an expression may reference properties of multiple services, e.g., to define overall average response time or total MIPs, etc. Alternatively, the same property may be associated with multiple services and hence, the SLO must hold for each service.

/wsag:GuaranteeTerm/wsag:ServiceScope/@ServiceName

The name of a service to which the guarantee term refers to. A guarantee term service scope applies to exactly one service.

/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.

4.2.6.2 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 be 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].

4.2.6.3 Business Value List

Associated with each wsag:ServiceLevelObjective is a wsag:BusinessValueList that contains multiple business values, each expressing a different value aspect of the objective. Depending on the scenario and value type, each value represents an assertion by one or both parties. For example, in an agreement representing resource reservation for job submission, the submitter may express "importance" of meeting an objective, while a provider may specify "confidence" or likelihood of meeting that objective. In an untrusted or cross-organizational scenario, the business value may be expressed as a joint assertion using "penalty" or "reward" value type. A penalty expresses indirectly both the importance to a consumer, where a higher penalty is more likely to induce provider to meet this objective, and also specifies compensation to be assessed for failing to meet the objective.

"Preference" is used to describe a list of fine-granularity business values for different alternatives, where satisfying each alternative results in a different business value. For example, a job submission may specify many resource configuration alternatives, each resulting in a different utility. Depending on the available resources, other competing jobs and the utility to be achieved, the resource provider makes appropriate resource allocation to maximize the overall utility.

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 service provider from fixed allocation of resources. A service 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:Preference> </wsag:Preference>?
  <wsag:CustomBusinessValue> … </wsag:CustomBusinessValue>*
</wsag:BusinessValueList>
```

/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 the penalty to be assessed for not meeting an objective. If multiple Penalty statements are present, they are applied alternatively, depending on the longest assessment interval applicable.

/wsag:BusinessValueList/wsag:Reward

This element (defined below) when present expresses reward to be assessed for meeting an objective. If multiple Reward statements are present, they are applied alternatively, depending on the longest assessment interval applicable.

/wsag:BusinessValueList/wsag:Preference

This element specifies a list of fine-granularity business values for different alternatives, where each alternative refers to a ServiceDescriptionTerm and its associated utility.

/wsag:BusinessValueList/wsag:CustomBusinessValue

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

4.2.6.3.1 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 service 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 thought 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.

4.2.6.3.2 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 service provider. The value expression for reward is similar to that of penalty.

4.2.6.3.3 Preference

"Preference" is used to describe a list of fine-granularity business values for different alternatives, where satisfying each alternative results in a different business value. For example, a job submission may specify many resource configuration alternatives, each resulting in a different utility. Depending on the available resources, other competing jobs and the utility to be achieved, the resource provider makes appropriate resource allocation to maximize the overall utility.

```
<wsag:Preference>
    <wsag:ServiceTermReference>xsd:string </wsag:ServiceTermReference>*
    <wsag:Utility>xsd:float</wsag:Utility>*
</wsag:Preference>
```

/wsag:Preference

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

/wsag:Preference/wsag:ServiceTermReference

This element can appear multiple times each ServiceTermReference references a ServiceTerm representing an alternative for meeting service level objective. Corresponding, utility (specified below) specifies utility in meeting this objective.

/wsag:Preference/wsag:Utility

This element can appear multiple times, one corresponding to each ServiceTermReference.

5 Agreement Template and Creation Constraints

To create an agreement, a client makes an 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 Figure 3:

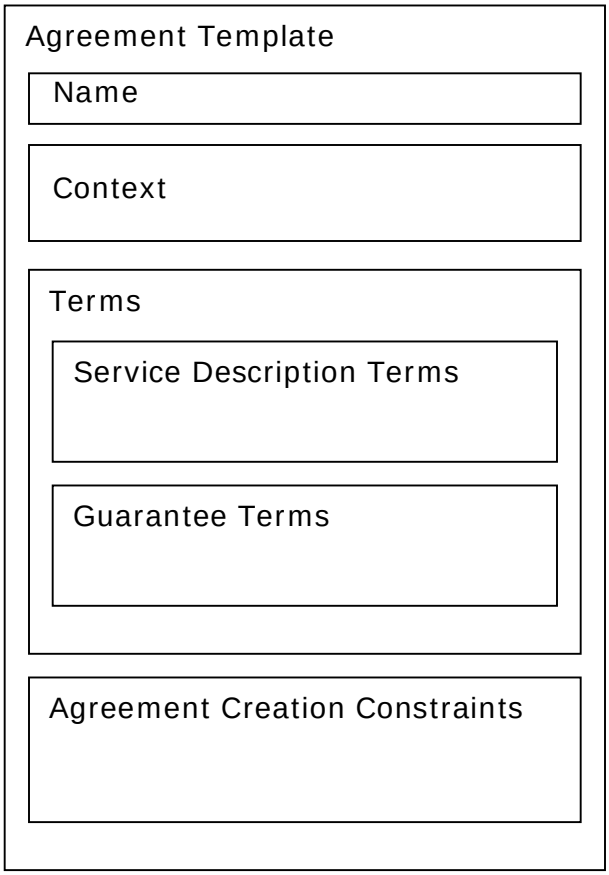


Figure 3: Structure of an agreement template.

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/wsag:Name

This is an OPTIONAL name that can be given to an agreement matching this template.

/wsag:template/wsag:Context

This is a REQUIRED element in the agreement 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, an agreement provider MAY publish an agreement template containing a creation constraint section, outlining agreements it is generally willing to accept. Whether the agreement 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

A restriction applies to the value that can be filled in by an agreement initiator at the specified location at agreement creation time. If all filled in values adhere to their respective restriction an agreement is compliant with its template. 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. Restrictions are not quality of service constraints, which are to be defined in guarantee terms.

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, MAY contain 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. Likewise, an agreement provider MUST NOT accept an agreement offer that does not comply with at least one of the templates it advertises. 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 wsag:CreationConstraints section of the agreement template.

In addition, certain portions of the Context section of the offer have a required relation to corresponding portions of the Context in the template. These are:

- wsag:AgreementProvider:The AgreementProvider value provided in the offer must match the value, if any, specified in the template.
- wsag:ExpirationTime: If the template context contains an ExpirationTime element, the ExpirationTime element of the offer MUST NOT be greater than that of the template.
- wsag:TemplateName: The TemplateName in the offer must exactly match the name provided in the template document against which compliance is being checked. If the TemplateName is not provided, the provider MAY use any policy to determine compliance. These MAY include rejecting all, testing against all templates, or evaluating independently of the templates advertised.

7 Runtime States

Agreements and Terms have a runtime state that can be monitored. The objective of term status monitoring is to observe agreement compliance at runtime. To interpret the state of a guarantee, the service term state must be known. If a service is not running, a guarantee term might not be determined. To interpret the state of a service term, the overall Agreement state must be known. If the Agreement is not accepted, the service and guarantee term states are not determined.

Verifying agreement and – particularly – terms states requires significant infrastructure and is dependent on the application environment and the domain. Hence, the verification of agreement and, term states is outside the scope of this specification.

7.1 Agreement States

The overall Agreement has a state derived from the Agreement protocol.

- Pending. The Pending state means that an Agreement offer has been received but it has been neither accepted nor rejected. This state can only result from use of the wsag:createPendingAgreement operation.
- Observed. The Observed state means that an Agreement offer has been received and accepted. This state MAY follow Pending.
- Rejected. The Rejected state means that an Agreement offer has been received and rejected. This state MAY follow Pending, and can only result from use of the wsag:createPendingAgreement operation.

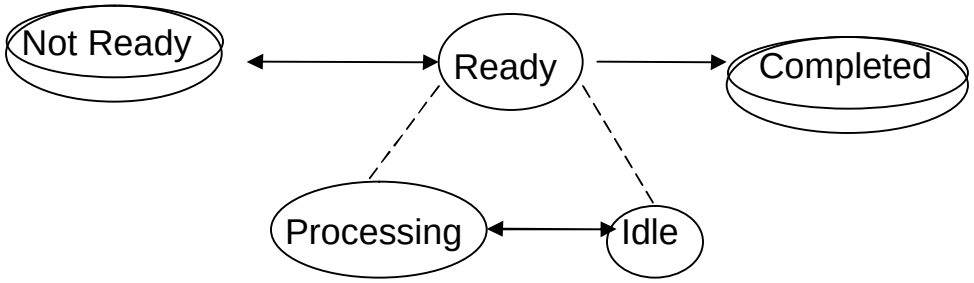
- Complete. The Complete state means that an Agreement offer has been received and accepted, and that all activites pertaining to the Agreement are finished

The Pending and Rejected states indicate that the responder is not obligated in any way. The Pending state indicates that the initiator is obligated if and only if the responder accepts the offer. The Observed and Complete states both indicate that both parties are obligated, and furthermore they are a simple conjunction of the term-level states. The accepted Agreement is Complete only when all service states are Complete, and the accepted Agreement is otherwise Observed.

7.2 Service Runtime States

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:



Not Ready, Ready and Completed are the normative primary states of a service description term. Each state can be extended with one or more sub-states in a specific usage domain. Processing and Idle are two normative sub-states of the primary state Ready.

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 service provider.
- Processing – The service is ready and currently processing a request or is otherwise active.
- Idle – The service is ready, however currently not being used.
- Completed – The service cannot used any more and any service provider activity performing a job is finished. This state does not express whether an execution of a job or service was successful.

Not Ready is the initial state of a service description term while the service is being activated or provisioned. Once a service is ready, it may cycle through the periods of active use and idling, represented by the sub-states of Processing and Idle, respectively. Once a service is completed and can not be reused further, the service description term reaches the terminal state, marked Completed.

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.

```
<wsag:ServiceTermState>
   wsag:ServiceTermStateType
</wsag:ServiceTermState>
```

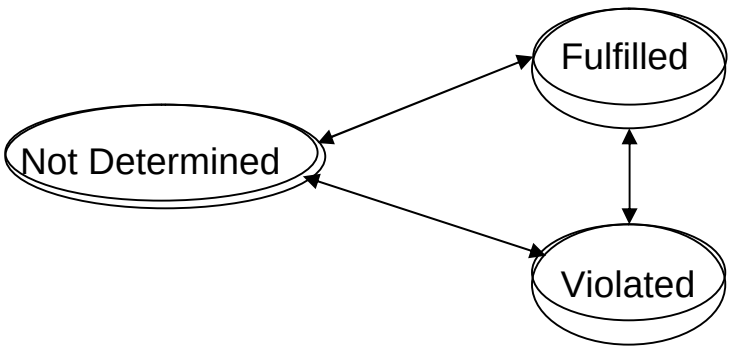
/wsag:ServiceTermState

This element has the type ServiceTermStateType and can hence have the values NotReady, Ready, Processing, or Completed.

7.3 Guarantee States

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.

NotDetermined is the initial state of a guarantee term, until a service is invoked or fulfilled and assessment is made. Depending on the assessment the terminal state can be either Fulfilled or Violated. For guarantee terms, that require recurring assessment, the term state after every assessment period may be in Fulfilled, Violated or Not Determined state. If there was no service activity in the preceding window, then the term state will be in Not Determined state.

```
<wsag:GuaranteeTermState>
   wsag:GuaranteeTermStateType
</wsag:GuaranteeTermState>
```

/wsag:GuaranteeTermState

This element has the type GuaranteeTermStateType and can hence have the values Fulfilled, Violated or NotDetermined.

8 Acceptance Model

The WS-Agreement protocol is based on a single round “offer, accept” message exchange. The basic synchronous wsag:createAgreement operation directly captures this exchange, but additional asynchronous patterns are supported as well. The semantics of this abstract exchange is that the initiator sends an obligating offer, with explicit terms of agreement, which the responder may accept or reject by unilateral decision.

The agreement relationship is in place as soon as the responder decides to accept, and the subsequent return messages inform the initiator of this decision. It is important to understand that the initiator may or may not be obliged to the terms of the agreement until this decision is made; furthermore, in practice the initiator may not be aware of the decision until some time after the decision is made. The obligations expressed in the agreement are not dependent on the initiator being informed of the decision.

8.1 Forms of Offer

The initiator may send his obligating offer in the input message to either the wsag:createAgreement or wsag:createPendingAgreement operations, depending on availability of these operations at the responder’s endpoint. Either form represents the same obligations towards the domain-specific service terms.

8.2 Forms of Acceptance

In response to a wsag:createAgreement offer, the responder accepts the offer by sending the Agreement EPR rather than a fault. The new Agreement MUST already be in an accepted state (Observed or Complete).

In response to a wsag:createPendingAgreement offer, the responder also sends an Agreement EPR but the new Agreement MAY be in any state. The Agreement MUST remain in the Pending state until the responder decides whether to accept, and following a decision to accept, the Agreement MUST transition to an accepted state (Observed or Complete). If the initiator specified an AgreementAcceptance EPR with the offer, the responder also MUST invoke the wsag:Accept operation to indicate its decision.

8.3 Forms of Rejection

In response to a wsag:createAgreement offer, the responder rejects the offer by sending a fault response.

In response to a wsag:createPendingAgreement offer, the responder MAY reject by sending a fault response, and if he returns an Agreement EPR he MAY later indicate rejection by changing the Agreement state from Pending to Rejected. If the initiator specified an AgreementAcceptance EPR with the offer and the responder returns an Agreement EPR, the responder also MUST invoke the wsag:Reject operation to indicate his decision.

8.4 Partial Ordering of Responses

In the case where the responder sends an Agreement EPR and must also invoke an operation on an initiator-supplied AgreementAcceptance service, the creation response and the Accept or Reject input may be sent in either order.

9 Port Types and Operations

In this section we detail the AgreementFactory, PendingAgreementFactory, Agreement and AgreementAcceptance port types. These port types can be used in various combinations to support a number of signaling scenarios:

1. Simple client-server. The AgreementFactory can be invoked by initiators to create a responder-side Agreement and to perform monitoring and management using only WS client mechanisms. Both port types would be implemented by the responder.
2. Simple peer-to-peer. As with the simple client-server scenario, the initiator invokes the AgreementFactory to create a responder-side Agreement. As additional input to the createAgreement call, the initiator passes the address of an initiator-side Agreement representing the offer he is making. As a result, both parties have an Agreement representing the same accepted agreement relationship between them, to allow symmetric messaging for monitoring and management.
3. Deferred (polling) client-server. The PendingAgreementFactory can be invoked by initiators to create a responder-side Agreement on which the responder's decision to accept or reject will later be expressed. The initiator may check the status of the Agreement by any available ResourceProperty query mechanism.
4. Asynchronous client-server. As with the deferred client-server scenario, the initator invokes the PendingAgreementFactory. As additional input to the createPendingAgreement call, the initiator passes the address of an AgreementAcceptance service to which the responder will send an explicit Accept or Reject message to communicate his decision (in addition to updating his Agreement status as in the basic deferred case). This allows for true asynchronous messaging over any binding technology where the initiator can present addressable endpoints.
5. Deferred peer-to-peer. With either basic or asynchronous deferred signaling, the initiator may also pass the address of an initiator-side Agreement to enable fully symmetric monitoring and management in addition to decoupling the responder's acceptance decision from the creation step.

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.

9.1 Port Type wsag:AgreementFactory

9.1.1 Operation wsag:createAgreement

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

9.1.1.1 Input

The form of the wsag:createAgreement input message is:

```
<wsag:createAgreementInput>
    <wsag:initiatorAgreementEPR>
        EPR1
    </wsag:initiatorAgreementEPR> ?
    <wsag:AgreementOffer>
        ...
    </wsag:AgreementOffer>
    <xsd:any>
        ...
    </xsd:any> *
</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, in order to view a symmetric representation of the agreement as viewed by the initiator. The responder MUST NOT invoke operations on EPR1 after returning a fault on this operation. The Agreement addressed by this EPR SHOULD be in the Pending state until this operation returns successfully.

/wsag:createAgreementInput/AgreementOffer

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

/wsag:createAgreementInput/xsd:any##other

These optional elements may be used to carry extensions which control augmented creation mechanisms. By default, all extensions are considered mandatory or critical, i.e. the responder MUST return a fault if any extension is not understood or the responder is unwilling to support the extension.

Extensions MAY be wrapped in <wsag:NoncriticalExtension/> to override this default, in which case a responder MAY ignore the extension and behave as if it were not present. A responder SHOULD obey non-critical extensions if it is able and willing. The meaning of extensions and how to obey them is domain-specific and MUST be understood from the extension content itself.

9.1.1.2 Result

The successful result of wsag:createAgreement is the EPR of a newly created Agreement in the Observed state:

```
<wsag:createAgreementResponse>
    <wsag:createdAgreementEPR>
        EPR2
    </wsag:createdAgreementEPR>
    <xsd:any>
        ...
    </xsd:any> *
</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, and the Agreement MUST be in the Observed or Complete states prior to the return of this response. Such an Agreement MUST NOT transition to the Rejected state.

/wsag:createAgreementResponse/xsd:any##other

These optional items MAY be used to carry extended content that is under the control of corresponding extensions in the input message.

9.1.1.3 Faults

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

9.2 Port Type wsag:PendingAgreementFactory

9.2.1 Operation wsag:createPendingAgreement

The wsag:createPendingAgreement operation is used to generate an Agreement when the decision process may be delayed too long to use (in practice) the wsag:createAgreement operation.

9.2.1.1 Input

The form of the wsag:createPendingAgreement input message is:

```
<wsag:createPendingAgreementInput>
    <wsag:agreementAcceptanceEPR>
        EPR1
    </wsag:agreementAcceptanceEPR> ?
    <wsag:initiatorAgreementEPR>
        EPR2
    </wsag:initiatorAgreementEPR> ?
```

```
    <wsag:agreementOffer>
        ...
    </wsag:agreementOffer>
    <xsd:any>
        ...
    </xsd:any>
</wsag:createPendingAgreementInput>
```

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

/wsag:createPendingAgreementInput/wsag:agreementAcceptanceEPR

This optional element is an endpoint reference (EPR) representing a contact point where the responder MUST invoke a subsequent wsag:Accept or wsag:Reject operation to communicate his decision regarding the offer. This invocation MAY precede or follow the response message to wsag:createPendingAgreement. If this element is omitted, the client SHOULD determine the decision result by other means such as querying the status of the resulting Agreement.

/wsag:createPendingAgreementInput/wsag:initiatorAgreementEPR

This optional element has the same semantics as in wsag:createAgreementInput.

/wsag:createPendingAgreementInput/agreementOffer

This element has the same semantics as in wsag:createAgreementInput.

/wsag:createPendingAgreementInput/xsd:any

These optional elements have the same semantics as in wsag:createAgreementInput.

The two optional EPRs, if specified, MAY address the same service which implements the required features of both the wsag:AgreementAcceptance and wsag:Agreement portTypes, or they MAY address separate services which are correlated by implementation-specific mechanisms. The service(s) addressed by EPR1 and EPR2 MUST implement the required features of wsag:AgreementAcceptance and wsag:Agreement, respectively.

9.2.1.2 Result

The successful result of the wsag:createPendingAgreement operation is the EPR of a newly created Agreement:

```
<wsag:createPendingAgreementResponse>
    <wsag:createdAgreementEPR>
        EPR3
    </wsag:createdAgreementEPR>
    <xsd:any>
        ...
    </xsd:any> *
</wsag:createPendingAgreementResponse>
```

The contents of the response message are understood identically to the response from wsag:createAgreement, except that the resulting Agreement MAY be in the Pending, Observed, Complete, or Rejected states. The responder MAY subsequently reject the offer, resulting in a delayed transition from the Pending to Rejected state.

9.2.1.3 Faults

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

9.2.2 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.

9.3 Port Type wsag:AgreementAcceptance

An AgreementAcceptance resource is associated with an Agreement in the Pending state to allow a deferred decision to be communicated as to whether the Agreement should be Observed or Rejected. The AgreementAcceptance shares this same overall state value with the associated Agreement. One should note that this is a port type on the agreement initiator side; not on the agreement provider side.

9.3.1 Operation wsag:Accept

An AgreementAcceptance resource that is in the Pending state MAY be accepted to transition to the Observed state.

9.3.1.1 Input

The form of the wsag:Accept input message is:

```
<wsag:AcceptInput>
    <xsd:any>
        ...
    </xsd:any> *
</wsag:AcceptInput>
```

The input is usually empty, but the wsag:Accept operation follows the same extensibility pattern as is described in wsag:createAgreement.

9.3.1.2 Result

The successful result of wsag:Accept indicates that the associated Agreement is now understood to be Observed.

```
<wsag:AcceptResponse>
    <xsd:any>
        ...
    </xsd:any> *
</wsag:AcceptResponce>
```

The result is usually empty, but the wsag:Accept operation follows the same extensibility pattern as is described in wsag:createAgreement.

9.3.1.3 Faults

A fault indicates that acceptance of this AgreementAcceptance resource is not possible and also MAY include the reason.

9.3.2 Operation wsag:Reject

An AgreementAcceptance resource that is in the Pending state MAY be rejected to transition to the Rejected state.

9.3.2.1 Input

The form of the wsag:Reject input message is:

```
<wsag:RejectInput>
    <xsd:any>
        ...
    </xsd:any> *
</wsag:RejectInput>
```

The input is usually empty, but the wsag:Reject operation follows the same extensibility pattern as is described in wsag:createAgreement.

9.3.2.2 Result

The successful result of wsag:Reject indicates that the associated Agreement is now understood to be Rejected.

```
<wsag:RejectResponse>
    <xsd:any>
        ...
    </xsd:any> *
</wsag:RejectResponce>
```

The result is usually empty, but the wsag:Reject operation follows the same extensibility pattern as is described in wsag:createAgreement.

9.3.2.3 Faults

A fault indicates that rejection of this AgreementAcceptance resource is not possible and also MAY include the reason.

9.4 Port Type wsag:Agreement

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

9.4.1 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.

9.4.2 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 type with a QueryResourceProperty operation as defined in [WS-ResourceProperties], in order to enable queries on the terms of the agreement in a more fine grain manner.

9.5 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.

9.5.1 Resource Property wsag:ServiceTermStateList

The property exposes a service state for each service description term that abstractly describes the state of a service, as defined in section 7.2.

The set of values are:

- NotReady
- Ready
- Completed

The primary state Ready has associated sub-states with values of Processing and Idle.

The list has the following structure:

```
<wsag:ServiceTermStateList>
    <wsag:NotReady termName="xs:string"> *
      …
    </wsag:NotReady>
    <wsag:Ready termName="xs:string"> *
        <Processing> … </Processing> ?
        <Idle> … </Idle> ?
        …
    </wsag:Ready>
    <wsag:Completed termName="xs:string"> *
      …
    </wsag:Completed>
</wsag:ServiceTermStateList>
```

/wsag:ServiceTermStateList
: List of service states indexed by name. The content in this list can appear in any order of NotReady, Ready and Completed elements, beyond of what the above-used notation can express.

/wsag:ServiceTermStateList/wsag:NotReady
: A service term in state not ready.

/wsag:ServiceTermStateList/wsag:NotReady/@termName
: Name of the term to which it refers.

/wsag:ServiceTermStateList/wsag:Ready
: A service term in state ready.

/wsag:ServiceTermStateList/wsag:Ready/@termName
: Name of the term to which it refers.

/wsag:ServiceTermStateList/wsag:Ready/Processing
: Substate of Processing.

/wsag:ServiceTermStateList/wsag:Ready/Idle

Substate of Idle.

/wsag:ServiceTermStateList/wsag:Completed

A service term in state completed.

/wsag:ServiceTermStateList/wsag:Completed/@termName

Name of the term to which it refers.

9.5.2 Resource Property wsag:GuaranteeTermStateList

This property represents a state of fulfillment for each guarantee term of the agreement, as defined in section 7.3.

The set of values for the top-level states are:

- 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.

The list has the following structure:

```
<wsag:GuaranteeTermStateList>
    <wsag:NotDermined termName=”xs:string”> *
      …
    </wsag:NotDermined>
    <wsag:Fulfilled termName=”xs:string”> *
        …
    </wsag:Fulfilled>
    <wsag:Violated termName=”xs:string”> *
      …
    </wsag:Violated>
</wsag:GuaranteeTermStateList>
```

/wsag:GuaranteeTermStateList

List of guarantee states indexed by name.

/wsag:GuaranteeTermStateList/wsag:NotDetermined

A guarantee term in state not determined.

/wsag:GuaranteeTermStateList/wsag:NotDetermined/@termName

Name of the term to which it refers.

/wsag:GuaranteeTermStateList/wsag:Fulfilled

A guarantee term in state fulfilled.

/wsag:GuaranteeTermStateList/wsag:Fulfilled/@termName

Name of the term to which it refers.

/wsag:GuaranteeTermStateList/wsag:Violated

A guarantee term in state violated.

/wsag:GuaranteeTermStateList/wsag:Violated/@termName

Name of the term to which it refers.

10 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 operation, 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.

11 Acknowledgements

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

Members of the Working Group, and other contributors to this specification include: Takuya Araki (NEC), Dominic Battre (University of Paderborn), Robert Filepp (IBM), Ian Foster (Globus Alliance / ANL), Robert Kearney (IBM), David Kaminsky (IBM), Carl Kesselman (ANL/USC/ISI), Chris Kurowski (PSNC), Jon MacLaren (Louisiana State University, Centre for Computation and Technology), Miron Livny (University of Wisconsin), Steven Newhouse (London e-Science Centre), Jeff Nick (IBM), Steven Pickles (University of Manchester) Volker Sander (*Forschungszentrum Jülich *), Art Sedighi (TIBCO), Chris Smith (Platform Computing), Ellen Stokes (IBM), John Sweitzer (IBM), Alan Weissberger (NEC), Wolfgang Ziegler (*Fraunhofer–Institute*).

12 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

[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

Agreement Types Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.ggf.org/namespaces/ws-agreement" elementFormDefault="qualified"
attributeFormDefault="qualified" 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">
        <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:TermTreeType"/>
        <xs:complexType name="TermTreeType">
                <xs:sequence>
                        <xs:element ref="wsag:All" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <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="ExpirationTime" type="xs:dateTime" minOccurs="0"/>
                        <xs:element name="TemplateName" type="xs:string" minOccurs="0"/>
                        <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:anyAttribute namespace="##other"/>
        </xs:complexType>
        <xs:element name="All" type="wsag:TermCompositorType"/>
        <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 ref="wsag:All"/>
                                <xs:element name="ServiceDescriptionTerm"
type="wsag:ServiceDescriptionTermType"/>
                                <xs:element name="ServiceReference"
type="wsag:ServiceReferenceType"/>
                                <xs:element name="ServiceProperties"
type="wsag:ServicePropertiesType"/>
                                <xs:element name="GuaranteeTerm" type="wsag:GuaranteeTermType"/>
                        </xs:choice>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="AgreementTemplateType">
                <xs:complexContent>
                        <xs:extension base="wsag:AgreementType">
                                <xs:sequence>
                                        <xs:element name="CreationConstraints"
type="wsag:ConstraintSectionType"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </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="TermType" abstract="true">
                <xs:attribute name="Name" type="xs:NCName" use="optional"/>
        </xs:complexType>
        <xs:complexType name="GuaranteeTermType">
                <xs:complexContent>
                        <xs:extension base="wsag:TermType">
                                <xs:sequence>
                                        <xs:element name="ServiceScope"
                                                type="wsag:ServiceSelectorType" minOccurs="0"
                                                maxOccurs="unbounded"/>
                                        <xs:element ref="wsag:QualifyingCondition" minOccurs="0"/>
                                        <xs:element ref="wsag:ServiceLevelObjective"/>
                                        <xs:element name="BusinessValueList"
type="wsag:BusinessValueListType"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="ServiceSelectorType">
                <xs:sequence>
                        <xs:any namespace="##other" processContents="lax"
                                minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="ServiceName" type="xs:NCName" />
        </xs:complexType>

        <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"
                                                        maxOccurs="unbounded"/>
                        <xs:element name="Reward" type="wsag:CompensationType" minOccurs="0"
                                                        maxOccurs="unbounded"/>
                        <xs:element name="Preference" type="wsag:PreferenceType" minOccurs="0"/>
                        <xs:element name="CustomBusinessValue" type="xs:anyType" minOccurs="0"
                                                        maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <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="PreferenceType">
                <xs:sequence minOccurs="1" maxOccurs="unbounded">
                        <xs:element name="ServiceTermReference"     type="xs:string"></xs:element>
                        <xs:element name="Utility" type="xs:float"></xs:element>
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="ServiceTermType" abstract="true">
                <xs:complexContent>
                        <xs:extension base="wsag:TermType">
                                <xs:attribute name="ServiceName" type="xs:NCName" />
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="ServiceReferenceType">
                <xs:complexContent>
                        <xs:extension base="wsag:ServiceTermType">
                                <xs:sequence>
                                        <xs:any namespace="##other" processContents="strict"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="ServiceDescriptionTermType">
                <xs:complexContent>
                        <xs:extension base="wsag:ServiceTermType">
                                <xs:sequence>
                                        <xs:any namespace="##other" processContents="strict"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="ServicePropertiesType">
                <xs:complexContent>
                        <xs:extension base="wsag:ServiceTermType">
                                <xs:sequence>
                                        <xs:element name="VariableSet" type="wsag:VariableSetType"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="ServiceNameSet">
                <xs:sequence>
                        <xs:element name="ServiceName" type="xs:NCName" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <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"/>
        <xs:simpleType name="AgreementRoleType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="AgreementInitiator "/>
                        <xs:enumeration value="AgreementProvider"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="ServiceRoleType">
                <xs:restriction base="xs:string">
                        <xs:enumeration value="ServiceConsumer "/>
                        <xs:enumeration value="ServiceProvider"/>
                </xs:restriction>
        </xs:simpleType>

        <!-- ////// fault section -->
        <xs:complexType name="ContinuingFaultType">
                <xs:complexContent>
                        <xs:extension base="wsbf:BaseFaultType"/>
                </xs:complexContent>
        </xs:complexType>
        <xs:element name="ContinuingFault" type="wsag:ContinuingFaultType"/>
        </xs:schema>
```

Factory Port Type WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<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: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: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

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<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: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:sequence>
                        </xs:complexType>
                        <!--======================-->
                        <!-- Operational input/output type declarations -->
                        <xs:element name="TerminateInput" type="wsag:TerminateInputType"/>
                        <xs:element name="TerminateResponse" type="wsag:TerminateOutputType"/>
                        <xs:complexType name="TerminateInputType">
                                <xs:sequence>
                                        <xs:any processContents="lax" namespace="##any"/>
                                </xs:sequence>
                        </xs:complexType>
                        <xs:complexType name="TerminateOutputType"/>
                </xs:schema>
        </wsdl:types>
        <wsdl:message name="TerminateInputMessage">
                <wsdl:part name="parameters" element="wsag:TerminateInput"/>
        </wsdl:message>
        <wsdl:message name="TerminateOuputMessage">
                <wsdl:part name="parameters" element="wsag:TerminateResponse"/>
        </wsdl:message>
        <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:operation name="Terminate">
                        <wsdl:input name="TerminateRequest" message="wsag:TerminateInputMessage"/>
                        <wsdl:output name="TerminateResponse"
message="wsag:TerminateOuputMessage"/>
                        <wsdl:fault name="ResourceUnknownFault"
message="wsrp:ResourceUnknownFault"/>
                </wsdl:operation>
        </wsdl:portType>
</wsdl:definitions>
```

AgreementState Port Type WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.ggf.org/namespaces/ws-agreement" elementFormDefault="qualified"
attributeFormDefault="qualified" 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">
        <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="TermStateType">
                <xs:choice minOccurs="0”>
                        <xs:any namespace="##other" processContents="lax"  />
                        <!--Processing and Idle only as substates of Ready-->
                        <xs:element name="Processing" type="wsag:InnerTermStateType" />
                        <xs:element name="Idle" type="wsag:InnerTermStateType" />
                </xs:choice>
                <xs:attribute name="termName" type="xs:string"/>
        </xs:complexType>

        <xs:complexType name="InnerTermStateType">
                <xs:sequence>
                        <xs:any namespace="##other" processContents="lax" />
                </xs:sequence>
        </xs:complexType>

        <xs:complexType name="GuaranteeTermStateListType">
                <xs:choice maxOccurs=”unbounded”>
                        <xs:element name="NotDetermined" type="wsag:TermStateType" />
                        <xs:element name="Fulfilled" type="wsag:TermStateType" />
                        <xs:element name="Violated" type="wsag:TermStateType" />
                </xs:choice>
        </xs:complexType>
        <xs:complexType name="ServiceTermStateListType">
                <xs:choice maxOccurs=”unbounded”>
                        <xs:element name="NotReady" type="wsag:TermStateType" />
                        <xs:element name="Ready" type="wsag:TermStateType" />
                        <xs:element name="Completed" type="wsag:TermStateType" />
                </xs:choice>
        </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 - Job Submission 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: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.

```
<?xml version="1.0" encoding="UTF-8"?>
<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="fileStageIn">
            <wsag:Location>//wsag:ServiceDescriptionTerm/job:fileStageIn</wsag:Location>
            <!--<job:fileStageIn> is allowed; no constraint on its value-->
        </wsag:Item>
        <wsag:Item wsag:name="CPUcount">
            <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="memory">
            <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 -->
        </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 time by which the service designated as “ComputeJob1” 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.

```
<?xml version="1.0" encoding="UTF-8"?>
<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:ServiceProperties wsag:ServiceName="ComputeJob1">
                <wsag:VariableSet>
                    <wsag:Variable wsag:Name="duration1" wsag:Metric="time:duration">

    <wsag:Location>/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name='fi
leStageIn1']</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='fi
leStageIn2']</wsag:Location>
                    </wsag:Variable>
                </wsag:VariableSet>
            </wsag:ServiceProperties>
            <wsag:ServiceProperties wsag:ServiceName="ComputeJob1">
                <wsag:VariableSet>
                    <wsag:Variable wsag:Name="endTime" wsag:Metric="job:endTime">
                        <wsag:Location/>
                    </wsag:Variable>
                </wsag:VariableSet>
            </wsag:ServiceProperties>
            <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>
                    <wsag:ServiceName>ComputeJob1</wsag:ServiceName>
                </wsag:ServiceScope>
                <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>
                    <wsag:ServiceName>ComputeJob1</wsag:ServiceName>
                </wsag:ServiceScope>
                <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

Since Agreement Providers MUST NOT change the offer, the returned Agreement is the same as the original offer.

Appendix 3 - Preference Example

Preference business values in guarantee terms can be used to guide which the choice of, for example, system configurations for jobs. The following example illustrates this.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:AgreementOffer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:job="http://www.gridforum.org/namespaces/job" xmlns:sdtc="http://foo.org/sdtc"
xsi:schemaLocation="http://www.ggf.org/namespaces/ws-agreement agreement_types.xsd
http://www.w3.org/2001/XMLSchema XMLSchema.xsd http://www.gridforum.org/namespaces/job job_terms.xsd
http://foo.org/sdtc SDTCondition.xsd ">
    <wsag:Name>Offer2</wsag:Name>
    <wsag:Context/>
    <wsag:Terms>
        <wsag:All>

        <!-- Rest of the job submission example-->

          <wsag:ExactlyOne>
            <wsag:All>
              <wsag:ServiceDescriptionTerm
              wsag:Name="numberOfCPUsHigh"
              wsag:ServiceName="BankingService">
                  <job:numberOfCPUs>32</job:numberOfCPUs>
              </wsag:ServiceDescriptionTerm>
              <wsag:ServiceDescriptionTerm
              wsag:Name="memoryPerCPUHigh"
              wsag:ServiceName="BankingService">
                  <job:realMemorySize>200</job:realMemorySize>
```

```
            </wsag:ServiceDescriptionTerm>
          </wsag:All>
          <wsag:All>
            <wsag:ServiceDescriptionTerm
            wsag:Name="numberOfCPUsLow"
            wsag:ServiceName="ComputeJob1">
                <job:numberOfCPUs>8</job:numberOfCPUs>
            </wsag:ServiceDescriptionTerm>
            <wsag:ServiceDescriptionTerm
            wsag:Name="memoryPerCPULow"
            wsag:ServiceName="ComputeJob1">
                <job:realMemorySize>1000</job:realMemorySize>
            </wsag:ServiceDescriptionTerm>
          </wsag:All>
        </wsag:ExactlyOne>
        <wsag:GuaranteeTerm wsag:Name="ConfigurationPreference">
          <wsag:ServiceScope>
            <wsag:ServiceName>ComputeJob1</wsag:ServiceName>
          </wsag:ServiceScope>
          <wsag:ServiceLevelObjective xsi:type="sdtc:OpType">
            <Or>
              <SDT>numberOfCPUsHigh</SDT>
              <SDT>numberOfCPUsLow</SDT>
            </Or>
          </wsag:ServiceLevelObjective>
          <wsag:BusinessValueList>
            <wsag:Preference>
                <wsag:ServiceTermReference>numberOfCPUsHigh
 </wsag:ServiceTermReference>
                <wsag:Utility>0.8</wsag:Utility>
                <wsag:ServiceTermReference>numberOfCPUsLow
 </wsag:ServiceTermReference>
                <wsag:Utility>0.5</wsag:Utility>
            </wsag:Preference>
          </wsag:BusinessValueList>
        </wsag:GuaranteeTerm>
      </wsag:All>
    </wsag:Terms>
</wsag:AgreementOffer>
```

In this example, the following simple condition language is used:

```
?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://foo.org/sdtc" xmlns:sdtc="http://foo.org/sdtc">
    <complexType name="OpType">
        <sequence>
            <choice minOccurs="1" maxOccurs="2">
                <element name="SDT" type="string"></element>
                <element name="And"
                  type="sdtc:OpType"></element>
                <element name="Or"
                  type="sdtc:OpType"></element>
            </choice>
        </sequence>
    </complexType>
</schema>
```

Appendix 4 - Reference Type Examples

The example shows a service that defines a Web as well as a Web service interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:Agreement xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement" xmlns:sdtc="http://foo.org/sdtc"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ggf.org/namespaces/ws-agreement agreement_types.xsd http://foo.org/sdtc
SDTCondition.xsd http://www.w3.org/2001/XMLSchema XMLSchema.xsd ">

<wsag:Name>Offer3</wsag:Name>
    <wsag:Context/>
    <wsag:Terms>
        <wsag:All>
            <wsag:ServiceDescriptionTerm
             wsag:Name="WSDLInterface"
             wsag:ServiceName="BankingService">
            <sdtc:WSDLReference>
              http://www.foo.org/interfaces/bank.wsdl
            </sdtc:WSDLReference>
             </wsag:ServiceDescriptionTerm>
             <wsag:ServiceDescriptionTerm
             wsag:Name="WebAccess"
             wsag:ServiceName="BankingService">
               <sdtc:URLPrefixDefinition>
```

```
                    http://www.foo.org/bank
                </sdtc:URLPrefixDefinition>
                 </wsag:ServiceDescriptionTerm>
            </wsag:All>

            <!-- More Terms -->

        </wsag:Terms>
</wsag:Agreement>
```

The following – domain-specific – simple schema is used for the references:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://foo.org/sdtc"
xmlns:sdtc="http://foo.org/sdtc">

      <element name="WSDLReference" type="anyURI"></element>
      <element name="URLPrefixDefinition" type="anyURI"></element>

</schema>
```