GFD-E.167
GRAAP-WG

Dominic Battré, TU Berlin (Editor)
Philipp Wieder, TU Dortmund University (Editor)
Wolfgang Ziegler, Fraunhofer SCAI (Editor)

March 8, 2010

WS-Agreement Specification Version 1.0 Experience Document

Status of This Document

This document provides information to the Grid, Distributed Systems and Cloud Computing community about the experience with the WS-Agreement Specification Version 1.0. In describes existing implementations of the standard and compares them. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Trademark

Abstract

This document describes the implementation experiences of independent implementations of WS-Agreement along with an overview of the projects that have implemented WS-Agreement so far. It also presents the features of WS-Agreement used by 8 of the implementations. Finally, the document contains information on set-up and results of an experiment where two independent implementations of WS-Agreement were used to mutually exchange templates describing jobs and create agreements.

Dominic Battré, TU Berlin (Editor)
Philipp Wieder, TU Dortmund University (Editor)
Wolfgang Ziegler, Fraunhofer SCAI (Editor)

March 8, 2010

Contents

1. Introduction

This document describes the implementation experiences of independent implementations of WS-Agreement along with an overview of the projects that have implemented WS-Agreement so far presented in Section 4. Moreover, it also presents the features of WS-Agreement used by 8 implementations where the projects replied to survey organised in 2008. The results of the survey are summarised in Section 5. This section also contains embedded comments related to further issues with the specification identified by the different implementations and reported in the responses to the survey.

Section 6 presents WSAG4J a generic WS-Agreement framework implemented in Java, currently the most complete implementation of the WS-Agreement specification.

Additionally, the document contains information on the set-up and results of an experiment where two independent implementations of WS-Agreement were used to mutually exchange templates describing jobs and create agreements based on these. This description can be found in Section 7.

The document concludes with a discussion of areas, where extensions of the specification will be carried out in the future, namely extending the capabilities of the WS-Agreement negotiation protocol.

2. Notational Conventions

The key words ‘MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC 2119 [BRADNER1].

3. WS-Agreement: Status Quo

3.1 Rationale for Writing this Document

The authors and contributors of the WS-Agreement specification deliberately kept the specification as generic as possible to allow for flexibility in implementations of the language and the protocol for describing and creating service level agreements, thus enabling a broad range of application areas. Therefore, WS-Agreement itself does not specify a domain specific term language but allows users to plug-in their own domain specific term languages to describe the terms of a service. Both properties guarantee that WS-Agreement may be used in arbitrary environments, however, render interoperability difficult.

Since the publication of the proposed recommendation in May 2007, the number of known implementations of WS-Agreement has been growing continuously. Depending on the context of the implementation, the different implementations have different foci and consequently use different features of WS-Agreement. Moreover, these implementations use different hosting environments for the web services, which adds another barrier for interoperation. The GRAAP-WG therefore decided to produce an experience document that covers more than the usual topics. The main parts of this document provide both, a brief presentation of most of the projects that have implemented WS-Agreement, and a description of the WS-Agreement constructs used in these implementations. The latter has been created based on the responses to a questionnaire that has been sent to all projects.

To address the interoperation issues, the document also contains a section describing the interoperation experiment carried out with the implementation of the AssessGrid project and the implementation of the WSAG4J framework in the PHOSPHORUS project. Both projects employ the Job Submission Description Language (JSDL) as term language but use different hosting environments for WS-Agreement.

Finally, to further reduce the complexity of using WS-Agreement and for lowering the threshold for interoperability, the GRAAP-WG has started working on profiles for WS-Agreement that reflect general use-cases. These profiles along with the specific term languages will be published in a separate informational document.

4. Projects Implementing WS-Agreement

This section provides an overview of the projects, which have implemented the WS-Agreement Specification Version 1.0 [GFD.107]. A list of all reported implementations (as 2007) can be found here [SOZ+07] and here [PBM08] or visit the up-to-date list maintained by the GRAAP-WG at https://forge.gridforum.org/sf/wiki/do/viewPage/projects.graap-wg/wiki/Implementations.

As presented below, the implementations spread across the most common Grid middleware stacks: GT4.x, UNICORE and GRIA. Some of the implementations don't rely on Grid middleware, like e.g. the AgentScape implementation.

In most of the projects WS-Agreement is used in the context of Resource Management and Scheduling, use-cases being job submission, resource selection based on requirements of a user, e.g. through an auction, advance reservation and agreement on QoS.

Nine of the projects listed below (AgentScape, Akogrimo, ASKALON, AssessGrid, BEinGRID, BREIN, CATNETS, JSS, VIOLA) use independent implementations without sharing code (except for publicly available WS-* libraries for, e.g. notification or security) while three are using different versions of the WSAG4J framework evolved over the last years (SmartLM, PHOSPHORUS, IANOS). Thus, this experience document describes twelve implementations whereof three are using different versions of the same framework.

In general, the implementations follow the description for the expected behaviour of the service entities (agreement providers and agreement consumers). In all implementations the agreement provider is also the resource provider (or an entity acting on its behalf), while the agreement consumer is the end-user (or an entity acting on its behalf, e.g. a Grid level scheduler or an agent).

Up to now, mostly Java is used for implementing web-service stacks in Grid environments. We are not aware of implementations in other languages. Since we do not have control over other implementations than those we drive ourselves, we can only speculate regarding the reason for this. As far as we are concerned, the reason using Java is the bounding of WS-Agreement to web services and the implicated need for proper tooling which is best provided for Java.

4.1 AgentScape

Name: AgentScape Negotiation Framework

Category: Framework

License type: BSD-like (no attribution)

Link: http://www.agentscape.org/index_html

Description: The Intelligent Interactive Distributed Systems (IIDS) group of the Vrije Universiteit Amsterdam develops the AgentScape framework that

provides mobile agents access to computing resources on heterogeneous systems across the Internet.

AgentScape is an agent platform that provides middleware infrastructure needed to support mobility, security, fault tolerance, distributed resource and service management, and service access, to heterogeneous agent applications. The multi-level AgentScape middleware infrastructure has been designed to be extensible and scalable. Within AgentScape, *agents* are active entities that reside within *locations*, and *services* are third-party software systems accessed by agents hosted by the AgentScape middleware. Agents in AgentScape can communicate with other agents and migrate from one location to another.

AgentScape uses WS-Agreement to manage all negotiations between agents and locations and between agents and external web services. WS-Agreement is used to determine the resources that an agent will be allowed to access when it has migrated to a new location. These leases are negotiated before migration and enforced when the agent arrives at the new location.

Grid ecosystems: The negotiation of resource access for applications is based on WS-Agreements. A mediator called domain coordinator (DC) in AgentScape represents multiple autonomous hosts and communicates with the mobile agent on behalf of these nodes. Agents can negotiate their options with DCs of multiple domains, being able to select the DC that provides the best offer.

WS-Agreement implementation: WS-Agreement is used to negotiate conditions and quality of service of resource access with domain coordinators. Hosts providing resources are aggregated into virtual domains. The DC represents the hosts within a virtual domain in the negotiation process. The WS-Agreement interaction model is extended to allow a more sophisticated negotiation. In this extended negotiation model, hosts provide an agreement interface to their DC. The DC aggregates templates offered by hosts into composed templates and makes these available to agents. The DC receives the agreement requests made by agents based on composed templates. The DC then negotiates an agreement with the hosts with the requested resources. The additional accept/reject interaction sequence allows agents to enter into negotiations with multiple providers and compare received offers. Resources that can be requested and used by agents include CPU time, communication bandwidth, amount of memory, disk space, web services that the agent is allowed to access and the number of calls of a web service that the agent is allowed to do. After the negotiation phase, a host manager monitors and controls the resource usage to ensure that agreements are met.

4.2 Akogrimo

Name: Akogrimo SLA sub-system

Category: A SLA architecture based on Web Services

License type: Dual License (similar to LGPL for academic usage)

Link: http://www.akogrimo.org

Description: More information is available in the “Final Implementation Report of Grid Application Support Service layer” deliverable, available in the Download/Material section of the Akogrimo website[1]. In order to support business applications in a mobile Grid computing environment, the link between the service that presents to the Grid Middleware and the underlying network has to be efficient, in order to support efficient implementation of monitoring, negotiation and service management. Central to this achievement is the SLA Management subsystem at Grid Middleware layer that has to encompass and mirror the SLA subsystem at Network layer including contract definition, SLA negotiation, SLA monitoring and SLA enforcement according to defined policies. In order to join the two SLA layers the main point is to build a new sub layer upon the Grid middleware able to create a negotiation mechanism between providers and consumers of services. In addition, the middleware SLA Enforcement and monitoring subsystems have also the supervisor role in order to verify that the negotiated contract conditions of all running services are met. In order to combine the two layers and in particular to handle service change at the network layer notification is needed.

In Akogrimo the WS-Notification specification is implemented for alerting about abnormal situations so that SLA Management can undertake effective corrective decisions according to defined policies. This tight coupling based around negotiation allows the Grid middleware to become aware of network capabilities aiding efficient cross layer co-operation. A clear example is shown in the management of the Quality of Service. The SLA contract and its negotiation consider QoS parameters that belong to both grid resources (CPU use, Memory, Disk space, etc) and network capabilities (bandwidth, priorities for packet traffic, etc) by means of network bundles or profiles that telecom operators provide. Thus, the application’s QoS requests are mapped on these infrastructure QoS parameters.

This novelty is completed with the close interactions between network and grid at runtime. Thus, any changes on network performance are taken into account by the process that is responsible for the monitoring of QoS parameters and corrective actions and penalties can be applied according to the defined policy in a per-case basis. This management of SLA with respect to QoS illustrates how in the new “Next Generation Grid architectures” SLA is handled as a live adjustable quantity. Here the SLA management is well supported aiding flexibility and adaptability in order to manage externally hosted services toward a combined business goal.

When a Customer asks for an Agreement (step 1) to an “Agreement Provider”, it retrieves information related to the chosen service (interacting with a Discovery Information Service, step 2). This information takes the form of High Level (HL) SLA Template about the service that takes into account some “Human Understandable” QoS values. These values are afterwards translated according to a mapping policy to the respective “low level” (LL) QoS parameters, which are transparent to the final user and are the actual measurable grid and network properties, which are monitored in run-tine (step 3). Then the real negotiation phase starts: the HL SLA Template contains information related to the LL SLA Template that contains the low level

1 http://www.akogrimo.org/modules.php?name=UpDownload&req=viewdownload&cid=5

requirements to negotiate (step 4). Finally, the Application Provider interacting with the Infrastructure Layer, looks for the best suitable host (step 5) that is able to deliver the Application taking into account “well defined” low level QoS parameters. If the negotiation is successfully, the two contracts, HL and LL, are prepared and stored in the Agreement Repository (step 5).

Grid ecosystems: A Customer relies on an Application Provider, which in turn relies on an Infrastructure provider. The User requests a high-level SLA to the Application provider (Gold, Silver or Bronze), which describes the service to be accessible by the user. The Application Provider requests a low-level SLA to the Infrastructure Provider, which defines the resources that are needed for the application execution.

4.3 ASKALON

Name: ASKALON GridARM

Category: Framework

License type: Dual license model (commercial & open source), defined by the ASKALON Project

Link: http://www.dps.uibk.ac.at/projects/askalon/

Description: ASKALON is a Grid project of the Distributed and Parallel Systems Group at the University of Innsbruck. The main goal is to simplify the development and optimisation of applications that can utilise a Grid for computation. ASKALON is used to develop and port scientific applications as workflows in the Austrian Grid project. The developers designed an XML-based Abstract Grid Workflow Language (AGWL) to compose job workflows.

Grid ecosystems: A resource manager remotely deploys software e.g. by using the Globus Toolkit middleware with the GridFTP protocol and the Globus Resource Allocation Manager (GRAM).

WS-Agreement implementation: SLAs can be made with the Grid resource for a specified timeframe by using the GridARM Agreement package. GridARM ensures that a defined capacity and capability is available in the agreed timeframe including parameters like number of CPUs. The agreement management consists of two parts: The AgreementNegotiator and the AgreementService. The AgreementNegotiator works as an agreement factory service. During the agreement negotiation process with the client, multiple agreement offers are created based on the information provided by the client as an AgreementTemplate. The client can accept one or more of the offers or reject all of them. The AgreementService manages particular agreements. After the negotiation process is finished all interaction addressing e.g. agreement access and updates is done by interacting with the AgreementService using an End Point Reference (EPR). In ASKALON, the client (consumer) always creates agreement templates and is therefore always the agreement initiator. The provider creates one or more offers which are accepted or rejected by the client.

4.4 AssessGrid

Name: AssessGrid Negotiation Manager

Category: Generic SLA Framework

License type: Apache Licence 2.0

Link: https://cit-server.cit.tu-berlin.de/trac/negmgr/wiki

Description: AssessGrid is a European project, which started in April 2006 and ended in March 2009. AssessGrid introduces risk management and assessment to Grid computing to facilitate a wider adoption of Grid technologies in business and society. Risk assessment helps providers to make decisions on suitable SLA offers by relating the risk of failure to penalty fees. Similarly, end-users get knowledge about the risk of an SLA violation by a resource provider that helps to make appropriate decisions regarding acceptable costs and penalty fees. A broker is the matchmaker between end-users and providers. The broker provides a time / cost / risk optimised assignment of SLA requests to SLA offers.

Grid ecosystems: The Negotiation Manager provides access to a distributed, planning based RMS called Open Computing Center Software (OpenCCS, see www.openccs.eu). The Negotiation Manager is embedded in a Globus Toolkit 4 (GT4) environment and makes heavy use of GT4 components for WS-SecureConversation (encryption, authorization, and authentication), WS-Notification, credential delegation, and GridFTP for file-staging. The Negotiation Manager consists of a generic component that can be used by new projects and a concrete extension that is suited for planning based RMS. Currently, the OpenCCS scheduler is supported but support for further schedulers can be implemented through a plug-in concept. Jobs are described in the form of JSDL, JSDL-POSIX and JSDL-SPMD.

WS-Agreement implementation: The Negotiation Manager manages the life-cycle of a compute job, comprising negotiation for price, penalty and guarantee probability of failure boundaries, communication with the underlying RMS, performing file staging, monitoring the execution progress, and providing status information in a WS-Agreement compliant way.

The Negotiation Manager provides access to the RMS over WS-Agreement. The major negotiable SLA parameters are: General parameters like number of nodes, amount of memory, job runtime, deadline for job completion, and guaranteed probability of failure boundaries. An optional quote mechanism (invitation to treat) allows requesting price information for not yet agreed SLAs without being binding for either party. This is useful for co-allocation and workflows. Special cancellation policies allow for cheap cancellation of SLAs during the first few minutes of their existence (also useful for the mentioned problems). After receiving the necessary delegated credentials, the negotiation manager performs file staging from a users' GridFTP directory onto a cluster and notifies the RMS that computation may commence. During the computation, the RMS is monitored and in case an SLA cannot be fulfilled, the Negotiation Manager attempts to subcontract another provider for the job execution (by resuming a generated checkpoint or by restarting the job). Besides the Negotiation Manager on the provider layer, another implementation exists in the broker layer. This Broker deals with reputation management and meta-scheduling for workflows. A user can create an SLA with a broker, which may subcontract one or several providers to execute a

single job or an entire workflow. AssessGrid provides a user interface based on Gridsphere to modify parameters of the SLA template, to negotiate, to submit SLA requests, and to monitor them. A second user interface exists in the form of a command line tool, which provides comparable features.

Generic WS-Agreement implementation: The Negotiation Manager consists of two software components. The first component comprises a generic implementation of WS-Agreement that can be used by various projects which decide to use the Globus Toolkit as a hosting environment. The second component comprises a concrete, domain specific implementation beyond the scope of pure WS-Agreement that supports features needed for AssessGrid.

The generic component, the Negotiation Manager framework, contains an Apache Ant based build system that performs stub generation and package compilation, assembly, and deployment. The build system uses inheritance of WSDL Port Types as supported by the Globus Toolkit WSDL pre-processor such that concrete implementations can freely extend the WS-Agreement port-types and resource properties documents by custom features. This has been used for example to add a quote (invitation to treat) mechanism and a mechanism to delegate credentials to a provider. Furthermore, the package system is designed so that several AgreementFactory instances can live within a single Globus Toolkit hosting environment.

The framework component provides furthermore authentication and authorization of users over distinguished names and WS-SecureConversation. Three user groups exist (administrators, SLA owners, and SLA users) with different access privileges (read/write access to all SLAs, read/write to specific SLAs, and read only to specific SLAs respectively). This is important because a third party might need the privilege to monitor an SLA while not possessing the privilege to terminate it. By default only the creator of an SLA may access it.

Templates are stored in a relational database. They can be queried but also modified over WS-ResourceFramework mechanisms that respect access privileges. The template store and also individual agreements can be monitored using WS-Notification mechanisms. This allows a broker for example to be notified of new templates or a provider to wait for the finalization of an outsourced SLA.

During the creation of an agreement, the Negotiation Manager tests agreement offers against the creation constraints and rejects offers that violate those. The test for compliance does not yet support the full set of constraints proposed by WS-Agreement. Simple numeric comparisons (e.g. to restrict the number of processors requested) and string comparisons for enumerations (e.g. to request a specific operating system) are supported as of now.

Several features were considered too domain specific to be included in the Negotiation Manager framework and have been pushed up into the Assess Grid specific implementation. These can be easily extracted and copied into new projects. Example for such features are:

- Persistence of SLA instances
- Credential Delegation

- An asynchronous state-machine that reacts upon events (stemming from a RMS or triggered by wall clock time) by transitions or operations (e.g. a file staging operation can be triggered at a certain point in time or because the RMS has finished a computation)
- Cancellation Policies that describe a cancellation fee of an SLA at a certain point in time and guarantee over time monotonously increasing cancellation fees
- Rejection reasons for rejected SLA requests
- Helper classes to conveniently modify xs:any tags
- A quote (invitation to treat) mechanism to request prices and the availability of resources without creating a binding agreement.
- Client

For details we refer to AssessGrid Deliverable D4.2, which is available from www.assessgrid.eu.

4.5 BEinGRID

Name: BEinGRID Negotiation Manager

Category: Component provided as a Web Service – and architecture based on Web Services

License type: Apache V2

Link: https://gforge.beingrid.eu/gf/project/slanegotiator

See also https://gforge.beingrid.eu/gf/project/sla4gt4/ for a complete SLA framework supporting WS-Agreement (March 2007 specification)

Description: BEinGRID – Business Experiments in Grid - is an ICT FP6 project of initially 75 partner organisations. The main objective of BEinGRID is to foster the adoption of Grid technologies for businesses and thereby crossing the chasm between the early market dominated by few visionary customers and the mainstream market dominated by a large number of pragmatic customers.

The BEinGRID project released 25 so called Grid Business experiments (BEs). Based on a clear business case, each BE developed a prototypic implementation for their specific requirements. Several BEs required SLAs, and provided a support for requirement analysis, which we hope has been wide enough. The obligation to accommodate different needs produced a generic architecture for SLAs, which accommodates SLA Negotiation, SLA Evaluation, SLA accounting, as well as scheduler optimization through SLAs.

Grid ecosystems: One of the BEinGRID SLA analysis lines followed the BEs that were based on GRIA (a service-oriented infrastructure designed to support B2B collaborations). These use a different SLA specification and are not developed in this document. On the other hand, three different BEs decided to use SLAs in their Globus Toolkit 4 environment. One dealt with Mobile Fraud Management, another with dynamic capacity markets, and the last one targeted eHealth. They are described below.

BEinGrid BE20 – Mobile Fraud Management

A central Fraud Management System (FMS) offers its fraud detection and management services to a group of mobile network operators (MNO). When an operator joins the group and wishes to use the FMS, new accounts for their Fraud Analysts and FMS manager are created for this operator. The FMS administrator needs to configure the different subsystems for the new operator (data management and data federation parameters, and other). The MNOs negotiate a Service Level Agreement with the FMS system, in order to regulate the QoS (availability, response time), which will be provided as well as protection and coverage guarantees. The established SLA contract will be later monitored to ensure its fulfilment, and it could be dynamically re-negotiated by the operators.

The main purpose of the SLA is to set the DataFederation information (included in one of the Service Description Terms (SDT)). Also, at negotiation time it is checked that the consumer server is accessible (what we call "Consumer Obligations"), but that check is only done at negotiation time. The telecom operators and the fraud management system have a high-level, legally binding contract, which is the one which actually regulates their business relationship.

BEinGrid BE22 – Agrogrid

AgroGrid develops and implements a full life-cycle solution for dynamic capacity markets, mainly supply chain companies in the European food and agriculture industry.

Traditionally, companies in the agricultural sector operate in business structures with long-term contract relations. To react on unexpected or unplanned changes in supply and demand of capacities, the BE enables companies to deploy their capacities extensively and, simultaneously, to ensure food safety via efficient tracking & tracing of goods and automated SLA-Monitoring & Evaluation.

The SLA-Negotiation is provided to the user through a portlet (based on gridsphere) inside the AgroGrid portal. It serves as frontend to the BEinGRID SLA-Negotiator implementation, which allows negotiation of SLAs between capacity requester and provider (delivery of food between food-provider and food-consumer).

BEinGrid BE25 – Business Experiment in enhanced Intensity-Modulated Radiation Therapy (IMRT) planning using Grid services on-demand (BEinEIMRT)

BEinEIMRT provides on-demand e-Health computational services to Health organisations like clinics and hospitals. The health organisations produce medical images, and rely on BEinEIMRT for services like tumor detection and radiotherapy planning. All services are provided by the Centro de

Supercomputación de Galicia (CESGA), which is the customary provider. CESGA has a set of computational resources on which to execute the submitted tasks. When CESGA is under peak demand, its resources cannot match the QoS, which has been signed in a framework agreement. To comply with this contract, an external resource provider is contacted. External resources are then added for a limited time period to the CESGA infrastructure, so the framework SLAs can be respected. The extra resources are needed on a short period (several hours to a maximum of a week). The external resources are obtained through the signature of an SLA with the external provider, using the BEinGRID SLA Negotiator component.

The negotiation is automatic and performed by the scheduler (GridWay), within the limits of the preSLAs the administrator has defined according to the framework agreement with the external provider. Furthermore, the CESGA administrator can configure some additional parameters about the terms and conditions of the negotiation.

4.6 BREIN

Name: SLA Negotiation

Category: Component in a SLA Management Framework

License type: LGPL

Link: http://www.eu-brein.com/

Description: Brein is a European project. It provides an e-business concept developed in recent Grid research, namely the concept of so-called "dynamic virtual organisations" towards a more business-centric model, by enhancing the system with methods from artificial intelligence, intelligent systems, semantic web etc. Thus, the BREIN project will enable business participants to easily and effectively use Grid technologies for their respective business needs.

Grid ecosystems: The SLA management Framework is based on Globus Toolkit 4 enviroment and apply the GT4 components for WS-Notifcation, object serialization.

WS-Agreement implementation: In general, SLA Management components are based on the WS-Resource design pattern, because the component was implemented as GT4 Java WS Core service, which in turn allows the implementation of stateful services as defined by WS-RF. The SLA contracts are based on WS-Agreement specification, which defines schemas for SLA Templates.

SLA Negotiation provides two main functionalities. One is to manage Templates, which include the task of creating SLA Templates, storing the Templates in a repository and retrieve the templates. The implementation of the interfaces is suggested by WS-Agreement. The other functionality is to compare the received offers and provide the best bid to the customer.

After the negotiation, the SLA Manager will start the monitors and supervise the resource usage in order to make sure, if the agreements are met.

4.7 CATNETS

Name: Catallaxy paradigm for decentralized operation of dynamic application networks

Category: Agent-based framework for service markets

License type: Open Source, license defined by the CATNETS project

Link: http://www.catnets.uni-bayreuth.de/

Description: CATNETS is a project of several universities and research centers across Europe with the objective to determine the applicability of a decentralized economic self-organization mechanism for resource allocation in application layer networks (ALN), which include Grid systems. The name CATNETS is based on an economic self-organization approach of a free market, the Catallaxy. CATNETS simulates the ALN environment by an economy, where the resources are for example processor time or storage space, while the economic actors are computers or web services. The application service and compute resource allocation of Application Layer Networks is broken down into two types of interrelated markets: A Grid resource market, where computational and data resources are traded and a service market where application services are traded. These services provide particular application functionality, e.g. query execution or molecule docking. In these separate markets complex services buy basic services, which buy raw resources. In this Catallaxy approach, the market is self-organizing which means that no centralized broker is required.

Grid ecosystems: In the prototype implementation the middleware is implemented as a set of simple specialised agents using the light-weight agents platform of the Decentralised Information Ecosystem Technologies (DIET) project. The agents provide for example access to markets, negotiations, object discovery and communication. The management of local resources is based on the WS-Resource Framework offered by Globus Toolkit 4. Middleware is further implemented using JXTA technology.

WS-Agreement implementation: WS-Agreement is used in the implementation of both the service market and the resource market. CATNETS defines separate bidding language for the service and the resource market, which are used by agents to submit bids for services or resources. These languages are mapped onto WS-Agreement via domain-specific schemes. The offers are encoded in XML using WS-Agreement and JSDL. In the resource market basic services can submit sell orders to the order books with WS-Agreement and the resource services can submit buy orders to the order books. After submission of all bids to the auctioneer, the allocation and the corresponding prices are determined, which results in an agreement. The activity on the service market is quite similar. The WS-Agreement implementation of CATNETS is technically integrated into the Triana workflow engine, which allows visualisation of Agreement Templates and Offers. It also enables the workflow to be paused until an Agreement Offer has been confirmed.

4.8 Umeå University

Name: Job Submission Service (JSS)

Category: Scheduling & Resource Management framework

License type: Apache License, Version 2.0

Link: http://www.cs.umu.se/research/grid/jss/index.html

Description: JSS developed at the Umeå University is a broker aiming at identifying the set of resources that minimizes the Total Time to Delivery (TTD), or part thereof, for each individual job submission. In order to do this, the broker makes an a priori estimation of the whole or parts of the TTD for all resources of interest before making the selection. The TTD estimation includes performing benchmark-based execution time estimations, estimating file transfer times, and performing advance reservations of resources in order to obtain a guaranteed batch-queue waiting time. For resources not providing all information required or a reservation capability, less accurate estimations are performed. On the Grid resource, an authorization callout mechanism is used to validate that (i) the requested reservation identifier exists and (ii) the reservation actually was created by the same user that submits the job. JSS is currently used e.g. in NorduGrid and Swegrid.

Grid ecosystems: JSS with integrated WS-Agreement is currently available for two Grid middleware environments: JSS provides support for Globus Toolkit 4 and NorduGrid/ARC. Integration with another Grid middleware only consists of writing the authorization plug-in described above.

WS-Agreement implementation: JSS is using a WS-Agreement implementation that was done at the Umeå University. The implementation originally was planned to be based on Cremona (an early WS-Agreement implementation by IBM), which turned out not to be feasible as IBM did not give access to the source code. The GT4/WSRF-based implementation is rather straightforward, with createAgreement() requests forwarded by the AgreementFactory to a decision making plug-in, which interacts with the local resource management system. One such plug-in is used to create advance reservations, and hence interacts with the batch system scheduler. JSS currently support the Maui scheduler, although others (supporting advance reservation) easily can be added. JSS also implements a two-phase mechanism, where reservations will be released shortly after their creation, unless they are confirmed. This is implemented using WS-Resource Lifetime, as each Agreement is modelled as a WS-Resource. The terms used to negotiate advance reservations are

- number of CPUs,
- duration,
- earliest allowed start time,
- latest allowed start time,
- malleability flag.

The semantics of a malleable reservation is that the local scheduler may modify the reservation start time freely, as long as the start time stays in the [earliest, latest] allowed start time window. Previous research demonstrates that this behaviour reduces the utilization drop induced by using advance reservations. JSS has however, due to lack of support in the local scheduler, not been able to implement malleable reservations, although the system is prepared to support it. Reservations are created by the job submission service

during resource brokering. When the job is submitted to the selected resource, an identifier for the created reservation is included in the job description.

4.9 SmartLM

Name: WS-Agreement for Java (WSAG4J)

Category: Generic framework

License type: BSD

Link: http://packcs-e0.scai.fraunhofer.de/mss-project/index.html

Description: SmartLM is a European project funded in FP7. SmartLM aims at rendering mechanisms for managing and using software licenses in a more fair and flexible way. SmartLM licenses may be used seamlessly in local cluster environments, as well as in local or remote Grid and Cloud environments, and under circumstances that the SOA concept presents. In SmartLM licenses are managed as agreements between the user and the license management system, extending the conventional Service Level Agreements (SLAs) which are made today between sellers and buyers in the market. The SLA defines the terms of license usage for running a license-protected application. These terms comprise e.g. the features of the application to be used, the number of processors for parallel execution, the estimated time of the application execution, the price for using the license. Negotiation is supported when the license is requested, e.g. to allow for co-allocation of computational resources and licenses, to find a suitable time for the execution with the requested features or to minimise the price. Re-negotiation is supported during run-time, e.g. for extending or reducing of the estimated-run time, to add or remove features. Negotiation and re-negotiation are implemented on top of WS-Agreement while keeping the enhanced version of WS-Agreement upward compatible. More details can be found on the project website: http://www.smartlm.eu.

Grid ecosystems: SmartLM has a licence management service that uses WS-Agreement as interface to either the user's client or a meta-scheduling service managing the co-allocation. As meta-scheduling service we use the MSS (developed in the VIOLA and PHOSPHORUS project as described in the next section). Basically, SmartLM itself is middleware independent, we have testbeds with both UNICORE 6 and Globus Toolkit 4.2. The MSS also is middleware independent using different adapters for different middleware systems, currently UNICORE 6 and Globus Toolkit 4.2. Integration with another Grid middleware only requires writing another adapter.

WS-Agreement implementation: SmartLM uses the WSAG4J framework (see section 6 for details), which is a WS-Agreement implementation in Java that has been developed at the Fraunhofer Institute SCAI. For the description of jobs we use JSDL. However, since JSDL does not allow specifying license terms along with the applications a new schema has been defined in SmartLM to describe license terms. Terms used to describe a license are, e.g.

- features of an application to be used,
- number of CPUs for parallel execution of the application,

- estimated duration,
- earliest start-time,
- latest end-time.

WS-Agreement is used both for SLAs on the computational resources for the execution of the license-protected application and the SLAs for reserving a specific license for a dedicated time.

4.10 VIOLA/PHOSPHORUS/IANOS

Name: MetaScheduling Service

Category: Meta-scheduler

License type: BSD

Link: http://packcs-e0.scai.fraunhofer.de/mss-project/index.html

Description: In the VIOLA project an optical test-bed between multiple partners in Germany has been implemented. The main goals were the test of advanced network architectures, development of software for user-driven dynamical provision of bandwidth and test of parallel applications. The project ended in April 2007 but the MSS was and is being further developed in a number of other projects like PHOSPHORUS [http://www.ist-phosphorus.eu/] or IANOS [http://www.ianos.org].

Grid ecosystems: Grid applications in VIOLA were run on three Linux-based PC-Clusters, a SUN-Cluster and a Cray X-D1 with a total peak performance of 900 GFLOPS. The VIOLA Grid is based on UNICORE. A single instance of a MetaScheduling Service integrated into the UNICORE middleware is able to perform co-ordinated CPU and network bandwidth reservation between the clusters in the Grid, enabling distributed applications on these systems [http://www.viola-testbed.de/].

WS-Agreement implementation: The VIOLA MetaScheduling Service MSS is responsible for negotiation of resource allocation with the local scheduling systems. It is implemented as a Web Service receiving a list of resources preselected by a resource selection service. The resource reservation is based on WS-Agreement. Network resources are reserved through a WS-Agreement Interface with the Adapter of the NRMS ARGON (VIOLA) and the HARMONY network resource brokering system (PHOSPHORUS). Resource reservations are negotiated through adapters with local scheduling systems also using WS-Agreement. Furthermore, the negotiation between the MSS and the UNICORE Client is based on WS-Agreement. When a UNICORE Client wants to make a reservation, it sends the resource request to the MSS as a WS-Agreement template. The MetaScheduling Service then negotiates a potential start time for the Job and requests reservation of the network and computational resources. After successful completion of this reservation the MSS sends an End Point Reference of the created WS-Agreement back to the UNICORE Client.

5. Constructs used in WS-Agreement – An Analysis

5.1 Introduction

The following projects participated in the questionnaire and contributed to this document (Abbreviation used in the table is given in parentheses):

- AssessGrid (AG)
- Job Submission Service (JSS)
- Phosphorus (WSAG4U)
- BREIN (BREIN)
- BEinGRID (BE20, BE22, and BE25 are different business experiments within BEinGRID)
- AgentScape (AS)

The following legend is used to describe what has been implemented:

- y = yes
- n = no
- p = partially
- - = no because parent element not implemented

5.2 Top level Agreement-element

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/Agreement	y	y	y	y	y	y	y	y

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@AgreementId	y [1]	n	y [2]	y	y	y	y	

[1] UUID generated by Agreement Initiator, needs to be globally unique

[2] Counter that specifies the version of an agreement (will be incremented when an agreement changes when re-negotiated)

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/Name	y	y	y	y	y	y	y	y

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/AgreementContext	y	y	y	y	y	y	y	y

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/Terms	y	y	y	y	y	y	y	y

5.3 Context-element

Element /Context	AG y	JSS y	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS y

Element @any attribute	AG n	JSS n	WSAG4U n	BREIN n	BE20 n	BE22 n	BE25 n	AS

Element /AgreementInitiator	AG y (1)	JSS y	WSAG4U n (2)	BREIN y	BE20 y	BE22 y	BE25 n	AS y

(1) Custom tag that contains a Distinguished Name identifying the user, such as:

```
<wsag:AgreementInitiator xsi:type="assessgrid:DistinguishedName_Type">
  /O=Grid/OU=GlobusTest/OU=simpleCA-assessgrid/CN=Dominic Battre
</wsag:AgreementInitiator>
```

(2) to be implemented in future version

Element /AgreementResponder	AG y (1)	JSS n	WSAG4U n (2)	BREIN y	BE20 y	BE22 y	BE25 n	AS y (3)

(1) Custom tag that contains a Distinguished Name identifying the user, see AgreementInitiator

(2) to be implemented in future version

(3) AgreementProvider

Element /ServiceProvider	AG y (1)	JSS n	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS

(1) The ServiceProvider is always the Agreement Responder

Element /ExpirationTime	AG y (1)	JSS n	WSAG4U n (2)	BREIN y	BE20 y	BE22 y	BE25 y	AS y (3)

(1) This is part of the template but ignored, as the meaning is not quite clear: Either it means (1) at the time of the ExpirationTime, the SLA loses its meaning (maybe then this should be part of the guarantees), (2) at this time, the WS-Resource is deleted

(2) to be implemented in future version

(3) Duration

Element /TemplateId	AG y (1)	JSS n	WSAG4U y (2)	BREIN y	BE20 y	BE22 y	BE25 y	AS

(1) This is used to distinguish two kinds of templates with different structure (1) regular SLA for a job submission, (2) SLA used to outsource a checkpointed job that needs to be executed on a remote site.

(2) Required

Element /TemplateName	AG y	JSS n	WSAG4U y (1)	BREIN y	BE20 y	BE22 y	BE25 y	AS y (2)

(1) Required

(2) Template

Element /any	AG n	JSS n	WSAG4U y (1)	BREIN y (2)	BE20 n	BE22 n	BE25 n	AS

(1) Template based sessions, Negotiation

(2) For BEinGRID, added a pricing element for the whole agreement

5.4 Terms

Element @Name of Term base-element	AG y (1)	JSS n	WSAG4U y (2)	BREIN n	BE20 y	BE22	BE25	AS n

(1) Used when checking creation constraints in order to report problems

(2) Used at template design time

Element /Terms	AG y	JSS y	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS y

Element /All	AG y	JSS y	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS y

Element /All (nested)	AG n	JSS n	WSAG4U y	BREIN n	BE20 y	BE22 n	BE25 n	AS

Element /OneOrMore	AG n	JSS n	WSAG4U y	BREIN n	BE20 n	BE22 n	BE25 n	AS

Element /ExactlyOne	AG n	JSS n	WSAG4U y	BREIN n	BE20 n	BE22 n	BE25 n	AS

Element /ServiceDescription-Term	AG y (1)	JSS y	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS y

(1) Description of the service to be delivered using JSDL-POSIX or JSDL-SPMD plus some extension regarding probability of failure guarantees. An entire job is described as one JSDL job definition which is contained in a single ServiceDescriptionTerm.

Element /ServiceReference	AG n	JSS n	WSAG4U n	BREIN y	BE20 n	BE22 y	BE25 n	AS

Element /ServiceProperties	AG n	JSS y	WSAG4U n (1)	BREIN y	BE20 n	BE22 n	BE25 n	AS

(1) To be implemented for automatic guarantee term evaluation

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS

/GuaranteeTerm	y [1]	y	y	y	y	y	y	

[1] The AssessGrid project had problems modeling the SLAs because in their scenario, SLAs can be either fulfilled (job was executed correctly) or violated (job failed because a machine crashed). The SLA consists of several guarantees, but it is desired that the *entire* SLA succeeds or fails. Therefore, most guarantees have no reward and penalty. Two meta-guarantees (ProviderFulfillsAllObligations and ConsumerFulfillsAllObligations) are introduced that carry the reward and penalty of the entire SLA. The following invariant holds:
- ProviderFulfillsAllObligations.reward = ConsumerFulfillsAllObligations.penalty = SLA reward
- ProviderFulfillsAllObligations. penalty = SLA penalty
- ConsumerFulfillsAllObligations.reward = 0

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/ServiceDescriptionTerm	y	y	y	y	y	y [1]	y	y

[1] Defining the product characteristics through <metric name="..." type="..." unit="..."> tags

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@Name	y	y	y	y	y	y	y	y

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@ServiceName	y [1]	n	y	y	y	y	y	y

[1] In the Provider implementation (where the Negotiation Manager encapsulates the RMS) this attribute is ignored, as only one service is allowed per SLA. In the Broker implementation, the broker maps workflows to SLAs with one or more providers. The service names are used to identify the individual tasks.

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/any	y	y	y	y	n	y	y [1]	

[1] Defining the remote resource characteristics through several SDT terms, each with a different tag content: numberOfCPUs, cpuMemory, cpuStore, numberCPU, pricePerHour

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/ServiceReference	n [1]	n	n	n	n	y	n	n

[1] The AssessGrid developers did not understand how to use this from the specification

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@Name	-	-	-	-	-	y	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@ServiceName	-	-	-	-	-	y	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/any	-	-	-	-	-	y [1]	-	-

[1] Specifying the reference through an element tag with contains a URL

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/ServiceProperties	n	y	n [1]	n	n	y [1]	n	n

(1) The properties below will be implemented and used for automatic guarantee term evaluation in a later release

(2) Basically describes the tags that are found in the SDT found previously

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@Name	-	n	-	-	-	y	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@ServiceName	-	n	-	-	-	y (1)	-	-

(1) Cross references to SDT above

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/VariableSet	-	y	-	-	-	y	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/Variable	-	y	-	-	-	y	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@Name	-	y	-	-	-	y	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@Metric	-	N	-	-	-	y	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/Location	-	n	-		-	n (1)	-	-

(1) The location tag is used but has no content

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/GuaranteeTerm	y	y	y	y	y	y	y	n (1)

(1) Under active development

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@Name	y	y	y	n	y	y	y	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@Obligated	y (1)	n	?	n	y	y	y	-

(1) Guarantees by both parties are employed

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/ServiceScope	y	y	y	n	y	y (1)	n	-

(1) Cross references to SDT above

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
@ServiceName	y (1)	y	y	-	y	y	-	-

(1) See ServiceDecriptionTerm for details

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
Any	n	n	n	-	n	n	-	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/QualifyingCondition	y (1)	n	n	n	n	n	n	-

(1) The AssessGrid project supports a concept called cancellation policies. This allows SLAs to be canceled for a low price during a certain period of time before the earliest start time. This period is described by an time interval that may be open to either side or closed on both sides as shown in the example below:

```
<ns1:GuaranteeTerm ns1:Name="CancellationPolicy2" ns1:Obligated="ServiceConsumer">
   <ns1:ServiceScope ns1:ServiceName="ARMINIUS@UPB"/>
   <ns1:QualifyingCondition xsi:type="ns10:TimeInterval_Type"
   xmlns:ns10="http://www.assessgrid.eu/2007/02/types"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <ns10:endExcl>2009-02-23T19:45:00.749Z</ns10:endExcl>
   </ns1:QualifyingCondition>
   <ns1:ServiceLevelObjective>
   <ns1:CustomServiceLevel xsi:type="ns11:ConsumerDoesNotCancelAgreement_Type"
xmlns:ns11="http://www.assessgrid.eu/2007/02/types"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
   </ns1:ServiceLevelObjective>
   <ns1:BusinessValueList>
   <ns1:Penalty>
       <ns1:AssessmentInterval xsi:nil="true"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
    <ns1:ValueUnit>EUR</ns1:ValueUnit>
       <ns1:ValueExpression xsi:type="ns12:ValueExpression_Type"
        xmlns:ns12="http://www.assessgrid.eu/2007/02/types"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            -0,5
       </ns1:ValueExpression>
     </ns1:Penalty>
    </ns1:BusinessValueList>
</ns1:GuaranteeTerm>
```

The penalty is negative in order to represent a refund (the user has to pay the SLA penalty because it violated the ConsumerDoesNotCancelAgreement guarantee.

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/ServiceLevelObject-ive	y	y	n	y	y	y	y	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/BusinessValueList	n	n	n	y	y	y	y	-

Element	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
/ServiceLevelObjective	y	n/a (1)	n	y	y	y	y	-

(1) This was an xsd:anyType in the draft implemented and is hence used rather freely (see example)

Element /KPITarget	AG n	JSS -	WSAG4U -	BREIN y	BE20 n	BE22 y	BE25 y	AS -

Element /KPIName	AG -	JSS -	WSAG4U -	BREIN y	BE20 -	BE22 y	BE25 y	AS -

Element /Target	AG -	JSS -	WSAG4U -	BREIN y	BE20 -	BE22 y	BE25 n	AS -

Element /CustomServiceLevel	AG y (1)	JSS -	WSAG4U -	BREIN y	BE20 y	BE22 y (2)	BE25 y (3)	AS -

(1) All SLOs are represented by simple XML Elements:

```
<wsag:CustomServiceLevel xsi:type="assessgrid:ProviderFulfillsAllObligations_Type"/>
<wsag:CustomServiceLevel xsi:type="assessgrid:ConsumerFulfillsAllObligations_Type"/>
<wsag:CustomServiceLevel xsi:type="assessgrid:ScheduleRestriction_Type">
    <assessgrid:EarliestStartTime>
         2007-03-01T00:00:00+01:00
    </assessgrid:EarliestStartTime>
    <assessgrid:LatestFinishTime>
         2007-03-01T12:00:00+01:00
    </assessgrid:LatestFinishTime>
</wsag:CustomServiceLevel>
<wsag:CustomServiceLevel xsi:type="assessgrid:MaxStageInDuration_Type">
    PT20S
</wsag:CustomServiceLevel>
<wsag:CustomServiceLevel xsi:type="assessgrid:MaxStageOutDuration_Type">
    PT20S
</wsag:CustomServiceLevel>
<wsag:CustomServiceLevel xsi:type="assessgrid:ConsumerDoesNotCancelAgreement_Type" />
```

(2) Defines the evaluation of the guarantee through functions like “EXACT, GREATERTHAN, etc.”

(3) Defines the evaluation of the guarantee through comparison functions like “< 0.8”

Element /BusinessValueList	AG n	JSS n (1)	WSAG4U	BREIN y	BE20 y (2)	BE22 y (3)	BE25 y	AS n (4)

(1) see comment below

(2) tag is present, but never used or assessed

(3) Tag is present, but used only on the Price Guarantee Term

(4) Under active development

Element /Importance	AG -	JSS -	WSAG4U	BREIN n	BE20 y (1)	BE22 n	BE25 n	AS -

(1) tag is present, but never used or assessed

Element /Penalty	AG -	JSS -	WSAG4U	BREIN y	BE20 y (1)	BE22 n	BE25 n	AS -

(1) tag is present, but never used or assessed

Element /Reward	AG -	JSS -	WSAG4U -	BREIN n	BE20 n	BE22 n	BE25 n	AS -

Element /Preference	AG -	JSS -	WSAG4U -	BREIN n	BE20 n	BE22 n	BE25 y	AS -

Element /CustomBusinessValue	AG -	JSS -	WSAG4U -	BREIN n	BE20 n	BE22 y (1)	BE25 n	AS -

(1) tag is present, but only used in the Price Guarantee Term

Element /Penalty	AG y	JSS n (1)	WSAG4U n (2)	BREIN y	BE20 y (3)	BE22 n	BE25 n	AS n (4)

(1) The project targets a traditional academic computing environment where compute resources are allocated to researches by a committee based on scientific contributions. In such an environment, where resource consumption is not based on economic compensation (although accounting and quota enforcement do occur), it makes little sense to discuss business value and/or economical penalty.

(2) the properties below will be implemented and used for automatic guarantee term evaluation in a later release

(3) tag is present, but never used or assessed

(4) Under active development

Element /AssessmentInterval	AG y (1)	JSS n	WSAG4U -	BREIN y	BE20 y	BE22 -	BE25 -	AS -

(1) This is only used to fulfill the schema requirements. The RMS detects violations itself, so this is not based on polling but on pushing events.

Element /TimeInterval	AG y (1)	JSS -	WSAG4U -	BREIN y	BE20 y	BE22 -	BE25 -	AS -

(1) This is only used to fulfill the schema requirements. The RMS detects violations itself, so this is not based on polling but on pushing events.

Element /Count	AG n (1)	JSS -	WSAG4U -	BREIN n	BE20 n	BE22 -	BE25 -	AS -

(1) This is not used but the meaning is not clear either.

Element /ValueUnit	AG y (1)	JSS -	WSAG4U -	BREIN y	BE20 y	BE22 -	BE25 -	AS -

(1) Only one currency supported (“EUR”)

Element /ValueExpression	AG y (1)	JSS -	WSAG4U -	BREIN y	BE20 y	BE22 -	BE25 -	AS -

(1) The value expression contains only constant numbers like for example:
<wsag:ValueExpression xsi:type="assessgrid:ValueExpression_Type"> 1000.0 </wsag:ValueExpression>

Element /Preference	AG n	JSS -	WSAG4U n (1)	BREIN n	BE20 n	BE22 n	BE25 y	AS n (2)

(1) the properties below will be implemented and used for automatic guarantee term evaluation in a later release

(2) Under active development

Element /ServiceTermRef-erence	AG -	JSS -	WSAG4U -	BREIN -	BE20 -	BE22 -	BE25 y (1)	AS -

(1) References one of the SDTs above

Element /Utility	AG -	JSS -	WSAG4U -	BREIN -	BE20 -	BE22 -	BE25 y	AS -

5.5 Templates

Element /Template	AG y (1)	JSS y (2)	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS y

(1) Templates are stored statically; they are not generated at runtime.
(2) This is supported, but only to the extent that the AgreementInitator uses the name of the template to ensure that the Provider supports a certain type of Agreements. No CreationConstraints, etc. are used, as the availability of a free time slot (a backfill window) does not necessarily mean that any user may create a reservation for that slot. We hence resort to a per-case based negotiation with minimal pre-knowledge of constraints.

Element @TemplateId	AG y	JSS n	WSAG4U y	BREIN y	BE20 ?	BE22 y	BE25 y	AS

Element /Name	AG y	JSS y	WSAG4U y	BREIN y	BE20 ?	BE22 y	BE25 y	AS y

Element /AgreementContext	AG y	JSS n	WSAG4U y	BREIN y	BE20 ?	BE22 y	BE25 y	AS y

Element /Terms	AG y	JSS n	WSAG4U y	BREIN y	BE20 ?	BE22 y	BE25 y	AS y

Element /CreationConstraints	AG y (1)	JSS n	WSAG4U y (2)	BREIN n	BE20 ?	BE22 n (3)	BE25 n (4)	AS y

(1) Creation constraints are not completely supported by an XML Schema validator. This is probably not possible in the Globus Toolkit hosting environment because namespaces get lost at an early stage.

(2) Generic validation support for arbitrary Creation Constraints that use ItemConstraint

(3) Creation Constraints are not used, instead the service properties are used to define the service terms – but no limitations on bounds are provided

(4) Creation Constraints tags exists but empty. Creation Constraints tags are not used, instead the constraints are defined in a framework agreement

Element /Item	AGy	JSS-	WSAG4Uy	BREIN-	BE20?	BE22-	BE25-	ASy

Element /Constraint	AGn	JSS-	WSAG4Uy	BREIN-	BE20?	BE22-	BE25-	AS

Element /Item	AGy	JSSn	WSAG4Uy	BREINn	BE20y	BE22n	BE25-	ASy

Element @Name	AGy (1)	JSS-	WSAG4Uy	BREIN-	BE20y	BE22-	BE25-	AS

(1) Used for reporting problems

Element /Location	AGy (1)	JSS-	WSAG4Uy	BREIN-	BE20y	BE22-	BE25-	ASy

(1) XPath expressions are used with the exception that namespace-prefix binding is fixed and not related to the namespace-prefixes in the SOAP message. This is because the binding gets lost in a Globus Toolkit environment when the SOAP message is translated into a Java Bean.

Element /ItemConstraint	AGp (1)	JSS-	WSAG4Uy	BREIN-	BE20y	BE22-	BE25-	AS

(1) Supported are:

- xs:(max,min)(In,Ex)clusive
- restrictions to enumerations of strings

Overall the implemented validation is rather limited and the suggested specification is considered very difficult to implement.

Element /restriction	AGp	JSS-	WSAG4Uy	BREIN-	BE20n	BE22-	BE25-	ASy

Element /group	AGn	JSS-	WSAG4Uy	BREIN-	BE20n	BE22-	BE25-	AS

Element /all	AGn	JSS-	WSAG4Uy	BREIN-	BE20n	BE22-	BE25-	AS

Element /choice	AGn	JSS-	WSAG4Uy	BREIN-	BE20n	BE22-	BE25-	AS

Element /sequence	AGn	JSS-	WSAG4Uy	BREIN	BE20n	BE22-	BE25-	AS

Element /any	AGn	JSS-	WSAG4Un (1)	BREIN-	BE20y	BE22-	BE25-	AS

(1) Will be used in future releases to specify/enforce cardinality of XPath results

5.6 Agreement States

Agreement States	AGy	JSSn (1)	WSAG4Uy	BREINy	BE20n (2)	BE22?	BE25?	ASn

(1) In this project work, the authors concluded that monitoring of the state of an advance reservation did not fit into the envisioned state model. An advance reservation either exists, or stops to exist, the latter case being an unrecoverable error. This contrasts, to e.g., state monitoring of a guaranteed network bandwidth agreement.

(2) The Agreement is not currently being monitored, but will be in the future – all states would be used.

State	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
Pending	y	-	y	n	-			-
Pending and Terminating	n	-	n	n	-			-
Observed	y	-	y	y	-			-
Observed and Terminating	n	-	y	n	-			-
Rejected	n (1)	-	n	n	-			-

(1) Agreements are rejected with an exception at the time of the createAgreement call

State	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
Complete	n	-	y	n	-			-
Terminated	y	-	y	y	-			-

5.7 Service Run-time States

Service Run-time States	AGy	JSSn	WSAG4Uy	BREINn	BE20n (1)	BE22?	BE25?	ASn

(1) The Agreement is not currently being monitored, but will be in the future – all states would be used.

State	AG	JSS	WSAG4U	BREIN	BE20	BE22	BE25	AS
Not Ready	y	-	y	-	-			-

State Ready	AG y	JSS -	WSAG4U y	BREIN -	BE20 -	BE22	BE25	AS -

State Processing	AG n	JSS -	WSAG4U y	BREIN -	BE20 -	BE22	BE25	AS -

State Idle	AG n	JSS -	WSAG4U n	BREIN -	BE20 -	BE22	BE25	AS -

State Completed	AG y	JSS -	WSAG4U y	BREIN -	BE20 -	BE22	BE25	AS -

5.8 Guarantee States

Guarantee States	AG y	JSS n	WSAG4U y	BREIN n	BE20 n [1]	BE22 ?	BE25 ?	AS n

[1] The Agreement is not currently being monitored, but will be in the future – all states would be used.

State Fulfilled	AG y	JSS -	WSAG4U y	BREIN -	BE20 -	BE22	BE25	AS -

State Violated	AG y	JSS -	WSAG4U y	BREIN -	BE20 -	BE22	BE25	AS -

State Not Determined	AG y	JSS -	WSAG4U y	BREIN -	BE20 -	BE22	BE25	AS -

5.9 Port Types

Port Type Agreement Factory	AG y	JSS y	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS n

Port Type Pending Agreement Factory	AG n	JSS n [1]	WSAG4U n	BREIN y	BE20 n	BE22 ?	BE25 ?	AS n

[1] Did not exist in draft specification

Port Type Agreement	AG y	JSS y	WSAG4U y	BREIN y	BE20 y	BE22 y	BE25 y	AS n

Port Type Agreement Acceptance	AG n	JSS n [1]	WSAG4U n	BREIN n [2]	BE20 n	BE22 ?	BE25 ?	AS n

[1] Did not exist in draft specification

[2] Pending Agreement Factory but not Agreement Acceptance?

Port Type Agreement State	AG y	JSS n	WSAG4U y	BREIN y	BE20 n	BE22 ?	BE25 ?	AS n

Port Type Custom Port Types	AG y [1]	JSS n [2]	WSAG4U n [3]	BREIN n	BE20 n	BE22 ?	BE25 ?	AS n

[1] see below

[2] see below

[3] WSRF-ServiceGroups, WSRF-Lifetime

6. WSAG4J – A Generic WS-Agreement Framework

In this section, WSAG4J is detailed, as it is found to be one of the most advanced and complete software package which implements the WS-Agreement 1.0 specification. Nonetheless, there are competing solutions, which can also be considered, for example those listed in chapter 4. The most up-to-date list is maintained by the GRAAP-WG: https://forge.gridforum.org/sf/wiki/do/viewPage/projects.graap-wg/wiki/Implementations

WS-Agreement for Java (WSAG4J) is a generic implementation of the WS-Agreement protocol. It supports common functionality to create and monitor agreements in a generic way and enables users to quickly build and deploy WS-Agreement based services.

WSAG4J implements the Agreement Factory port type for the creation of agreements, and the Agreement State port type for monitoring the states of existing agreements. At the current state, WSAG4J does not support the Pending Agreement Factory port type and the Agreement Acceptance port type; this will be subject for a future release.

WSAG4J follows a declarative approach to support and manage the whole lifecycle of an agreement, starting from the definition of an agreement template, over the deployment of the templates in factories, to the management of the agreement itself. The main components and their interaction are depicted in Figure 1. The main features of WSAG4J are listed below:

- Host multiple agreement factories within one WSAG4J engine
- Customizable persistence layer for Agreement Factory and Agreement instances
- Default implementation of persistence layer based on configuration file persistence
- Easy to use server and client side API
- Free configurable factory actions within one WSAG4J engine
- Definition of Agreement Templates based on XML files
- Dynamic template creation by using macros in template definition
- Dynamic agreement offer validation based on creation constraints
- Dynamic agreement guarantee evaluation and generation of accounting events
- Negotiation of agreement templates
- Support of WS-Security and WS-Policy

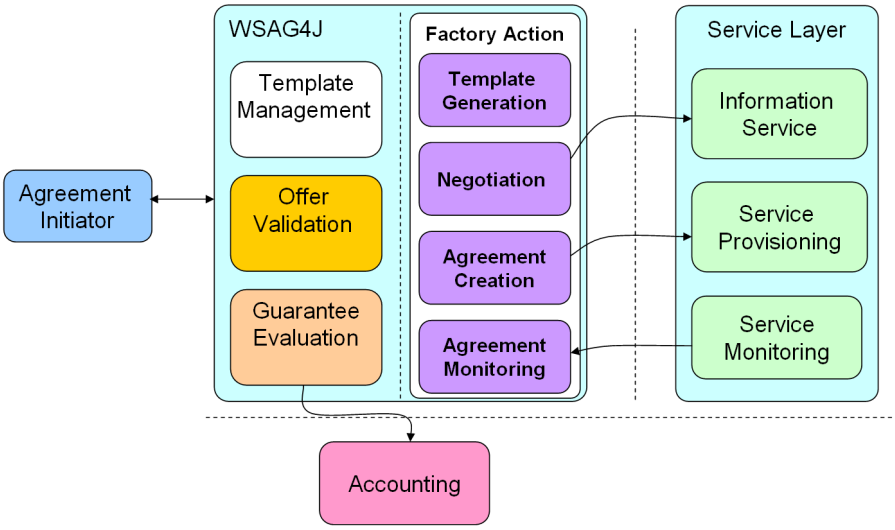

Figure 1: WSAG4J components

As already mentioned, WSAG4J emphasizes the definition of SLA templates and utilizes the information provided in the templates to validate agreement offer compliance at SLA creation time and to monitor agreement compliance during SLA execution time.

Agreement template design

WSAG4J supports developers at designing SLA templates by allowing them to publish templates in form of XML files rather than implementing them via an API. Furthermore, WSAG4J enables developers to dynamically generate SLA templates by using a macro language within the template definition files.

Since the developer defines the template as an XML document, the template files can be edited easily, and the developer does not need to implement agreement templates on API level but can put his focus on the design of the template rather than on the technical details and implementation details. Therefore, the level of entry for designing and implementing WS-Agreement based services is drastically lowered.

Agreement offer validation

WS-Agreement supports the definition of so called creation constraints as part of agreement templates. These creations constrains define how a valid agreement offer based on a specific template is constructed. On the one hand this comprises the structure of the agreement offer, e.g. the service description terms, the structure of a JDSL document contained in a service description term, and the elements that must be contained in this document. On the other hand, creation constraints can define concrete values that a specific element in an agreement offer may take, e.g. the minimum and maximum CPU speed of a computing system.

In case an agreement offer is created based on a template that defines a set of creation constraints, these constraints are automatically evaluated by the WSAG4J engine before the offer is passed to the associated agreement creation action. In that way, only agreements based on valid offers are

created by a WSAG4J service.

Agreement guarantee evaluation

WSAG4J also provides automatic ways for guarantee evaluation. Similar to the validation of creation constraints, the definition of guarantee terms serves as a foundation for the guarantee evaluation. WSAG4J proposes certain best practices that must be followed, in order to evaluate guarantee terms successfully.

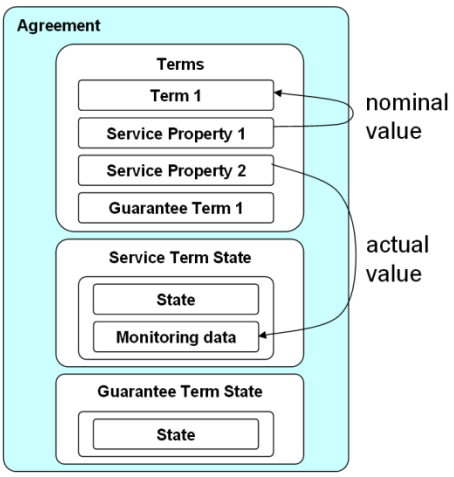

Figure 2: WS-Agreement guarantee evaluation.

The basic principle for defining guarantee terms that can be automatically validated is to define guarantees only over measurable properties of an SLA. WS-Agreement supports the definition of properties via the ServiceProperties language element. A service property can be seen as similar to a variable in a programming language. It binds a (property) name to a certain value of the agreement properties. This is done by referring to this value via an XPath expression.

The agreement properties comprise static data like agreement name, context and service terms that were part of the agreement offer, and dynamic data such as the agreement state and the state of the different service and guarantee terms. Service term states usually comprise monitoring data that is periodically updated. In contrast to that, guarantee term states are automatically derived from service term states by the WSAG4J engine.

In WSAG4J we assume that guarantees define a relationship between a requested service level (nominal value) and a provided service level (actual value). In WS-Agreement, this relationship is expressed in the service level objective (SLO) of a guarantee term. WSAG4J supports the flexible definition of service level objectives and guarantee terms. In order to define a SLO, the requested and provided service levels (nominal and actual values) are exposed as service properties (see Figure 2). The SLO then defines a relationship over these service properties using a simple expression language.

Like creation constraints, guarantee terms are defined at the design time of a SLA template. Once an SLA has been created, the WSAG4J engine periodically evaluates the state of all guarantee terms that are part of this SLA. Furthermore, WSAG4J will account fulfilments and violations of

guarantees.

Conclusion

WSAG4J is a generic framework that support users to easily develop and run WS-Agreement based services. It implements basic mechanisms for creating valid SLAs and managing agreement compliance during runtime. It ships with its own web service stack and provides support for WS-Security and WS-Policy. Furthermore, a simple, action based programming model enables users to easily deploy new agreement templates with custom agreement creation and monitoring strategies in a WSAG4J server and the usage of custom persistence layers is supported.

7. Interoperation Testing based on the AssessGrid and the VIOLA Implementations

The general sequence of messages to create an agreement between two parties is the following: A user (i.e. agreement initiator) exists at administrative domain A, and an AgreementFactory (i.e. the agreement responder) exists at administrative domain B (see also Figure 3). The AgreementFactory publishes a set of agreement templates via its WSRF Resource Properties. The agreement initiator may now query the AgreementFactory for its agreement templates, which in turn returns the published templates to the agreement initiator. The agreement initiator selects the template most suitable to its needs. Based on the selected template, a new agreement offer is created. This new offer is then adapted to the requirements of the agreement initiator. The agreement initiator may change the service descriptions according to its needs, taking into account the constraints posed by the agreement responder. The agreement responder may for example specify a default value for the total available resources for a computation within an agreement template. When the agreement initiator creates an offer based on the template, it can now adjust the total requested resources according to its needs. The created offer is then sent from the agreement initiator to the responder, indicating that a new agreement should be created. The agreement responder can now choose whether or not to create the agreement. In case the responder chooses to accept an offer, it creates a new agreement instance and sends the endpoint reference (EPR) of the agreement instance to the agreement initiator. The agreement initiator can now query the agreement instance at the given EPR for its resource properties, using WSRF's GetResourceProperties() method.

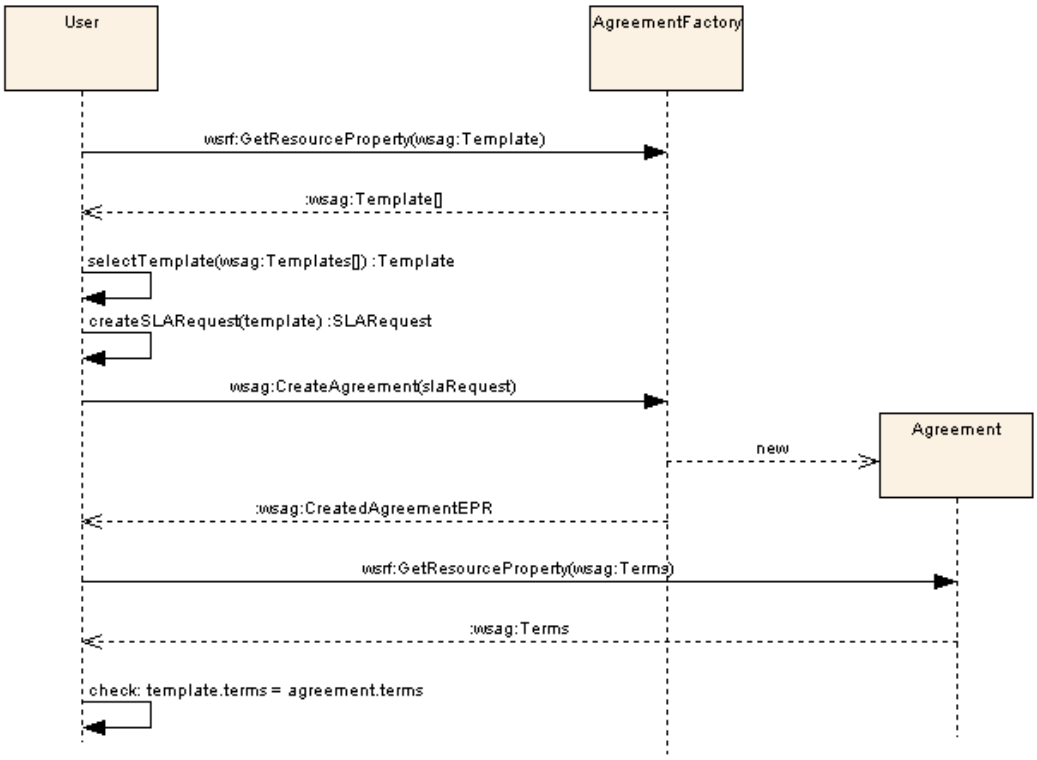

Figure 3: WS-Agreement sequence of messages to create an agreement.

In order to test interoperability of independent grid scheduler implementations using the WS-Agreement protocol, an interoperability scenario was set up featuring the AssessGrid Negotiation Manager (*NegMgr*) and the VIOLA MetaScheduling Service (*MSS*). The goal of this interoperation test was to submit a computational job using the MSS client (based on Apache Muse 2.2.0) to the NegMgr AgreementFactory (based on Globus 4.0). The interoperation scenario realised interoperability on the protocol level to test WS-Agreement and its associated protocols. Please note that the interoperability tests carried out did not deal with any issues related to security.

In the following the interoperation scenario is described in detail (please refer also to Figure 3):

The MSS discovers the NegMgr's AgreementFactory endpoint (based on the configuration).
The MSS queries the NegMgr's AgreementFactory for available templates.
The MSS selects the template for submitting compute jobs.
The MSS creates an AgreementOffer based on the selected template.
The MSS calls the createAgreement() operation on NegMgr's AgreementFactory.
The NegMgr submits the job and creates a new Agreement service instance.
The NegMgr returns the EPR for created Agreement service instance.
The MSS monitors the agreement's state and the ServiceTerms state.

The interoperation tests performed in the GRAAP working group focus on the execution of compute jobs between the AssessGrid Negotiation Manager and the VIOLA MetaScheduling Service. Both implementations use different

software stacks in order to implement the WS-Agreement protocol. The VIOLA MSS implementation is based on the Apache Muse 2.2.0 software stack, while the NegMgr is based on the Globus 4.0 software stack. The different software stacks used to implement the WS-Agreement protocol layer implement different versions of the WSRF protocol. The Globus 4.0 framework implements a draft version of the WSRF protocol (WSRF-1.2 draft), while the Apache Muse 2.2.0 framework implements the final version of the specification (WSRF 1.2 final). Therefore, an interoperability layer was required in order to enable the MSS client to create agreements with the NegMgr agreement factory. This interoperability layer was realized in form of an interoperability proxy, which was responsible for achieving interoperability on protocol level. The interoperability proxy uses XSLT to transform the WSRF messages produced by the MSS client into WSRF messages that are understood by the AssessGrid NegMgr. Figure 4 shows the communication flow described below using the interoperability proxy.

The MSS client sends a request to the interoperability proxy (1). The

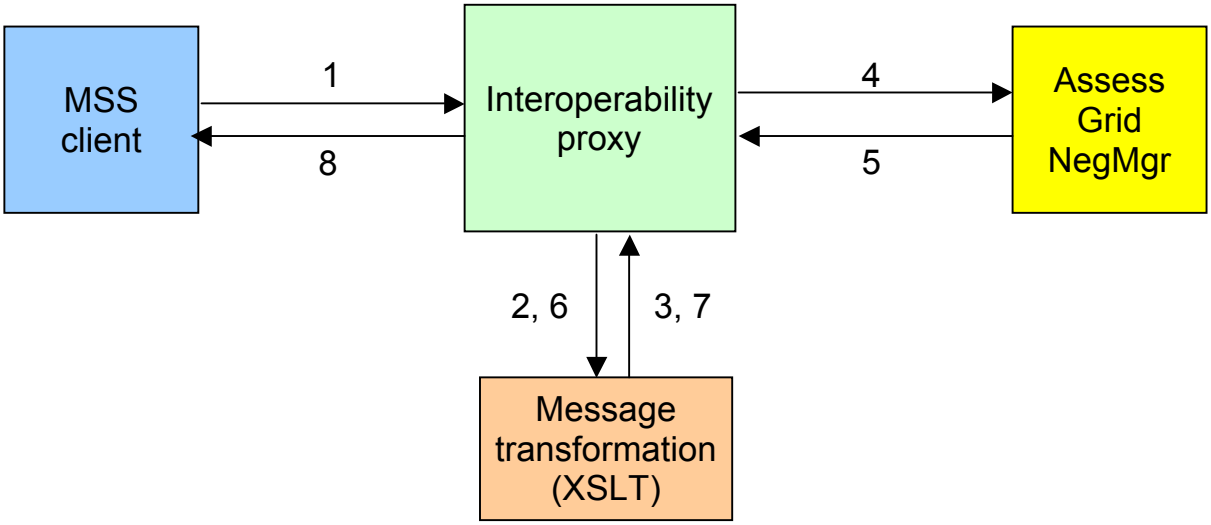

Figure 4: Communication flow using the interoperability proxy

interoperability proxy applies a set of XSL style sheets on the message in order to transform the 'WSRF 1.2 final' specific parts of the message into the 'WSRF 1.2 draft' specific form (2,3). The transformed message is then forwarded to the AssessGrid NegMgr (4) and the message is processed. The NegMgr then sends back a response message to the interoperability proxy (5). The interoperability proxy again applies a set of XSL style sheets on the message in order to transform the 'WSRF 1.2 draft' specific parts of the message into the 'WSRF 1.2 final' specific form (6,7). The transformed response is then returned to the MSS client (8).

This interoperability tests illustrate, how interoperability on message level can be achieved. Only the WSRF related parts of the messages have to be transformed. The WS-Agreement specific parts did not require any transformation.

8. Conclusions

As presented in this document, WS-Agreement has increasingly been used as

technology for creating Service Level Agreements in a variety of projects and environments. Since the information about WS-Agreement implementations was gathered at the end of 2008, this number has increased since then; e.g. a couple of new European projects have begun that build upon WS-Agreement when implementing their infrastructure for SLA management and monitoring.

The investigation regarding WS-Agreement usage and experiences from implementing and using WS-Agreement shows that there are two aspects that need to be considered for interoperability. While no problems have been reported on the protocol level and the language elements of WS-Agreement, it turned out that

- the use of different hosting environments and
- the use of different term languages for service description terms and guarantee terms

may lead to incompatibility issues.

As presented in this document (see section 7), a feasible solution to the problems arising from different hosting environment is the deployment of a proxy, which performs the necessary transformations of the XML documents rendered differently by the hosting environments used.

For the issues resulting from using different term languages, the GRAAP-WG is already working on a number of profiles for WS-Agreement covering the most common use-cases. These will be published in separate documents.

Another issue that was reported over the last two years since the WS-Agreement specification was published as GFD.107 is the limited negotiation capability of WS-Agreement. There are a number of use-cases where negotiation is required either when creating the agreement or during the life-time of an agreement. As a consequence, the group has started discussion on negotiation and re-negotiation more than a year ago and reached an agreement on how to add (re-)negotiation without breaking the compatibility with WS-Agreement version 1.0. At the time of creating this experience document, the discussions have converged already towards on solution and even a first implementation of negotiation has become available. The working group is currently in the process of creating a GFD with the necessary extensions to WS-Agreement 1.0.

9. Contributors

Dominic Battré (Editor)
TU Berlin
Email: dominic.battre@tu-berlin.de

Thomas Fahringer
Innsbruck University
Email: tf@dps.uibk.ac.at

Bastian Koller
HLRS
Email: koller@hlrs.de

David Mobach
Vrije Universiteit
Email: dga.mobach@few.vu.nl

Omer Rana
Cardiff University
Email: o.f.rana@cs.cardiff.ac.uk

Igor Rosenberg
Atos Origin
Email: igor.rosenberg@atosresearch.eu

Johan Tordsson
Umeå University
Email: tordsson@cs.umu.se

Oliver Wäldrich
Fraunhofer SCAI
Email: oliver.waeldrich@scai.fraunhofer.de

Philipp Wieder (Editor)
Dortmund University of Technology
Email: philipp.wieder@udo.edu

Wolfgang Ziegler (Editor)
Fraunhofer SCAI
Email: wolfgang.ziegler@scai.fraunhofer.de

10. Glossary

ALN	Application Layer Networks, integrating different Internet overlay network approaches, like Grid and P2P systems.
BE	Business Experiment, sub-projects of the BEinGRID project
DC	Domain Coordinator, a mediator in AgentScape representing multiple autonomous hosts and communicating with the mobile agent on behalf of these nodes
EPR	End Point Reference
GT4	Globus Toolkit Version 4 Grid middleware
IMRT	Intensity-Modulated Radiation Therapy, a recent radiation technique allowing complex radiation schemes, especially applied near sensitive healthy organs.
JSDL	Job Submission Description Language, OGF specification. JSDL is used to describe the requirements of computational jobs for submission to resources
JXTA	Technology to create peer-to-peer (P2P) applications based on Java technology.
MSS	MetaScheduling Service, a Grid-level Scheduler for arbitrary types of resources with native support for WS-Agreement
OpenCCS	Computing Center Software. Planning and topology based resource management for networked high-performance computers.
QoS	Quality of Service, described the quality of the requested or delivered service.
RMS	Resource Management Systems, managing local (most often computational) resources
SDT	Service Description Term, describe the essential characteristics of a service provided or requested.
SLA	Service Level Agreement, a binding agreement between service provider and service consumer about the QoS delivered and consumed. The degree of bindingness might differ depending on domain and purpose.
SLO	Service Level Objective, expresses in WS-Agreement the relationship defined in the agreement guarantees between a requested service level (nominal value) and a provided service level (actual value).
SOA	Service Oriented Architecture, provides a set of principles of governing concepts used during phases of systems development and integration. Such an architecture will package functionality as interoperable services: functions provided as a service are available to be used from systems

created by other organizations.

WSDL Web Service Description Language, a description language for Web Services for the exchange of XML-based messages.

WSRF Web Service Resource Framewor, a generic and open framework for modeling and accessing stateful resources using Web services.

XSLT Extensible Stylesheet Language (XSL) Transformations (XSLT), a declarative XML-based language used for the transformation of XML documents into other XML documents.

11. Intellectual Property Statement

The OGF 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 OGF Secretariat.

The OGF 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 OGF Executive Director.

12. Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

13. Full Copyright Notice

14. References

Note that only permanent documents are cited as references. Other items, such as Web pages or working groups, are cited inline (i.e., see the Open Grid Forum, http://www.ogf.org).

[RHJ08] Rosenberg, I., Heek, R., and Juan, A.: An SLA Framework for the GT4 Grid Middleware, Collaboration and the Knowledge Economy: Issues, Applications, Case Studies (eChallenges 2008), 2008.

[SOZ+07] Seidel, J., Wäldrich, O., Ziegler, W., Wieder, P., and Yahyapour, R.: Using SLA for resource management and scheduling – a survey, CoreGRID Technical Report TR-0096.

[BRADNER1] Bradner, S.: Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. March 1997.

[GFD.107] Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, Ming Xu: Web Service Agreement (WS-Agreement). GFD.107 proposed recommendation, available at <http://www.ogf.org/documents/GFD.107.pdf>.

[PBM08] Parkin, M. Badia, R. M., Martrat, J.: A Comparison of SLA Use in Six of the European Commissions FP6 Projects, CoreGRID Technical Report TR-0129.

15. Appendix A – Example SLA Templates & Agreements

Generic Agreement Template with SDTs using JSDL and guarantees with assessment interval definitions

```
<wsag:Template xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-
agreement">
  <wsag:Context>
    <wsag:AgreementInitiator/>
    <wsag:AgreementResponder/>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <wsag:ExpirationTime>2009-04-
08T20:24:15.408+02:00</wsag:ExpirationTime>
    <wsag:TemplateId>1</wsag:TemplateId>

<wsag:TemplateName>GuaranteeEvaluationTemplate</wsag:TemplateName>
  </wsag:Context>
  <wsag:Terms>
    <wsag:All>
      <wsag:ServiceDescriptionTerm wsag:Name="Term1"
wsag:ServiceName="Service1">
        <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
          <jsdl:JobDescription>
            <jsdl:Application>
              <jsdl:ApplicationName>KillerApp1</jsdl:ApplicationName>
              <jsdl:ApplicationVersion>1.0</jsdl:ApplicationVersion>
              <jsdl:Description>My first Killer
Application</jsdl:Description>
            </jsdl:Application>
            <jsdl:Resources>
              <jsdl:IndividualCPUSpeed>
                <jsdl:Exact>2.0E9</jsdl:Exact>
              </jsdl:IndividualCPUSpeed>
              <jsdl:IndividualCPUCount>
                <jsdl:Exact>2.0</jsdl:Exact>
              </jsdl:IndividualCPUCount>
              <jsdl:TotalResourceCount>
                <jsdl:Exact>16.0</jsdl:Exact>
              </jsdl:TotalResourceCount>
            </jsdl:Resources>
          </jsdl:JobDescription>
        </jsdl:JobDefinition>
      </wsag:ServiceDescriptionTerm>
      <wsag:GuaranteeTerm wsag:Name="CPU_SPEED_GUARANTEE">
        <wsag:ServiceScope wsag:ServiceName="Service1"/>
        <wsag:ServiceLevelObjective>
          <wsag:KPITarget>
            <wsag:KPIName>CPU SPEED</wsag:KPIName>
            <wsag:CustomServiceLevel>REQ_CPU_SPEED &lt;=
ACT_CPU_SPEED</wsag:CustomServiceLevel>
          </wsag:KPITarget>
        </wsag:ServiceLevelObjective>
        <wsag:BusinessValueList>
          <wsag:Penalty>
            <wsag:AssessmentInterval>
              <wsag:TimeInterval>P5M</wsag:TimeInterval>
            </wsag:AssessmentInterval>
            <wsag:ValueUnit>EUR</wsag:ValueUnit>
            <wsag:ValueExpression>5</wsag:ValueExpression>
          </wsag:Penalty>
```

```
          <wsag:Reward>
            <wsag:AssessmentInterval>
              <wsag:TimeInterval>P5M</wsag:TimeInterval>
            </wsag:AssessmentInterval>
            <wsag:ValueUnit>EUR</wsag:ValueUnit>
            <wsag:ValueExpression>10</wsag:ValueExpression>
          </wsag:Reward>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
      <wsag:ServiceProperties wsag:Name="Service_Properties_1"
wsag:ServiceName="Service1">
        <wsag:VariableSet>
          <wsag:Variable wsag:Name="REQ_CPU_SPEED"
wsag:Metric="xsd:integer">
            <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsa
g:Name =
'Term1']/jsdl:JobDefinition/jsdl:JobDescription/jsdl:Resources/jsdl:I
ndividualCPUSpeed/jsdl:Exact</wsag:Location>
          </wsag:Variable>
          <wsag:Variable wsag:Name="ACT_CPU_SPEED"
wsag:Metric="xsd:integer">
            <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
jsdl-posix='http://schemas.ggf.org/jsdl/2005/11/jsdl-posix';declare
namespace wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:ServiceTermState[@wsag:termName='Term1']/jsdl:J
obDefinition/jsdl:JobDescription/jsdl:Resources/jsdl:IndividualCPUSpe
ed/jsdl:Exact</wsag:Location>
          </wsag:Variable>
        </wsag:VariableSet>
      </wsag:ServiceProperties>
    </wsag:All>
  </wsag:Terms>
</wsag:Template>
```

Agreement with guarantees, penalties, agreement and term states (resulting from template above)

```
<wsag:AgreementProperties
xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement">
  <wsag:Context>
    <wsag:AgreementInitiator/>
    <wsag:AgreementResponder/>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <wsag:ExpirationTime>2009-04-
08T20:24:15.408+02:00</wsag:ExpirationTime>
    <wsag:TemplateId>1</wsag:TemplateId>

<wsag:TemplateName>GuaranteeEvaluationTemplate</wsag:TemplateName>
  </wsag:Context>
  <wsag:Terms>
    <wsag:All>
      <wsag:ServiceDescriptionTerm wsag:Name="Term1"
wsag:ServiceName="Service1">
        <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
          <jsdl:JobDescription>
            <jsdl:Application>
```

```
              <jsdl:ApplicationName>KillerApp1</jsdl:ApplicationName>
              <jsdl:ApplicationVersion>1.0</jsdl:ApplicationVersion>
              <jsdl:Description>My first Killer
Application</jsdl:Description>
            </jsdl:Application>
            <jsdl:Resources>
              <jsdl:IndividualCPUSpeed>
                <jsdl:Exact>2.0E9</jsdl:Exact>
              </jsdl:IndividualCPUSpeed>
              <jsdl:IndividualCPUCount>
                <jsdl:Exact>2.0</jsdl:Exact>
              </jsdl:IndividualCPUCount>
              <jsdl:TotalResourceCount>
                <jsdl:Exact>16.0</jsdl:Exact>
              </jsdl:TotalResourceCount>
            </jsdl:Resources>
          </jsdl:JobDescription>
        </jsdl:JobDefinition>
      </wsag:ServiceDescriptionTerm>
      <wsag:GuaranteeTerm wsag:Name="CPU_SPEED_GUARANTEE">
        <wsag:ServiceScope wsag:ServiceName="Service1"/>
        <wsag:ServiceLevelObjective>
          <wsag:KPITarget>
            <wsag:KPIName>CPU SPEED</wsag:KPIName>
            <wsag:CustomServiceLevel>REQ_CPU_SPEED &lt;=
ACT_CPU_SPEED</wsag:CustomServiceLevel>
          </wsag:KPITarget>
        </wsag:ServiceLevelObjective>
        <wsag:BusinessValueList>
          <wsag:Penalty>
            <wsag:AssessmentInterval>
              <wsag:TimeInterval>P5M</wsag:TimeInterval>
            </wsag:AssessmentInterval>
            <wsag:ValueUnit>EUR</wsag:ValueUnit>
            <wsag:ValueExpression>5</wsag:ValueExpression>
          </wsag:Penalty>
          <wsag:Reward>
            <wsag:AssessmentInterval>
              <wsag:TimeInterval>P5M</wsag:TimeInterval>
            </wsag:AssessmentInterval>
            <wsag:ValueUnit>EUR</wsag:ValueUnit>
            <wsag:ValueExpression>10</wsag:ValueExpression>
          </wsag:Reward>
        </wsag:BusinessValueList>
      </wsag:GuaranteeTerm>
      <wsag:ServiceProperties wsag:Name="Service_Properties_1"
wsag:ServiceName="Service1">
        <wsag:VariableSet>
          <wsag:Variable wsag:Name="REQ_CPU_SPEED"
wsag:Metric="xsd:integer">
            <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl'; declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-agreement';
$this/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name='Ter
m1']/jsdl:JobDefinition/jsdl:JobDescription/jsdl:Resources/jsdl:Indiv
idualCPUSpeed/jsdl:Exact</wsag:Location>
          </wsag:Variable>
          <wsag:Variable wsag:Name="ACT_CPU_SPEED"
wsag:Metric="xsd:integer">
            <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl'; declare namespace
```

```
wsag='http://schemas.ggf.org/graap/2007/03/ws-agreement';
$this/wsag:ServiceTermState[@wsag:termName='Term1']/jsdl:JobDefinitio
n/jsdl:JobDescription/jsdl:Resources/jsdl:IndividualCPUSpeed/jsdl:Exa
ct</wsag:Location>
          </wsag:Variable>
        </wsag:VariableSet>
      </wsag:ServiceProperties>
    </wsag:All>
  </wsag:Terms>
  <wsag:AgreementState>
    <wsag:State>Observed</wsag:State>
  </wsag:AgreementState>
  <wsag:ServiceTermState wsag:termName="Term1">
    <wsag:State>Ready</wsag:State>
    <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
      <jsdl:JobDescription>
        <jsdl:Application>
          <jsdl:ApplicationName>KillerApp1</jsdl:ApplicationName>
          <jsdl:ApplicationVersion>1.0</jsdl:ApplicationVersion>
          <jsdl:Description>My first Killer
Application</jsdl:Description>
        </jsdl:Application>
        <jsdl:Resources>
          <jsdl:IndividualCPUSpeed>
            <jsdl:Exact>2.1E9</jsdl:Exact>
          </jsdl:IndividualCPUSpeed>
          <jsdl:IndividualCPUCount>
            <jsdl:Exact>2.0</jsdl:Exact>
          </jsdl:IndividualCPUCount>
          <jsdl:TotalResourceCount>
            <jsdl:Exact>16.0</jsdl:Exact>
          </jsdl:TotalResourceCount>
        </jsdl:Resources>
      </jsdl:JobDescription>
    </jsdl:JobDefinition>
  </wsag:ServiceTermState>
</wsag:AgreementProperties>
```

Agreement Template with Creation Constraints (UNICORE integration)

```
<wsag:Template wsag:TemplateId="1"
xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement">
  <wsag:Name>COMPUTE-JOB</wsag:Name>
  <wsag:Context>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <wsag:TemplateId>1</wsag:TemplateId>
    <wsag:TemplateName>COMPUTE-JOB</wsag:TemplateName>
    <eng:WSAG4JSession
xmlns:eng="http://schemas.scai.fraunhofer.de/2008/11/wsag4j/engine">
      <eng:SessionID>12115dfe963-419678bc139598cf</eng:SessionID>
    </eng:WSAG4JSession>
  </wsag:Context>
  <wsag:Terms>
    <wsag:All>
      <wsag:ExactlyOne>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
```

```
                <jsdl:ApplicationName>WISDOM-
PACK</jsdl:ApplicationName>

<jsdl:ApplicationVersion>1.0</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>WISDOM-
UNPACK</jsdl:ApplicationName>

<jsdl:ApplicationVersion>1.0</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>WISDOM-
MKDIR</jsdl:ApplicationName>

<jsdl:ApplicationVersion>1.0</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>FDS</jsdl:ApplicationName>

<jsdl:ApplicationVersion>4.x</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>FlexX</jsdl:ApplicationName>

<jsdl:ApplicationVersion>2.0</jsdl:ApplicationVersion>

<jsdl:Description>DEFAUL_RUNTIME=5500</jsdl:Description>
              </jsdl:Application>
```

```
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>Date</jsdl:ApplicationName>

<jsdl:ApplicationVersion>1.0</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>POVRay</jsdl:ApplicationName>

<jsdl:ApplicationVersion>3.5</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>Bash
shell</jsdl:ApplicationName>

<jsdl:ApplicationVersion>3.1.16</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>C shell</jsdl:ApplicationName>

<jsdl:ApplicationVersion>6.14.00</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
```

```
              <jsdl:Application>
                <jsdl:ApplicationName>Korn
shell</jsdl:ApplicationName>
                <jsdl:ApplicationVersion>Version M 1993-12-28
q</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>Perl</jsdl:ApplicationName>

<jsdl:ApplicationVersion>5.8.8</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
        <wsag:ServiceDescriptionTerm wsag:Name="APPLICATION_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Application>
                <jsdl:ApplicationName>Python
Script</jsdl:ApplicationName>

<jsdl:ApplicationVersion>2.4.2</jsdl:ApplicationVersion>
              </jsdl:Application>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
      </wsag:ExactlyOne>
      <wsag:ExactlyOne>
        <wsag:ServiceDescriptionTerm wsag:Name="RESOURCE_STD"
wsag:ServiceName="UNICORE6">
          <jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
            <jsdl:JobDescription>
              <jsdl:Resources>
                <jsdl:IndividualCPUTime>
                  <jsdl:Exact>3600.0</jsdl:Exact>
                </jsdl:IndividualCPUTime>
                <jsdl:IndividualCPUCount>
                  <jsdl:Exact>2.0</jsdl:Exact>
                </jsdl:IndividualCPUCount>
                <jsdl:TotalResourceCount>
                  <jsdl:Exact>1.0</jsdl:Exact>
                </jsdl:TotalResourceCount>
              </jsdl:Resources>
            </jsdl:JobDescription>
          </jsdl:JobDefinition>
        </wsag:ServiceDescriptionTerm>
      </wsag:ExactlyOne>
    </wsag:All>
  </wsag:Terms>
  <wsag:CreationConstraints>
```

```
    <wsag:Item wsag:Name="JobDescriptionTypeConstraint">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="JobIdentification"
type="jsdl:JobIdentification_Type" minOccurs="0" maxOccurs="0"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Application" type="jsdl:Application_Type"
minOccurs="0" maxOccurs="0"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Resources" type="jsdl:Resources_Type"
minOccurs="1" maxOccurs="1"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="DataStaging" type="jsdl:DataStaging_Type"
minOccurs="0" maxOccurs="0"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item wsag:Name="JobDefinition_ALL">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm/jsdl:JobDefinition</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="JobDescription" minOccurs="1"
maxOccurs="1" type="jsdl:JobDescription_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription@Resources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="JobIdentification" minOccurs="0"
maxOccurs="0" type="jsdl:JobIdentification_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Application" minOccurs="0" maxOccurs="0"
type="jsdl:Application_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Resources" minOccurs="1" maxOccurs="1"
type="jsdl:Resources_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="DataStaging" minOccurs="0" maxOccurs="0"
type="jsdl:DataStaging_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
```

```
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Resources@Resources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription/jsdl:Resources</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="IndividualCPUTime" minOccurs="1"
maxOccurs="1" type="jsdl:RangeValue_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="IndividualCPUCount" minOccurs="1"
maxOccurs="1" type="jsdl:RangeValue_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="TotalResourceCount" minOccurs="1"
maxOccurs="1" type="jsdl:RangeValue_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Resources_IndividualCPUTime@R
esources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription/jsdl:Resources/jsdl:IndividualCPUTime</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="UpperBoundedRange" minOccurs="0"
maxOccurs="0" type="jsdl:Boundary_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="LowerBoundedRange" minOccurs="0"
maxOccurs="0" type="jsdl:Boundary_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Exact" minOccurs="0" maxOccurs="1"
type="jsdl:Exact_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Range" minOccurs="0" maxOccurs="0"
type="jsdl:Range_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Resources_IndividualCPUTime_E
xact@Resources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription/jsdl:Resources/jsdl:IndividualCPUTime/jsdl:Exact</wsag:L
ocation>
      <wsag:ItemConstraint>
```

```
        <xs:minInclusive value="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        <xs:maxInclusive value="86400.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Resources_IndividualCPUCount@
Resources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription/jsdl:Resources/jsdl:IndividualCPUCount</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="UpperBoundedRange" minOccurs="0"
maxOccurs="0" type="jsdl:Boundary_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="LowerBoundedRange" minOccurs="0"
maxOccurs="0" type="jsdl:Boundary_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Exact" minOccurs="0" maxOccurs="1"
type="jsdl:Exact_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Range" minOccurs="0" maxOccurs="0"
type="jsdl:Range_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Resources_IndividualCPUCount_
Exact@Resources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription/jsdl:Resources/jsdl:IndividualCPUCount/jsdl:Exact</wsag:
Location>
      <wsag:ItemConstraint>
        <xs:minInclusive value="2.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        <xs:maxInclusive value="2.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Resources_TotalResourceCount@
Resources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription/jsdl:Resources/jsdl:TotalResourceCount</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="UpperBoundedRange" minOccurs="0"
```

```
maxOccurs="0" type="jsdl:Boundary_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="LowerBoundedRange" minOccurs="0"
maxOccurs="0" type="jsdl:Boundary_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Exact" minOccurs="0" maxOccurs="1"
type="jsdl:Exact_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Range" minOccurs="0" maxOccurs="0"
type="jsdl:Range_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Resources_TotalResourceCount_
Exact@Resources_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='RESOURCE_STD']/jsdl:JobDefinition/jsdl:Jo
bDescription/jsdl:Resources/jsdl:TotalResourceCount/jsdl:Exact</wsag:
Location>
      <wsag:ItemConstraint>
        <xs:minInclusive value="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
        <xs:maxInclusive value="10.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription@Application_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD']/jsdl:JobDefinition/jsdl
:JobDescription</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="JobIdentification" minOccurs="0"
maxOccurs="0" type="jsdl:JobIdentification_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Application" minOccurs="1" maxOccurs="1"
type="jsdl:Application_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="Resources" minOccurs="0" maxOccurs="0"
type="jsdl:Resources_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
          <xs:element name="DataStaging" minOccurs="0" maxOccurs="0"
type="jsdl:DataStaging_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application@Application_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
```

```
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD']/jsdl:JobDefinition/jsdl
:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" minOccurs="0"
maxOccurs="1" type="xs:string"/>
          <xs:element name="ApplicationVersion" minOccurs="0"
maxOccurs="1" type="xs:string"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1"
type="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_1']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="WISDOM-PACK"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="1.0"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_2']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="WISDOM-UNPACK"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="1.0"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
```

```
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_3']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="WISDOM-MKDIR"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="1.0"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_4']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="FDS"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="4.x"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
```

```
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_5']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="FlexX"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="2.0"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl">
                <xs:enumeration value="DEFAUL_RUNTIME=5500"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_6']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="Date"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="1.0"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
```

```
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_7']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="POVRay"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="3.5"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_8']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="Bash shell"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="3.1.16"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_9']/jsdl:JobDefinition/js
dl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```
          <xs:element name="ApplicationName" type="xs:string"
fixed="C shell"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="6.14.00"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_10']/jsdl:JobDefinition/j
sdl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="Korn shell"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="Version M 1993-12-28 q"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_11']/jsdl:JobDefinition/j
sdl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="Perl"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="5.8.8"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
```

```
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item
wsag:Name="JobDefinition_JobDescription_Application_ApplicationName@A
pplication_SDT">
      <wsag:Location>declare namespace
jsdl='http://schemas.ggf.org/jsdl/2005/11/jsdl';declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All/wsag:Service
DescriptionTerm[@wsag:Name='APPLICATION_STD_12']/jsdl:JobDefinition/j
sdl:JobDescription/jsdl:Application</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="ApplicationName" type="xs:string"
fixed="Python Script"/>
          <xs:element name="ApplicationVersion" type="xs:string"
fixed="2.4.2"/>
          <xs:element name="Description" minOccurs="0" maxOccurs="1">
            <xs:simpleType>
              <xs:restriction base="jsdl:Description_Type"
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"/>
            </xs:simpleType>
          </xs:element>
          <xs:any
namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
processContents="strict" minOccurs="0"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item wsag:Name="AgreementOffer">
      <wsag:Location>declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element ref="wsag:Name" minOccurs="0" maxOccurs="0"/>
          <xs:element ref="wsag:Context" minOccurs="1"
maxOccurs="1"/>
          <xs:element ref="wsag:Terms" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item wsag:Name="AgreementOffer_Context">
      <wsag:Location>declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Context</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="AgreementInitiator" type="xs:anyType"
minOccurs="0" maxOccurs="0"/>
          <xs:element name="AgreementResponder" type="xs:anyType"
minOccurs="0" maxOccurs="0"/>
          <xs:element name="ServiceProvider"
type="wsag:AgreementRoleType" minOccurs="1" maxOccurs="1"/>
```

```
          <xs:element name="ExpirationTime" type="xs:dateTime"
minOccurs="0" maxOccurs="0"/>
          <xs:element name="TemplateId" type="xs:string"
minOccurs="1" maxOccurs="1"/>
          <xs:element name="TemplateName" type="xs:string"
minOccurs="1" maxOccurs="1"/>
          <xs:any
namespace="http://schemas.scai.fraunhofer.de/2008/11/wsag4j/engine"
processContents="strict"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item wsag:Name="AgreementOffer_Context_ServiceProvider">
      <wsag:Location>declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Context/wsag:ServiceProvide
r</wsag:Location>
      <wsag:ItemConstraint>
        <xs:simpleType xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:restriction base="wsag:AgreementRoleType">
            <xs:enumeration value="AgreementResponder"/>
          </xs:restriction>
        </xs:simpleType>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item wsag:Name="AgreementOffer_Terms">
      <wsag:Location>declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms</wsag:Location>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:element name="All" minOccurs="1" maxOccurs="1"
type="wsag:TermCompositorType"/>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
    <wsag:Item wsag:Name="AgreementOffer_Terms_All">
      <wsag:Location>declare namespace
wsag='http://schemas.ggf.org/graap/2007/03/ws-
agreement';$this/wsag:AgreementOffer/wsag:Terms/wsag:All</wsag:Locati
on>
      <wsag:ItemConstraint>
        <xs:sequence xmlns:xs="http://www.w3.org/2001/XMLSchema">
          <xs:choice maxOccurs="2">
            <xs:element name="ExactlyOne" minOccurs="0" maxOccurs="0"
type="wsag:TermCompositorType"/>
            <xs:element name="OneOrMore" minOccurs="0" maxOccurs="0"
type="wsag:TermCompositorType"/>
            <xs:element ref="wsag:All" minOccurs="0" maxOccurs="0"/>
            <xs:element name="ServiceDescriptionTerm" minOccurs="1"
maxOccurs="1" type="wsag:ServiceDescriptionTermType"/>
            <xs:element name="ServiceReference" minOccurs="0"
maxOccurs="0" type="wsag:ServiceReferenceType"/>
            <xs:element name="ServiceProperties" minOccurs="0"
maxOccurs="0" type="wsag:ServicePropertiesType"/>
            <xs:element name="GuaranteeTerm" minOccurs="0"
maxOccurs="0" type="wsag:GuaranteeTermType"/>
          </xs:choice>
        </xs:sequence>
      </wsag:ItemConstraint>
    </wsag:Item>
```

```
  </wsag:CreationConstraints>
</wsag:Template>
```