034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number D.1.2

East/West interfaces for NRPS

Due date of deliverable: 2007-10-31
Actual submission date: 17.06.2009
Document code: Phosphorus-WP1-D.1.2

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
SURFnet

Abstract

This document, part of the deliverable named “East/West interfaces for NRPS”, defines the interfaces and methods that have been designed to provide NRPS interoperability by means of the Network Service Plane. This document is a complement for the prototype that implements such interfaces and methods. The prototype consists of a set of software modules for the NRPS that demonstrate the interoperability between UCLP, ARGON and DRAC, through the establishment of end-to-end provisioning services based on the parameters of the less constrain system involved.

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Project: Phosphorus
Deliverable Number: D.1.2
Date of Issue: 17/06/09
EC Contract No.: 034115
Document Code: Phosphorus-WP1-D.1.2

Table of Contents

0 Executive Summary 6

1 System Architecture for NRPS Interoperability 8
- 1.1 Interoperability Layer 8
- 1.2 Web Services 8
- 1.3 Supporting Software Components 9
- 1.4 NRPS Registration 10

2 Software Components 11
- 2.1 NRPS Adapter 11
  - 2.1.1 DRAC Adapter 14
  - 2.1.2 ARGON Adapter 15
  - 2.1.3 UCLP Adapter 16
- 2.2 NRPS Manager 17
- 2.3 Path Computer 18

3 Detailed Formal Description of East/West Interface Operations and Data Types 19
- 3.1 Reservation Service 19
- 3.2 Topology Service 20
- 3.3 Data Types 20
- 3.4 Software Development and Tools 22

4 Conclusions 24

5 References 24

6 Acronyms 25

Table of Figures

Figure 1.1: High level view of the system's architecture with position of east/west interfaces, supporting software components and main communication paths as arrows. 10

Figure 2.2: Schema of the Adapters' implementations 12

Figure 2.3: WS interface at the top of the common part of the NRPS Adapter 12

Figure 2.4: Implementation of the skeleton 13

Figure 2.5: NRPS Adapter architecture and operations 14

Figure 2.6: Behaviour of the UCLP NRPS Adapter 16

Figure 2.7: Subscriber relationship in the Phosphorus WS architecture using the UCLP Adapter 17

0 Executive Summary

This deliverable describes the set of interfaces for the Network Resource Provisioning System (NRPS), which enable interoperability between UCLP, ARGON, DRAC and the Thin NRPS for GMPLS, through the establishment of end-to-end provisioning services based on the parameters of the most constrained system involved.

These interfaces are named east/west interfaces because they are positioned such as to enable interoperability of multiple NRPSs with one superposed coordinating system, the Network Service Plane (NSP). The interfaces enable interoperability because they define both common data types and common operations that the NRPSs much adhere to when they take part in NSP coordinated resource provisioning. For that, both functionality and data types are aligned within a layer between an NRPS and the NSP, while the NSP uses the common interface to communicate with each NRPS. Adapters specifically suited for individual NRPSs are part of the actual implementation of this interoperability layer. These Adapters are designed according to the east/west interfaces' definitions. They implement the common operations by translating them to NRPS specific operations. Thus interoperability is achieved. It's important to note that an individual NRPS may actually support either more or less sophisticated resource provisioning compared to the NSP. In the former case the NRPSs additional functionality must be ignored by the NSP; in the latter, the NRPS will need to implement this NSP functionality before it can be fully integrated.

The integrated individual domains are controlled by the one NSP. The NSP is the controller of the underlying NRPSs that allows their interoperability. Because the individual NRPSs are closed systems and are only able to control their own domain, to enable cross-domain path provisioning there needs to be within the NSP: 1) knowledge about the integrated domains cross connections, and 2) a path provisioning system. For that, the NSP has a topology database that contains knowledge about all integrated domains and the cross connections between them, and a path computer that can calculate multi-domain spanning paths using this topology knowledge. The integrated NRPSs share topology knowledge with the NSP through a topology service that runs within the NSP. Moreover, the architecture of the NSP facilitates interoperation with other projects opening the way to a distributed heterogeneous resource management solution for GRID applications.

End-to-end path provisioning is achieved by coordinated requests for reservation to reservation services that are present in the integrated NRPS Adapters. Each Adapter translates the reservation request to requests understood by its underlying domain controller. The returned reservations add up to one end-to-end resource

reservation spanning multiple domains. Together, the topology and reservation service make up the east/west interfaces.

1 System Architecture for NRPS Interoperability

To illustrate the context in which the east/west interfaces and concrete software components play their role in NRPS interoperability this section describes them and the position they have in the system's architecture.

1.1 Interoperability Layer

Each existing NRPS uses its own specific interface operations and own specific data types. The key to NRPS interoperability is a reference set of interfaces and data types, i.e. the east/west interfaces and common data types and only these are used within the NSP. The specific operations for reservation of resources in end-to-end path provisioning are defined in a reservation interface. The specific operations necessary for updating the NSP with topology information is defined in the topology interface. To enable interoperability there is a layer between the NRPSs and NSP in which each NRPS's specific set of operations and their corresponding specific data types are translated to the operations and data types known within the NSP. For each individual NRPS this operational translation is done by a particular software component called an Adapter.

1.2 Web Services

Web services are the instrument for communication between NRPSs and the NSP. Web services are software implementations of abstract service interface (in our case the east/west interfaces) descriptions and they are defined within WSDL (Web Service Definition Language) description files. A WSDL describes the web service's functionality by means of a specific set of operations and the data type these operations can handle. The data types are separately defined through an XML Schema language within XML Schema files and referenced by the WSDL files. Once the web services are deployed they can be accessed through a web service client which references the same WSDL and XML Schema files that the web service is based upon. Thus the communication channel between client and service is established. Software code for web services can for the greater part be automatically created using the WSDL and XML Schema files by specific software tools.

1.3 Supporting Software Components

Apart from operational translation the NRPS adapter also enables the communication between the NRPS and NSP through an integrated reservation web service and a client for a topology web service that runs within the NSP. Both components are essential for end-to-end resource provisioning (see Section 2.1.1).

Other software components are necessary to coordinate NRPS interoperability for end-to-end resource provisioning.

A database within the NSP contains selected topology information of all integrated domains. Besides a unique name this includes the domains border endpoints, the corresponding inter-domain connections and a TNA prefixes for all user endpoints. Furthermore the address of the reservation web service that runs within the NRPS Adapters is stored. The NSP needs these web services' addresses to be able to invoke the services when requesting a reservation. This NRPS information is obtained by the NSP through an integrated topology web service. This topology service is called upon by each individual NRPS adapter whenever it wants to update information regarding its domain topology or its integrated reservation web service address.

A Path Computer is present within the NSP and it is able to compute a multi-domain spanning path using the topology information from the database. For each new request for end-to-end resource provisioning the Path Computer calculates a new path.

A reservation for end-to-end resource provisioning spans multiple domains. Therefore the reservation for this must be split up into individual requests for reservation for each domain involved. The coordination of this is performed within the NSP's NRPS manager.

Figure 1 shows the position of the above mentioned elements in the overall system architecture.

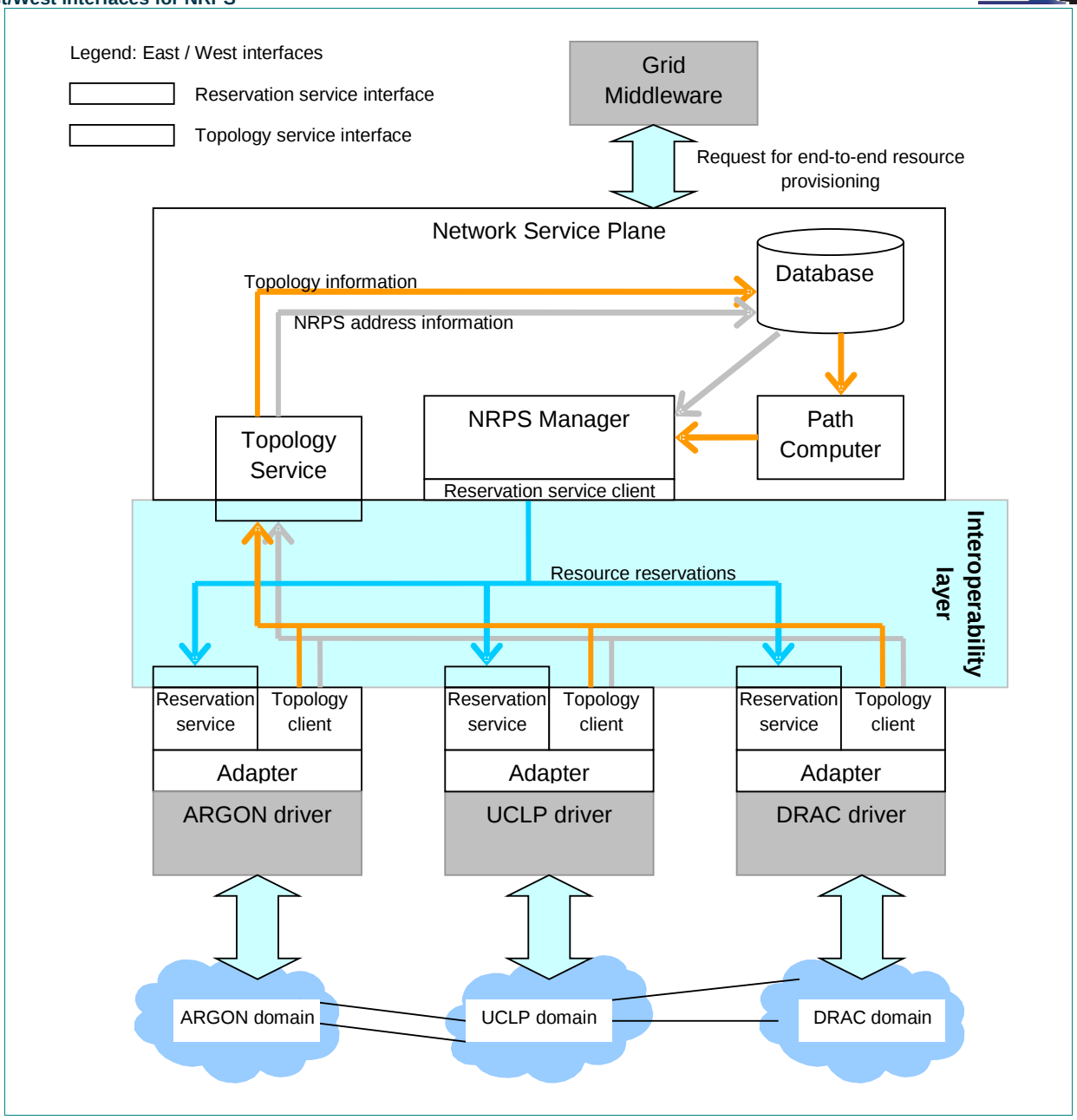

Figure 1.1: High level view of the system's architecture with position of east/west interfaces, supporting software components and main communication paths as arrows.

1.4 NRPS Registration

Apart from implementing the east/west interfaces as web services and the development of supporting software components the participation of NRPSs in end-to-end resource provisioning requires active registration of each NRPS to the NSP. Registering an NRPS domain to the NSP means that the address at which an NRPS

adapter's reservation web service operates (i.e. can be reached) is made known to the NSP. This is done by calling the NSP topology service's addDomain operation. This operation takes an adapter's registration web service endpoint reference (i.e. the web service's address) as a parameter. This endpoint reference is subsequently saved within the NSP's topology database.

2 Software Components

This section describes in detail all software components which enable end-to-end resource provisioning.

2.1 NRPS Adapter

For an individual NRPS to be integrated into the Phosphorus system it needs to be tailored by means of a software component called the NRPS Adapter. This Adapter acts as a wrapper and translates the NSP specific (i.e. common) operations and their according data types to NRPS specific operations and data types and vice versa.

The NRPS Adapter is located at the NRPS side (at the top of each NRPS). It implements all the required operations to invoke the reservation functions of the NRPSs. The Adapter has a common part for all the NRPSs and another specific one for each kind of NRPS (ARGON, DRAC and UCLP). The common part mainly consists of the communication interface for the NSP. Therefore, each kind of NRPS implements its own Adapter to translate the common requests of the reservation WS to the corresponding function of the NRPS.

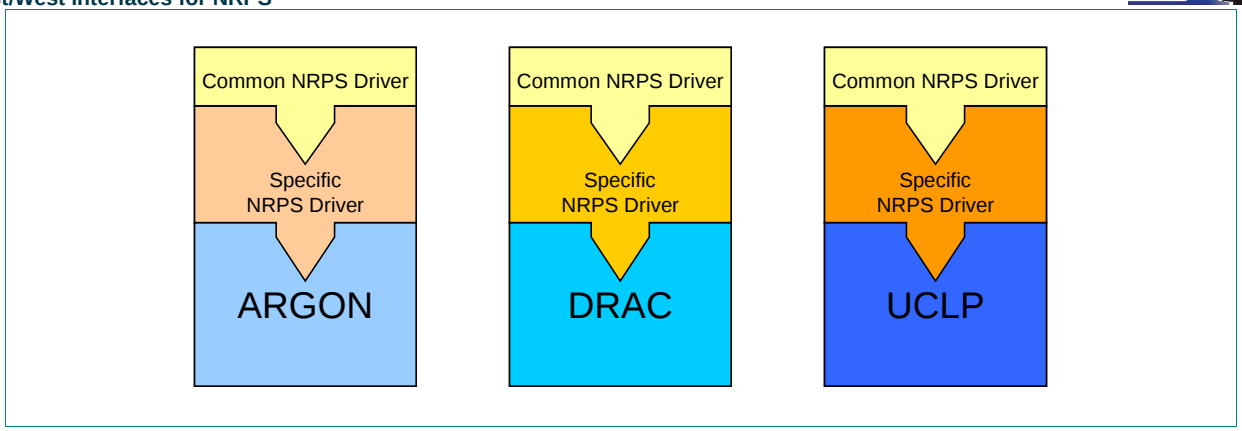

Figure 2.2: Schema of the Adapters' implementations

The way to communicate between the layers, either in one way or the other, is through Web Services. The NRPS Adapter implements an interface based in Web Services that is accessible by the NSP. The NSP, in turn, implements the Web Service to allow the notifications. The NSP will know detailed information about all the controlled NRPSs, including the location of the Web Services in order to be able to invoke their operations.

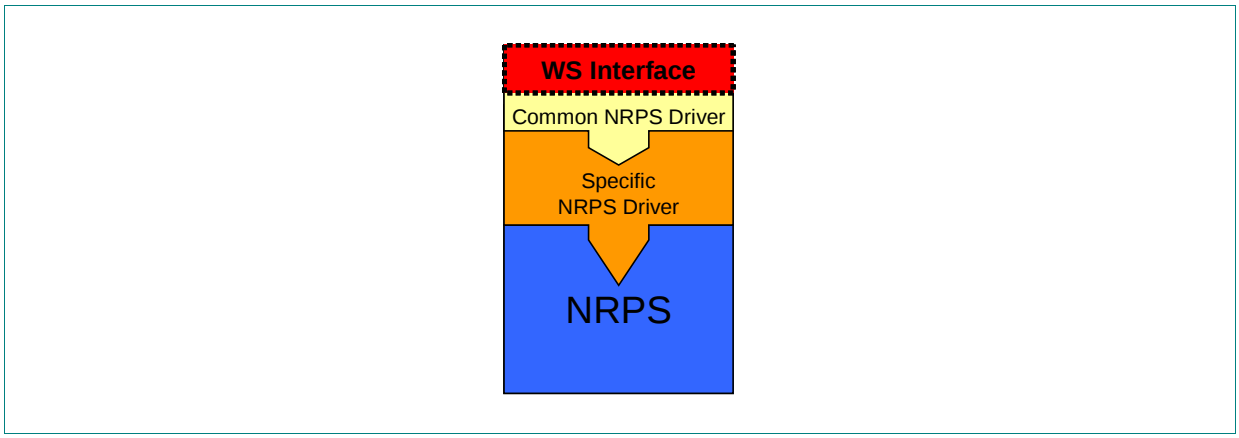

Figure 2.3: WS interface at the top of the common part of the NRPS Adapter

To implement the WS, the MUSE application is used. Once the WSDL that describes the Endpoint is complete, the command-line *wsdl2java* tool can be used to generate all of the code needed to either implement and deploy an Endpoint or create a proxy to an endpoint. This process creates two Java files: the interface and the skeleton, which is completed or extended with the full implementation of the services. To make the implementation and the later modifications easier, the skeleton could create an abstract object NRPS Adapter without specialization, and then call its methods; this way, the skeleton and the abstract NRPS Adapter will remain unchanged for all the NRPSs. Then, each NRPS should implement the specific part extending the abstract class.

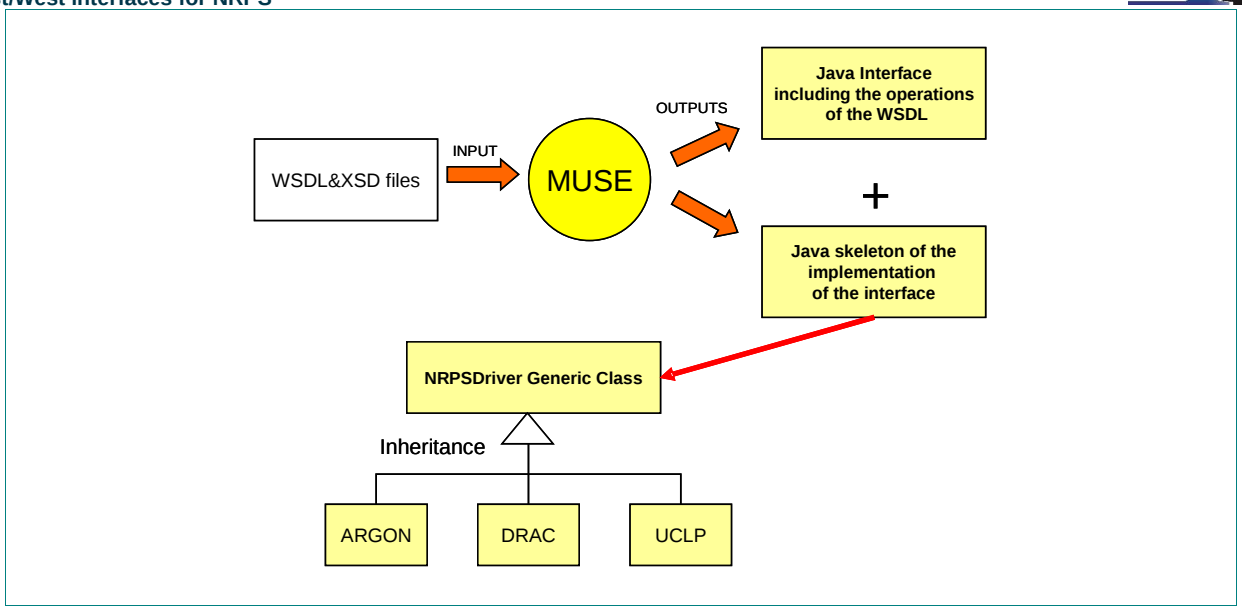

Figure 2.4: Implementation of the skeleton

The operations that the NRPS Adapter offers are defined in deliverable D1.1 of the project. Later, in the implementation phase, support operations can be added to offer other functionalities if needed. These are the main functions that offer the functionalities defined in D1.1:

- Availability request
- Reservation request
- Cancel reservation
- Status request
- Bind request
- Activation request
- Complete Job
- Cancel Job
- Retrieve features
- Retrieve endpoints

The NSP can invoke the operations offered by the WS Interface. These operations are implemented between the common Adapter and the specific one. The common part (the same for all the NRPSs) has the interface and the stubs needed to implement the operations, as well as the notification service. The specific part has a different implementation for each one of the NRPSs and developed by the different teams in charge of the particular NRPS. Hence, each team will have the stubs generated from the WSDLs defined in D1.1 and they will have to implement the services according to the features of each NRPS. The idea is lo leave unmodified the generated stubs and implement the services independently. This will allow simpler modifications in the future.

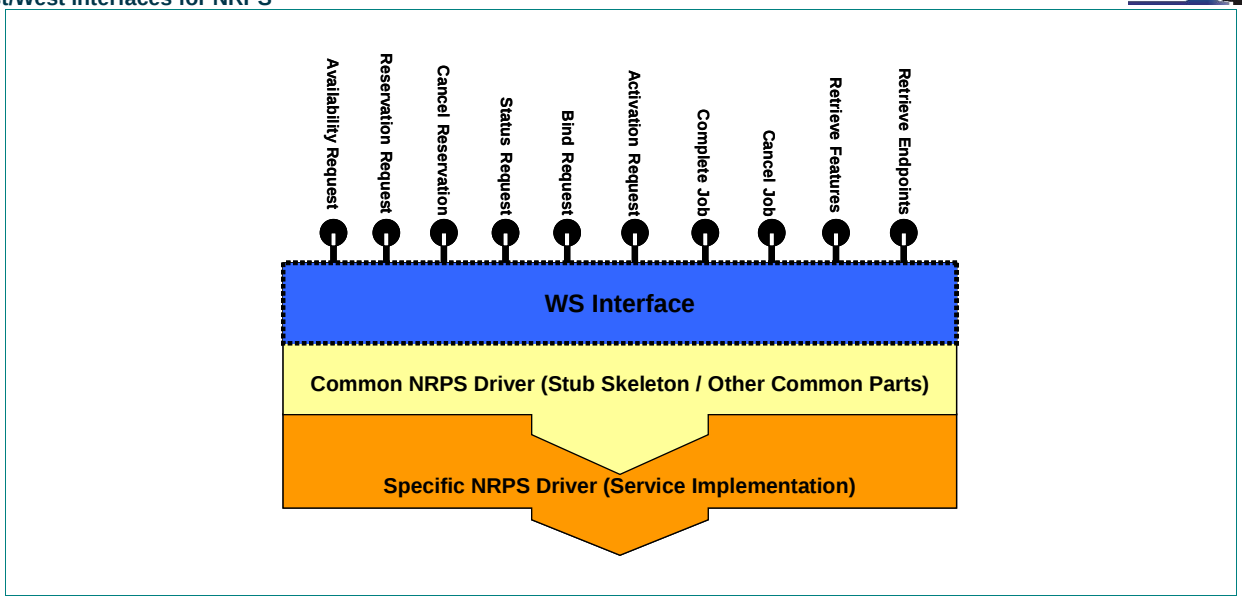

Figure 2.5: NRPS Adapter architecture and operations

Another task of the NRPS Adapter is to register automatically its domain through the "AddDomain" operation of the Topology Web Service of the Network Service Plane instance that connects the domains. Also, it must register its domain's border Endpoints ("push model") by calling the "addEndpoints" method of the same WS of the NSP. The Adapter can get the Endpoints registered in the NRPS periodically and send them to the NSP to have an updated version of the underlying topology.

The resource provisioning capacities of an existing domain controller may be (much) less or dissimilar compared to the NSP. In those cases developing the Adapter is much more a matter of implementing the reservation interface operations in contrast to the translation of existing operational features. Furthermore, some extended features may be supported only by a subset of the systems. The supported features of a system can be queried using the "getFeatures" operation of the Reservation Web Service.

Below are described the individual adapters for each of the UCLP, ARGON and DRAC domain controllers.

2.1.1 DRAC Adapter

The DRAC Adapter wraps the DRAC domain controller. It does so by extending a client for the DRAC controller's web services. The DRAC controller has 3 web services which can be used to reserve resources and manage the DRAC domain topology.

DRAC's web services provisioning capabilities resembles those of the NSP. Therefore developing the DRAC NRPS Adapter is more a matter of transformation of DRAC operations rather than implementing NSP operations.

Below both for reservation and topology service the necessary translations of DRAC specific operations and types to NSP specific operations and types are described.

Here are described the necessary operational translations to be made to transform an NSP reservation request to a DRAC controller reservation request. The request-response flow proceeds as follows:

1. A request is made within the NSP for the reservation web service of the Adapter. This request is performed by a web service client for the Adapter's reservation web service. This client is an integrated part of the NSP manager.

2. When the request has been received by the reservation web service the Adapter transforms this request into a request for the DRAC controller. This request is performed by a web service client for the DRAC controller's reservation web service. This client is an integrated part of the DRAC Adapter.

3. The DRAC controller's reservation web service returns a response.

4. The Adapter transforms this DRAC specific type response into an NSP common data type response, i.e. the response type expected by the original reservation request (step 1).

When reserving resources the DRAC controller creates reservation identifiers. These identifiers are of a different type compared to the NSP reservation identifiers. It is decided (for D1.2) that a reference table for mapping these identifiers is used within the adapter. The reservation identifier to be returned to the NSP is created within the Adapter and kept within the reference table.

2.1.2 ARGON Adapter

To a large extent, ARGON already implements the functionality that can be offered through the Reservation Web Service interface defined in D1.1. A major exception is the support for pre-reservations grouped into "jobs" that can be confirmed or cancelled as a batch with a single call. This optional new functionality is not required for basic NRPS inter-operability and will be added either in the ARGON adapter or in ARGON itself at a later stage.

The ARGON Adapter communicates with ARGON asynchronously via the Java Message Service (JMS) interface. The main task of the Adapter is to translate a request received across the Reservation Web Service interface to an appropriate message which is forwarded to the ARGON core. The Adapter execution thread then waits for the appropriate answer message which it translates to a reply that is sent back across the web service interface.

When the ARGON Adapter is started, it furthermore requests information on the available border endpoints from the core system and registers the domain and those border endpoints with the Network Service Plane instance. Since the ARGON Adapter implements this "PUSH model" to register its domain with the Network Service Plane, it does not offer a Topology Web Service as defined in D1.1.

2.1.3 UCLP Adapter

Since UCLP did not implement advance reservations in its original version, the system has been adapted to support them. Hence, the UCLP system was modified to add the missing functionalities in order to allow this kind of reservations. The adaptation was done by means of a new module (Advance Reservation System - ARS) that contains all the functionalities needed to create advance reservations as they are described in D1.1. When any of these functionalities are called, the ARS processes the request, computes the results and then, returns the response to the requestor (to the adapter or even a human user). The call of the methods is done through a WS, that passes the request to the ARS and this one communicates directly with the UCLP core (with the Lightpath WS specifically) to establish the connections. The ARS WS can be accessed by the UCLP Adapter or by a local management WEB-based application. Regarding the topology of the domain, it is saved in a configuration file that contains the endpoints, the links and the nodes of the domain, inserted through the UCLP GUI.

The UCLP Adapter, for its part, consists mainly of a “redirector” of requests. The only thing that it has to do is to translate the NSP requests to the UCLP requests and call the UCLP ARS WS functions. This translation consist in creating the request objects that the UCLP WS understands and fill them with the information of the objects that the NSP understands, since they are different (MUSE for the NSP and GT4 for UCLP). For the UCLP replies to the Adapter, the inverse operation has to be done.

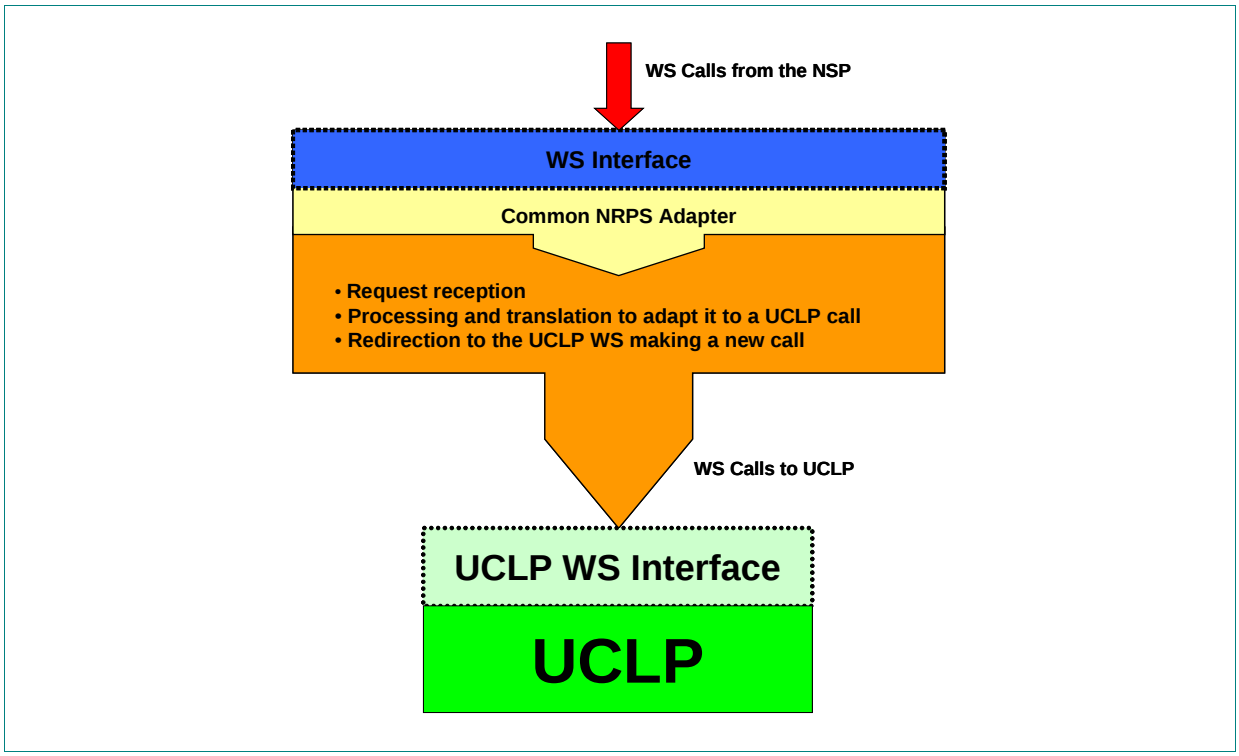

Figure 2.6: Behaviour of the UCLP NRPS Adapter

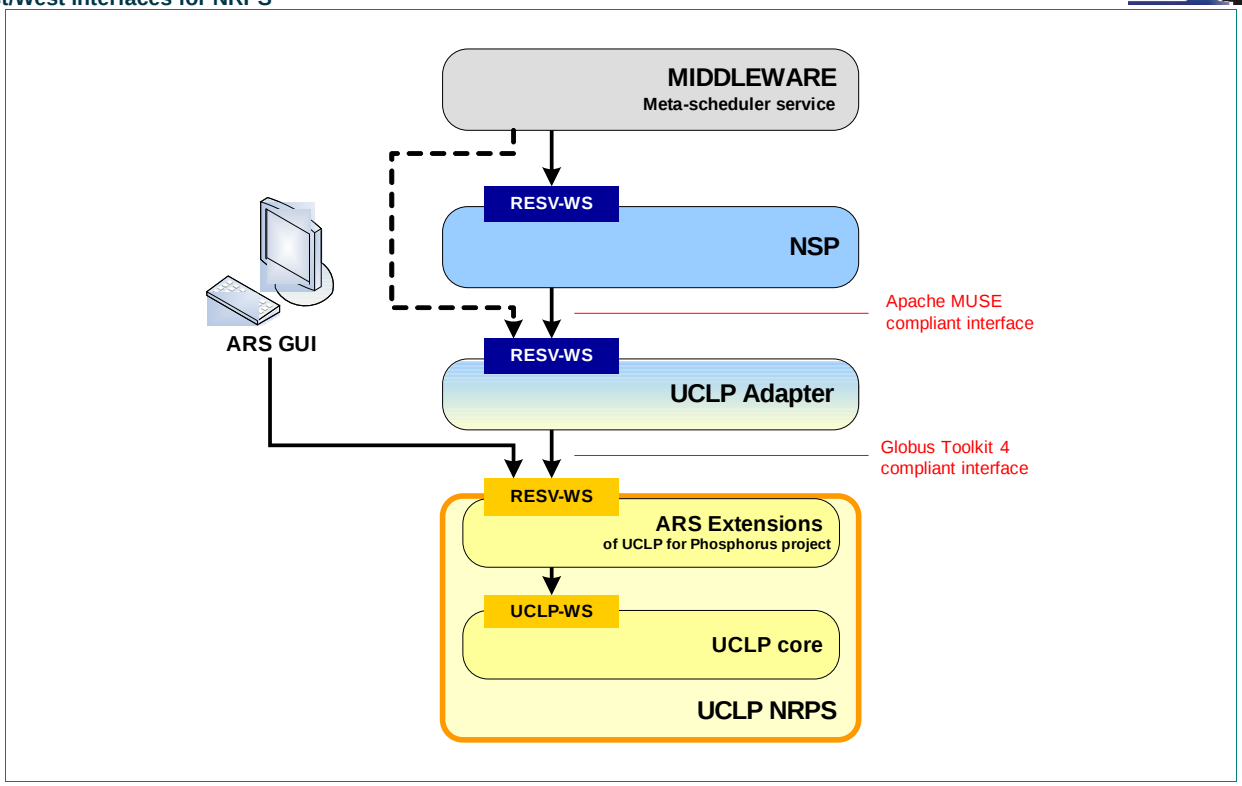

Figure 2.7: Subscriber relationship in the Phosphorus WS architecture using the UCLP Adapter

2.2 NRPS Manager

The NRPS Manager is a module of the NSP in charge of the direct communication with the NRPSs. When the NSP has to send an operation request to the NRPS layer, the manager coordinates the calls to all the NRPSs involved and returns the replies depending on the results received.

The manager uses controllers to manage each NRPS. These controllers are created when a request has to be sent, and they communicate with all the NRPSs in parallel in order to save time by allowing the NRPSs to process the calls at the same time. When all the responses are received, the manager returns them to the caller.

2.3 Path Computer

The Path Computer module is designed to calculate the inter-domain paths. The interfaces of the Path Computer are shown in Table 1. When a new instance of the Path Computer is created, it reads all border Endpoints of all domains and all inter-domain links. After that, one ore more services, with begin and end time, are added. Path computation requests are grouped on a per service basis. When calculating a path, blocking of resources (i.e. resources that are in use by another service that is at least partly overlapping in time) is taken into account. All paths belonging to a specific service are calculated at the same time but can be selectively read from the Path Computer one by one. Each path is returned as a list of tuples of Endpoints. Each tuple consists of two Endpoints of the same domain. The calculation of the intra-domain parts of a path is left to the Path Computer of the domain's NRPS. If an NRPS returns that one or both Endpoints are not available, or no connection between the endpoints is possible for this request, the unavailable resources can be pruned for this path computation and a new set of paths can be computed.

The Path Computer has the following interfaces:

Table 1: Path Computer Interfaces

Interface	Parameters	Exceptions	Result type	Description
addService	long startTime long endTime int serviceId	InvalidServiceIdException	void	Add a service to the path computer's state. The start and end times can be given in arbitrary time units since they are only used to calculate which services overlap in time and which do not.
addConnection	Endpoint source Endpoint destination int serviceId int connectionId	EndpointNotFoundFaultException DatabaseException InvalidServiceIdException InvalidConnectionIdException	void	For a specific service add a connection to the path computer's state from Endpoint source to Endpoint Destination.
computePaths	int serviceId	PathNotFoundFaultException InvalidServiceIdException	void	Compute all paths for all connections added for a specific service.
getPath	int serviceId int connectionId	InvalidConnectionIdException	List<Tuple< Endpoint, Endpoint>>	Get shortest path for a certain connection of a specific service.
pruneEdge	int serviceId int connectionId Endpoint src Endpoint dst	EndpointNotFoundFaultException InvalidServiceIdException InvalidConnectionIdException	void	Prune an intra-domain edge from the internal topology graph of this path computer instance.
pruneEndpoint	int serviceId int connectionId Endpoint endpoint	EndpointNotFoundFaultException InvalidServiceIdException InvalidConnectionIdException	void	Prune an Endpoint from the internal topology graph of this path computer instance.

3 Detailed Formal Description of East/West Interface Operations and Data Types

This section contains the formal description of NRPS interoperability services and data types as described in the WSDL service description files and the XML Schema files respectively. Considered only are those interface operations that have been implemented and their associated data types.

3.1 Reservation Service

Below are the definitions of the currently implemented Reservation service operations.

Table 1: Reservation service operations

WSDL operation name	**createReservation**
Description	Creates the reservation of a path between 2 endpoints considering the specified constraints
Input parameters (XML type)	Service : ServiceConstraintType JobIdentifierType : long NotificationConsumerURL : string
Output parameters (XML type)	JobIdentifierType : long ReservationIdentifierType : long Detailed result : ConnectionAvailabilityType
WSDL Operation name	**isAvailable**
Description	Checks whether the specified service is available
Input parameters (XML type)	Service: ServiceConstraintType JobIdentifierType : long
Output parameters (XML type)	DetailedResult : ConnectionAvailabilityType
WSDL Operation name	**getStatus**
Description	Returns the status of a service
Input parameters (XML type)	Service : ServiceConstraintType JobIdentifierType : long
Output parameters (XML type)	ServiceStatus : ServiceStatusType

3.2 Topology Service

Below are the definitions of the currently implemented Topology service operations.

Table 2: Topology service operations

WSDL operation name	**addDomain**
Description	Adds the domain to the NSP
Input parameters (XML type)	Domain : DomainInformationType
Output parameters (XML type)	Success : boolean
WSDL operation name	**addEndpoint**
Description	Adds an endpoint to the NSP
Input parameters (XML type)	Endpoint : EndpointType
Output parameters (XML type)	Success : boolean

3.3 Data Types

Below (Table 3) are described all data types that are used in the WSDL operations as described in sections 3.1 and 3.2. All types which make up the composite of another type are also described in the table.

Table 3: Data types

Data type name	**Description**	**(composite) Type**
ServiceConstraintType	Type of constraint on the service	*extends* ServiceType *sequence of* Connections : ConnectionConstraintType
ServiceType	Type of service	ServiceIdentifierType : int TypeOfReservation : ReservationType *one of* FixedReservationConstraints : FixedReservationConstraintType DeferrableReservationConstraints : DeferrableReservationConstraintType MalleableReservationConstraints : MalleableReservationConstraintType AutomaticActivation : boolean
ReservationType	Type of reservation	*one of* fixed : string deferrable : string malleable : string
FixedReservationConstraintType	Constraints for fixed reservations	StartTime : dateTime * *Indicates the time when the service should start* Duration : int * *Indicates the duration of the service in seconds*

DeferrableReservationConstraintType	Constraints for deferrable reservations	StartTime : dateTime * *The earliest point in time when the connection would be useful* Duration : int * *Indicates the duration of the service in seconds* Deadline : dateTime * *The latest point in time when the service will be useful*
MalleableReservationConstraintType		StartTime : dateTime * *The earliest point in time when the connection would be useful* Deadline : dateTime * *The latest point in time when the service will be useful*
ConnectionConstraintType	Type of constraint on the connection	*extends* ConnectionType MinBW : int MaxBW : int MaxDelay : int DataAmount : long
ConnectionType	Type of the connection	ConnectionIdentifierType : int Source : EndpointType Target : EndpointType Directionality : int * *Possible values: 0="unidirectional tree", 1="bidirectional tree", 3="full mesh"*
EndpointType	Information about the endpoint	EndpointID : EndpointIdentifierType Name : string Description : string Interface : EndpointInterfaceType DomainIdentifierType : string Bandwidth : int * *Bandwidth of the port in Mbps*
EndpointIdentifierType	Type used to identify endpoints	TNAType : string * *Type used for TNA addresses*
EndpointInterfaceType	Inter-domain, local end point	*one of* user : string border : string
ConnectionAvailabilityType	Availability of the connection	ServiceIdentifierType : string ConnectionIdentifierType : int Availability : int * *Allowed values: 0=available 1=source port unavailable 2=destination port unavailable 3=source and destination port unavailable 4=no path between source and destination 8=availability was not checked* MaxBW : int * *Maximum available bandwidth in Mbps (only set if the corresponding MaxBW was set in the availability request)*
ServiceStatusType	Type for the service status	*extends* Servicetype Status : StatusType DomainStatus : DomainStatusType Connections : ConnectionStatusType
StatusType	Type of the status	*one of* unknown : string pending : string active : string completed : string cancelled_by_user : string cancelled_by_system : string

		setup_in_progress : string teardown_in_progress : string
DomainStatusType	Type of the domain status	DomainIdentifierType : string Status : StatusType
ConnectionStatusType	Type of the status of the connection	*extends* ConnectionType Status : StatusType DomainStatus : DomainStatusType ActualBW : int * *Actual bandwidth in Mbps*
DomainInformationType	Type of the information of the domain	DomainIdentifierType : string Description : string ReservationEPR : anyURI * *Endpoint reference of the NRPS reservation adapter* TopologyEPR : anyURI * *Endpoint reference of the NRPS topology adapter* *list* TNAPrefixType : string * *List of TNA prefixes the domain is responsible for*

3.4 Software Development and Tools

A number of modeling, coding, code quality control and code management tools have been used for developing the software modules. They are summarized in Table 4.

Table 4: WP1 Development Tools

	Development tool
Modeling environment	StarUML
Software coding language	Java
Web Service implementation and xml to java type mapping frameworks	Apache Axis, Apache Muse, JAXB
Relational database tables and Java types mapping framework	Hibernate
IDE (Integrated Development Environment)	Eclipse
Server	Apache Tomcat
Database	MySQL
Code quality control	JUnit, FindBugs, JLint, PMD, Checkstyle

Code versioning and management	Trac

4 Conclusions

This deliverable has focussed on interoperability between individual NRPSs to enable a multi-domain spanning network resource provisioning. The deliverable describes how the implementation of a set of interfaces enables three NRPSs: UCLP, ARGON and DRAC, and the Thin NRPS for GMPLS to interoperate and be controlled by one superposed NSP. Also an essential set of supporting software components and their functions have been described.

The descriptions in this deliverable have already led to a working prototype, also delivered by WP1, which proves the concept of multi-domain spanning network resource provisioning. Results of running the prototype will be evaluated and enable us to further specify and improve the specifications of the interfaces and supporting software modules. This will then allow us to develop a more mature application which will then be enhanced to enable interoperability with the Grid Middleware.

5 References

[ARGON] - ARGON – Allocation and Reservation in Grid-enabled Optic Networks“ http://www.viola-testbed.de/content/index.php?id=reports
- Project home page: http://www.viola-testbed.de
- B2.3.2 (2006) "Program Documentation of the Reservation System User Interface" http://www.viola-testbed.de/content/index.php?id=reports

[DRAC] http://www.nortel.com/solutions/optical/collateral/nn110181.pdf

[UCLPv2] Project home page: http://www.uclp.ca/

[WP1-CODE] Subversion code repository of WP1.
https://svn.uva.netherlight.nl/projects/phosphorus-wp1/trunk/src/eu/ist_phosphorus/
Username: phosphorus-wp1-guest ; password: read4me

6 Acronyms

[ARGON]	Allocation and Reservation in Grid-enabled Optic Networks
[ARS]	Advance Reservation System
[DB]	Data Base
[DRAC]	Dynamic Resource Allocation Controller
[EPR]	Endpoint reference
[GMPLS]	Generalized Multi Protocol Label Switching
[NRPS]	Network Resource Provisioning System
[NSP]	Network Service Plane
[UCLPv2]	User Controlled Light Paths version 2
[URI]	Uniform Resource Identifier
[URL]	Uniform Resource Locator
[WP]	Work Package
[WS]	Web Service