**Journal of Internet Services and Applications manuscript No.**
(will be inserted by the editor)

# Heterogeneous Resource Federation with a Centralized Security Model for Information Extraction

**Milad Daivandy · Denis Hünich · René Jäkel · Steffen Metzger · Ralph Müller-Pfefferkorn · Bernd Schuller**

Received: date / Accepted: date

**Abstract** With the continuous growth of data generated in various scientific and commercial endeavors and the rising need for interdisciplinary studies and applications in e-Science, easy exchange of information and computation resources capable of processing large amounts of data to allow for ad-hoc co-operation becomes ever more important. Unfortunately, different communities often use incompatible resource management systems. The goal of this work is to alleviate the difficulties occurring on bridging the gap between different research eco-systems by federating resources and thus unifying resource access.

To this end, our solution presented in this paper outlines a secure, simple, yet highly interoperable and flexible architecture using RESTful Web services and WebDAV. While, first and foremost in the Grid computing domain, there are already standards and solutions in place addressing related problems, our solution differs from those approaches by allowing for federating data storage systems that are not aware of being federated. Access to these is enabled by our federation layer using storage system specific connectors. Hence, our federation approach is intended as an abstraction layer on top of existing storage or middleware solutions, allowing for a more uniform access mechanism. Additionally, our solution also allows for submission and management of computational jobs on said data, thereby federating not only data but also computational resources. Once resource access is unified, information from different data formats can be semantically unified by information extraction methods. It is our belief that the work in this paper can complement existing Grid computing efforts by facilitating access to data storage system not inherently available via commonly used Grid computing standards.

**Keywords** Grid Computing · Data Federation · Metadata Federation · Resource Federation · WebDAV · Single Sign-On · Identity Federation · Centralized Security · Identity Management · Trust Delegation · Interoperability · Uniform Access · Information Extraction · Knowledge Extraction

M. Daivandy
Jülich Supercomputing Centre, Forschungszentrum Jülich, 52428 Jülich, Germany
E-mail: j.daivandy@fz-juelich.de

D. Hünich
Center for Information Services and High Performance Computing (ZIH), 01062 Dresden, Germany
E-mail: denis.huenich@tu-dresden.de

R. Jäkel
Center for Information Services and High Performance Computing (ZIH), 01062 Dresden, Germany
E-mail: rene.jaekel@tu-dresden.de

S. Metzger
Max-Planck-Institut for Informatics,
66123 Saarbrücken, Germany
E-mail: smetzger@mpi-inf.mpg.de

R. Müller-Pfefferkorn
Center for Information Services and High Performance Computing (ZIH), 01062 Dresden, Germany
E-mail: ralph.mueller-pfefferkorn@tu-dresden.de

B. Schuller
Jülich Supercomputing Centre, Forschungszentrum Jülich, 52428 Jülich, Germany
E-mail: b.schuller@fz-juelich.de

## 1 Introduction

In recent years, the transparent and secure handling of large data sets has become increasingly important for a broad range of different scientific and commercial applications [7,23], in particular for handling huge amounts of scientific data for simulation or analysis in various research fields [6]. In the past years, Grid computing [17] has evolved for many different disciplines, ranging from solving computationally intensive tasks to the provision of services for application steering, user management and the creation of complex workflows, but also of resources for data management [37]. A strong advantage of Grid computing is the ability to maintain distributed and heterogeneous resources using a middleware layer for distributing computational jobs or delegating user access to data storage space. The use of distributed computing resources becomes more and more important in many research fields using state-of-the-art information systems to even enable collaborative work over institutional boundaries. To some extent, Grid and, more recently, Cloud resources can be used for distributing computing tasks or providing access to distributed data resources. Since, generally speaking, the resource demand in scientific computing is continuously on the rise and the potential for interdisciplinary work is large, an easy and secure access to computing resources becomes even more important.

For large-scale projects usually including lots of partners from different organizations, the provision of relevant data can not be assured by a single organization alone, but has to be organized among partners across organizational boundaries. In the past, these efforts were often realized by nation-wide or even international cross-linking of computing centers providing a Grid of computing resources [17]. For the scientific community, this has led to a rather large provision of access to High Performance Computing (HPC) and High Throughput Computing (HTC) resources by large data centers. Examples for this are XSede [41] in the U.S. or the EGI federation [11] supported by the EU. Other approaches, such as the national German Grid Initiative D-Grid [10], emphasised and spread the idea of using Grid resources and methods of distributed computing in general in the their local scientific communities. In this context, we have examined capabilities for further decreasing hurdles typically faced by user groups intending to adopt Grid computing to their daily work. The WisNetGrid [40] project of the aforementioned D-Grid initiative has investigated a more general approach for facilitating access to distributed storage solutions in general, ranging from traditional Grid middlewares over databases to Web-based resources.

By extending the range to additional data and information resources in general, the WisNetGrid project aims at furthering the potential for using data across different scientific disciplines and related fields, such as physics, biology, medicine, geographic information and humanities. Some of the aforementioned knowledge resources are publicly available via the Internet, others require authentication due to project-specific or commercial constraints. The common theme is that these resources, sometimes provided by governmental authorities [31,36], are often offered as databases or ontologies and thus might be of interest to interdisciplinary studies. However, these data are usually not accessible to Grid computing in a traditional and standardized manner. By having our access layer support a broader range of resource types and different underlying access control mechanisms, our solution is capable of providing uniform access to most distributed storage systems, both traditional Grid and non-Grid.

In this paper, we introduce a system for federating multi-organizational and heterogeneous computer resources into a uniform namespace and making them accessible by way of a uniform interface using the widely supported WebDAV standard. We understand resource federation as logically joining resources, primarily data resources, from different distributed heterogeneous resources without moving or copying data from these resources to the resource federation system. Our resource federation system contrasts from related work in so far as the resources to be federated are not aware of being used in a federation context. This means, that a particular storage system is still being operated by the individual institutions or operators and that access modalities already in place are not modified. Our solution federates storage systems by way of storage system specific connectors (e. g. a MySQL connector for federating a MySQL database) running in the federation system backend and making use of our security model based on delegated authentication by way of supplied user credentials. Hence, it is safe to classify our approach as user-centric. See section 3 for more details regarding both the architecture and the realization of this federation mechanism.

Operators of our resource federation system do not necessarily have to own the distributed resources to be federated, but have to take care of two matters. First, they need to negotiate and manage collaborative agreements for partners who wish to participate in the federation system. Second, they need to provide a connector element for each storage system to be federated. Although the main focus lies on the federation of data resources, our system also supports submission and management of jobs operating on these data resources. Also, this work outlines how essential information contained in federated data can be further unified and leveraged by applying information extraction methods. While information extraction is computationally intensive, our architecture allows for parallelizable computational resources to be used.

To illustrate how higher-level applications can easily make use of uniform data access and to provide an example for data federation and information unification, we discuss a use case from the humanities. In this use case, the information extraction module is a higher-level application that benefits from uniform access to data resources provided by the system described in detail in section 3. This enables the extraction module to provide the user with a uniform representation of information found in source files of multiple origins without the user having to deal with the particular storage systems involved himself.

In particular, assume a humanities-oriented application scenario, where the research community is trying to link works of and about particular authors. Such works might be available in digital form, but held by different legal entities. In adopting our solution, each group agreeing to share information only needs to integrate its data into the federation system, and all community members who are granted the corresponding permissions can access sources from all participating data providers via a uniform interface.

Additionally, the extraction system, acting on behalf of a particular user, can access all files the user holds access rights for. Thus, it can, for instance, lift entity mentions, i.e. mentions of persons, places etc. onto a semantically unambiguous level, linking e.g. different writings of a name or even different names for the same thing or person. This allows, for instance, to trace a particular semantic entity in different contexts without the need for expert users who know all possible ways the semantic entity could be addressed in these contexts. This also allows for interdisciplinary work and linking sources of different research communities. For instance, places or persons occurring in literary texts could be linked with information retrieved from historical or geological information sources. This use case and the interaction between the high-level extraction services with the resource federation component is described in section 5.

In the following section 2, we discuss our approach regarding an infrastructural centric view of larger consortia and focusing on technical aspects of related solutions with similar approaches to realizing federated access to distributed storage systems. We intend for our resource federation system to strongly support the user with accessing the resources required for his use case. To fulfill the aforementioned goals, developing a flexible and highly interoperable security solution was essential. Section 4 discusses how our solution is able to federate different distributed resources by accessing data from those resources while conforming to respective resource-specific authentication and authorization mechanisms.

## 2 Related Work

In the Grid and Cloud computing domains, middleware solutions are used as abstraction layers to facilitate access to and enable interoperability between (geographically) distributed computing infrastructures such as super computers, high-performance clusters and larger computing centers [18]. For accessing these computing resources, every middleware offers different services for job submission and management, data management on the application level, or user administration for Grid access and support.

Today, larger consortia have been established to focus efforts from individual data centers towards a more service-oriented approach for scientific communities to use larger computing resources, such as XSede [41] in the U.S. or data centers unified under the European Grid Infrastructure (EGI) [11]. This range of services is provided by a management layer to local data centers (service providers) running individual HPC or Grid resources. Typically, this does not mean that every provider offers the same services, either for accessing data storage space or for computing resources. Furthermore, services of different middleware solutions might not be compatible with each other.

The respective types of middleware solutions at particular data centers might also represent a technical hurdle to be taken by new user groups, or by trying to combine existing technical solutions from different user groups to enable collaborative work between these groups. Typical hurdles are different security infrastructures (e. g. authentication via username/password or X.509 certificates [8]) or differing data representations (e. g. database or file systems), which are handled differently by various middlewares.

Within the context of our project, we have investigated a more generalized approach for addressing a broad range of resources, not only Grid middlewares. Therefore, we do not compete with well-established Grid computing middlewares. Instead, we offer an additional way for accessing distributed resources by providing an additional access mechanism and do not intend to replace existing Grid solutions.

A different, but interesting approach to distributed Grid resources is the access mechanism via science gateways, which also hides the technical details of the specific data access mechanism or its clients from users and provides a transparent interface. Several approaches are available to address Grid resources [42,12], but often with a strong focus on community-only requirements and services. Sharing data or services among different groups is still not the main focus of these efforts.

Addressing the latter point, the data integration layer of three Grid projects from the German Grid initiative has been analyzed in terms of standardization [30] on the architectural level, but to our knowledge this work remains on the conceptual level. To tackle these interoperability concerns, the EMI project [14] aims at standardizing services of four different middleware solutions on a technical level. However, while the scope of the EMI project shares related goals with our work, EMI solely focuses on Grid resources, whereas our work additionaly considers non-Grid based data sources.

Using Grid computing, either for data or computing relevant concerns, also necessitates user authentication and authorization [22] mechanisms. To facilitate access and to increase usability of project-specific Grids, the development of Web browser based services, called Grid portals, was observed [15]. This combination of Web browser access and Grid computing requires passing user credentials throughout the federation system to transport user requests from the HTTP layer to the middleware execution layer. This concept of Single Sign-On was first adopted to the Grid computing domain by using proxy certificates [26,35]. In general, this concept is also integrated in science gateways as mentioned before, and is realized in various community based projects, e.g. as described for PolarGrid [20] or other community based projects, such as MosGrid [25, 19] (also a D-Grid project). Apart from the authentication federation via OpenID [32], PolarGrid supports a more general non-OpenID based mechanism to also support further authentication services, albeit these are still in development [20]. Nonetheless, this marks an interesting approach to extend access and visibility of Grid computing systems to other access mechanisms like Social Networks.

A further solution to pass user credentials via Web Single Sign-On across organizational boundaries is Shibboleth [21]. It is based on SOAP Web services [39] and uses SAML 2.0 [29] to interact with arbitrary identity providers. Apart from authentication, SAML 2.0 also provides means for authorization and for user attributes (key-value pairs of arbitrary data). In use cases where components are not based on platforms SOAP Web services are easily available for or applicable to, the SOAP Web service dependency of SAML 2.0 can be a drawback, especially regarding our intention to provide a highly interoperable system. The GridShib [27, 2] project combined Shibboleth and the Globus Toolkit to map SAML assertions to X.509 certificates [28]. However, the drawback is the restriction to the Globus Toolkit.

Concerning Web Single Sign-On in general, [33] outlines weaknesses of security protocols based on insecure properties of Public Key Infrastructure and the Domain Name System. Furthermore, a more secure cookie type is introduced and used as part of a proposal for a more secure Single Sign-On protocol. In contrast to usual cookies, it also contains a list of public keys denoting eligible target servers. During the SSL handshake between a Web browser with such a cookie and the target server, the target server needs to match one of the aforementioned public keys. A subsequent proof-of-concept with security evaluation regarding multiple attack types is given and shows that this concept secures against a malicious website impersonating a valid target server.

## 3 Resource Federation

Accessing and manipulating data stored in different data storage systems such as databases, Grid data management or file systems within the context of a single use case can be difficult and time-consuming. First, there are many possible data representations (e.g. entries in a database vs. file documents), access and security protocols. Second, a use case requiring collecting and analyzing data from different data storage systems must know how to communicate with all of them and how to interpret the received data. Adding support for a new type of data storage system necessitates some sort of connector extension with regard to the required communication and security protocols, but also requires support for new data formats. This becomes increasingly inconvenient as more resources of different resource types need to be integrated.

In Grid environments, heterogeneous resources are mostly hidden from the user by a middleware layer which provides a uniform view on the resources. The middleware communicates with the underlying resources and is responsible for transforming client requests into formats consumable by the actual requested resources and vice versa. In our case, the aforementioned resources might be different data storage systems, as described in the beginning of this section.

### 3.1 Uniform Access

The resource federation system described in this paper realizes such a middleware layer. It provides uniform access by using the WebDAV protocol[1], which

[1] The specifications can be found online: http://webdav.org/specs/

is an extension of the HTTP/1.1-protocol[2]. Although the HTTP/1.1 protocol already supports methods for reading (GET), writing (POST/PUT) and deleting (DELETE) resources, our resource federation system requires additional methods.

WebDAV provides further operations such as locking (LOCK/UNLOCK), copying (COPY) and moving (MOVE) resources on the respective data storage systems. Furthermore, methods for reading (PROPFIND) and updating (PROPPATCH) metadata of resources are supported which allows to represent resources and corresponding arbitrary metadata with one URI[3] even when they are stored at different locations. The increasing amount of data necessitates the use of data management systems with metadata management support. In the context of our work, we identified WebDAV as a suitable building block for our resource access and federation system, both regarding data and metadata management.

To provide the aforementioned uniform access our architecture consists of the following four components:

*SSO Database* Stores user information, authentication data and credentials

*SSO Server* Central access point of the security infrastructure

*Credential Manager* Graphical user interface for management of external credentials

*Resource Federator* Interface between the user and resources

Figure 1 shows how the four components interact with each other.

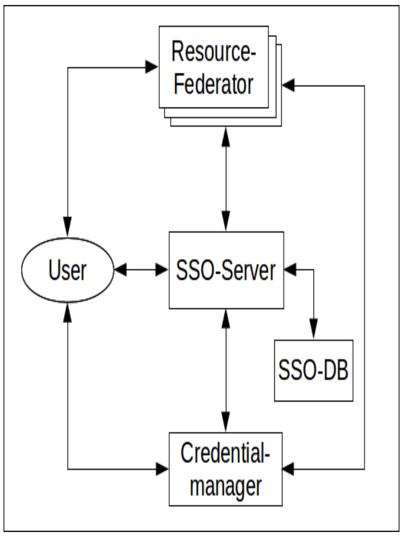

**Fig. 1** Interaction between the components

[2] http://www.w3.org/Protocols/rfc2616/rfc2616.html
[3] Unique Resource Identifier

The SSO Database stores user information (e.g. name or e-mail address), the authentication data for accessing the SSO Server (username and password) and external credentials used by the Resource Federator for delegated user authentication on connected (i. e. federated) resources. The credential type depends on the underlying security model of the resource in question (e. g. a password for a MySQL database or a X.509 certificate for a Grid resource). Credentials are stored plain or encrypted with the public key provided by the server running the Resource Federator. The encryption allows the resource providers to hide security relevant data from third party systems.

The SSO Server is the only component, which is directly accessing the SSO Database and is the starting point for other components requiring security mechanisms. The SSO Server provides both a graphical user interface and a Web service interface. See section 4.2 for more details.

The federation of resources in our namespace is done by the Resource Federator instances. Each of those is able to integrate and process CRUD operations on resources (represented via URIs) depending on the user privileges and supported functions of the used Connectors (see section 3.2 for further details). The privileges required to use an integrated resource are composed of the authentication data (verified by the SSO Server) and the credentials (verified by the service provider of the resource).

The Credential Manager provides a graphical user interface to the external credential management part of the SSO Server Web service interface. Therefore, the Credential Manager is populated with URIs to all Resource Federator instances in the federation used to build an internal collection of all federated resources with corresponding types (e. g. MySQL, IRODS, UNICORE, local file system etc.). This information is used to generate corresponding input masks for external credential management. A MySQL input mask provides fields for username and password specification, whereas a UNICORE input mask provides a Java Applet (which, by default, runs locally on the user's computer) to issue a signed SAML 2.0 trust delegation token using the user's private key. Optionally, the user can encrypt his external credential with the public key of the corresponding Resource Federator instance. Section 4.1 sheds more light on this security model.

## 3.2 Resource Federation

The aforementioned resource federation system is composed of three components:

– Web Server
– Routing Engine
– WebDAV Server

and is illustrated in Figure 2.

The *Web Server* interacts with the client according to the HTTP 1.1 protocol. A request is forwarded to the *Routing Engine* and the system's response is sent back to the client. The *Routing Engine* allows to define routes and to process routed messages using a set of intermediaries. The resource federation system defines a route by two endpoints (start and end points) and a certain number of intermediaries (elements for manipulating messages, from here on individually referred to as *Process*). The start point creates an exchange object (*Message*), stores the client request and an empty response in it and sends it to the endpoint. Before the message reaches the endpoint it is processed by the following *Processes*:

*Preparation* Sets the name of the requested source and the relative path of the data
*Location* Loads information about the respective data storage system
*Credential* Connects to the Single Sign-On server to retrieve a user's external credentials (see section 4.1) for the data storage system hosting the requested resource
*Connector* Provides the interface between the WebDAV server and the specified data storage system
*Exception* Aborts the route and provides detailed error handling for the responses

*Preparation* scans the requested URI, takes the data storage system name and the relative path of the requested resource and stores both in *Message*. Using the data storage system name, *Location* loads information (defined in a configuration file) about this data storage system and adds this information to the *Message*. Then, the external credential required to access the requested resource on behalf of the user is retrieved from the Single Sign-On server and stored in *Message*. *Connector* loads the connector for the requested data storage system with the collected credential and location information and hands it over to *Message*. The connector provides methods to process data and metadata: it is specialized for a certain type of resource and acts as an interface between the resource and *WebDAV Server*. The endpoint of the route creates the *WebDAV Server* environment with the collected data in *Message* and starts it. *WebDAV Server* processes the request with the help of the *Connector* and generates a response, which is sent to the Java Servlet in *Web Server*.

The generic resource federation supports the introduction of new connector types and *Process* implementations (e.g. a billing process). Therefore, extensibility does not necessitate source code changes. Instead the Camel route, defined in the configuration file, must be changed. It is also possible to use other WebDAV server implementations or even a completely different protocol as endpoint. It is up to the specific user group which process has to be provided.

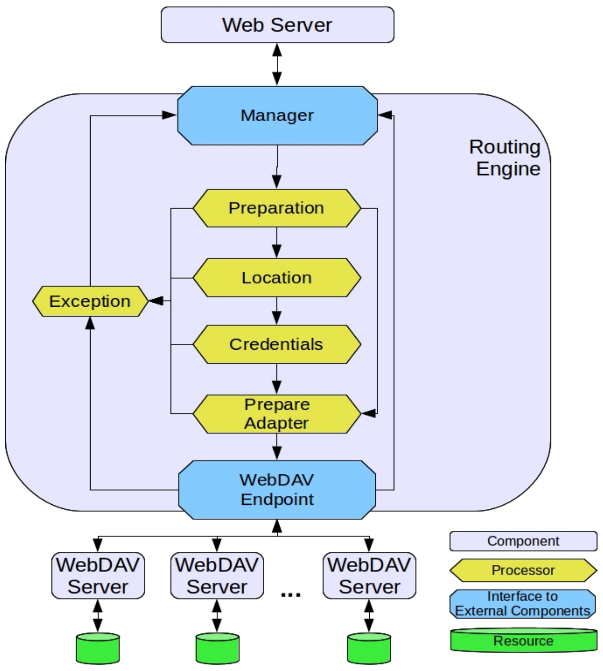

**Fig. 2** Structure of the Resource Federator.

## 3.3 Submission and Management of Computational Jobs

The architecture just described is not restricted to data access and manipulation. Other types of resources can be integrated as well. As a particularly important case, we have integrated data processing capabilities into our system. We chose to do this by implementing an connector which can submit jobs to the UNICORE Grid middleware. The user interacts with this job connector via a WebDAV directory. Job description documents can be uploaded (via HTTP POST) into this directory, while each submitted job corresponds to a file in the directory. Viewing these job files in a browser or downloading them allows to check job status.

The security system described in section 4 is perfectly suited to integrate UNICORE resources, since UNICORE uses a trust delegation system based on signed SAML 2.0 assertions [3]. UNICORE jobs can participate in an active security session, and manipulate data through the WebDAV interface.

## 4 Security Federation

As a multi-user distributed system comprised of multiple applications and potentially spanning multiple organizational boundaries, authentication (identity verification) and authorization (access control after successful authentication) are core requirements of the security infrastructure to apply.

Both for usability and administration considerations, a centralized Single Sign-On authentication approach is suitable for our resource federation system (see section 3). In this context, Single Sign-On is briefly described as follows: on accessing any given application within the resource federation system, a yet unauthenticated user is prompted to supply only one and the same valid security credential (a username and password combination) whereupon he is authenticated to the whole system. All applications thus protected form a shared Single Sign-On domain. Single Sign-Off specifies the reverse property where a user signing off at any given application within a Single Sign-On domain automatically terminates his access to all other applications within the same Single Sign-On domain.

Authorization comes into play after successful authentication: each subject has a set of roles that are used for access control.

In addition to this traditional sequence of authentication and authorization, we needed a trust delegation mechanism allowing our resource federation system to act on behalf of users to access federated external resources (see section 3). To this end, our security model is required to allow a user to manage his respective resource credentials, giving him the means necessary to add, configure and remove them. Also, our security model had to support users interacting with the system using both Web browsers and WebDAV clients, thereby necessitating two different security interfaces.

In addition to the aforementioned requirements, our security model had to accommodate the heterogeneous and distributed characteristics of the system described in chapter 3 and thus be interoperable. Finally, we intended our security solution to also be applicable to similar use cases aiming at heterogeneous resource federation, albeit without necessitating a complex security stack imposing too much of an interoperability overhead (e. g. SOAP-based, which is merited in SAML-based trust delegation scenarios, but not necessary for this scenario).

### 4.1 Security Model

Our security model consists of four actors: security provider, subject, service provider, and external system. Obviously, the security provider is the server-side part of our central security system we labeled 'Single Sign-On server'. Both terms are used interchangeably.

To define what a subject exactly encompasses, a definition is in order: in the course of this work, we regard any entity that can make requests to resources secured by the security provider as a subject. According to this, a subject can be a human, i. e. a user, or a non-human entity like a service, an application or, more generally, a computer system. A principal is an identifying attribute (or a set thereof) of an authenticated subject, such as a unique key or a user account. In our security model, it consists of a unique id, a password, roles, external credentials and further optional attributes, such as user details should the subject be a user.

A service provider is an application that delegates security to a Single Sign-On domain. Situated outside of a Single Sign-On domain are external systems containing resources a subject has access tokens to. This signifies use cases where a service provider acts as an intermediary between a subject and an external system containing resources requested by the subject. To fulfill this function, an intermediary service provider uses the corresponding external credential of the subject, thereby making it a necessity for the subject to trust the service provider with his extended credentials.

An external credential consists of a resource name denoting the external resource (e. g. a database), the subjects's ID on that external system (if applicable) and the actual credential (e. g. a password). Since the latter is persisted in serialized form, its format can be arbitrary. It can be a plain text password, a X.509 certificate [8] or even a SAML assertion [29].

In this security model, the combination of subject ID and password is used for authentication to the Single Sign-On domain. After successful authentication, the security provider can map the subject to its corresponding principal and thus provide respective service providers with its roles for access control and external credentials for trust delegation. Once authenticated, a subject is also identified by a unique session, itself consisting of a unique ID and the aforementioned principal. Said unique session ID is returned to a subject after successful authentication to be used as an identification token for subsequent requests. Its validity can be terminated on the client and server sides (see section 4.2 of this chapter).

For trust delegation to work, a subject must provide external credentials for the external resources he intends to access. This needs to be done beforehand and only once, unless the actual access token is changed on the external system's side. Optionally, the subject can opt to encrypt each external credential with the public key of a service provider, thereby restricting access to that service provider alone that can decrypt this credential using its private key.

### 4.2 Components and Interaction

The Single Sign-On server was implemented with the Java programming language as was the Single Sign-On client library used by service providers. Furthermore, principals and corresponding authenticated sessions are persisted using an extensible persistence layer allowing for both local and remote databases and thereby implicitly supporting data replication depending on the specific persistence backend.

The Single Sign-On server offers Web browser and Web service [39] interfaces, both relying on Transport Layer Security for channel security. The former uses Simple Transport Layer Security, the latter requires mutual authentication using the client-authenticated extensions.

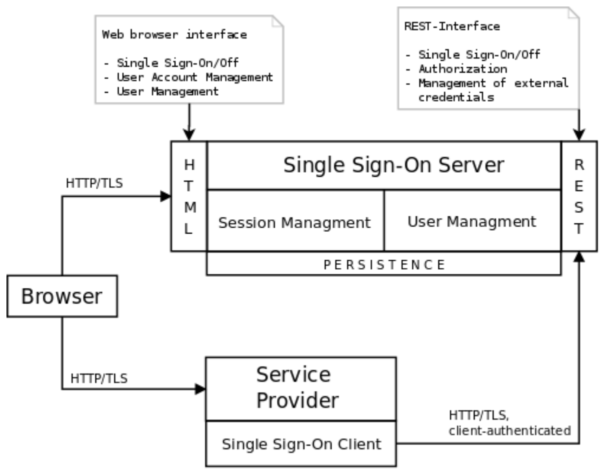

**Fig. 3** Security components

The Web browser interface is required for initial account registration and for account self-management. It also serves as the single point of Single Sign-On and Single Sign-Off for Web browser users and provides administrative functions, such as listing, locking, deleting user accounts and assigning user roles. The aforementioned unique session ID used as an identification token is stored as a Cookie in the user's Web browser, thereby rendering that Cookie a client-side reference to the user's actual session in the Single Sign-On server. The Cookie is secured by both the 'Secure' and 'HttpOnly' options, together limiting Cookie communication to encrypted HTTP connections. Cookie validity is terminated by the Single Sign-On server after a session's idle time exceeds a customizable global session lifetime. Client-side Cookie termination is either directly performed by the user via the Web browser interface Single Sign-Off, thereby triggering the subsequent deletion of the corresponding session in the Single Sign-On server. This also leads to the Cookie being deleted in the user's Web browser. Closing the Web browser without performing Single Sign-Off will also delete the Cookie, but with the session still existing in the Single Sign-On server until it expires. For Web browser Single Sign-On to work, a user's Web browser needs to store one clone of the aforementioned Cookie for each service provider. Since Cookies are inherently bound to WWW domains, we implemented the cross-domain Cookie sharing algorithm outlined in [4].

For our Web service interface, we employed the Representational State Transfer (REST) [16] paradigm, since it uses HTTP methods, thereby only requiring a very basic and practically ubiquitous Internet protocol. This design choice was further influenced by the availability of open-source HTTP client libraries for a plethora of programming languages and systems, thereby strongly supporting our goal of providing a highly interoperable security solution for resource federation and similar use cases. One of those open-source HTTP client libraries [38] supports 42 programming and scripting languages.

The Web service interface also provides Single Sign-On (see figure 4) and Single Sign-Off, but goes further by adding means for role and external credential retrieval. Also, external credentials can be added, updated and deleted. The aforementioned session ID is supplied as part of the URI of the Web service request from a Single Sign-On client instance to the Single Sign-On server. In the same vein, we decided to use Javascript Object Notation [9] as data interchange format, a lightweight text-based open format that lends itself to data serialization and transmission over computer networks. This combination of HTTP-based Web service and text-based data interchange format makes the security provider highly interoperable.

Security delegation for service providers is facilitated by way of a high-level Java-based Single Sign-On client library abstracting from lower-level Web service data transformation and transmission technicalities. Thus, each service provider can use its Single Sign-On client to authenticate and authorize user requests without having to provide its own fully-fledged security system. Non-Java-based service providers need to provide Single Sign-On client implementations of their own, which in itself only constitutes a low barrier given the aforementioned highly interoperable nature of the Web service interface.

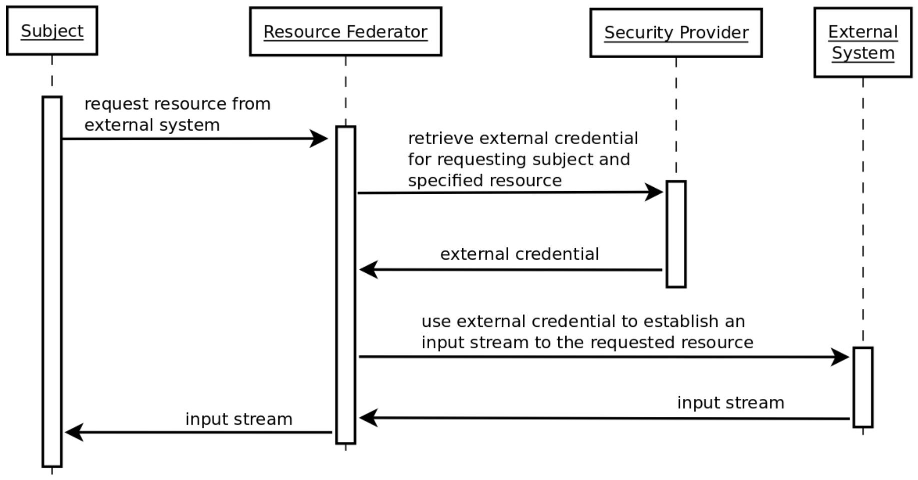
Subject
Resource Federator
Security Provider
External System
request resource from external system
retrieve external credential for requesting subject and specified resource
external credential
use external credential to establish an input stream to the requested resource
input stream
input stream

**Fig. 5** External credential - based trust delegation

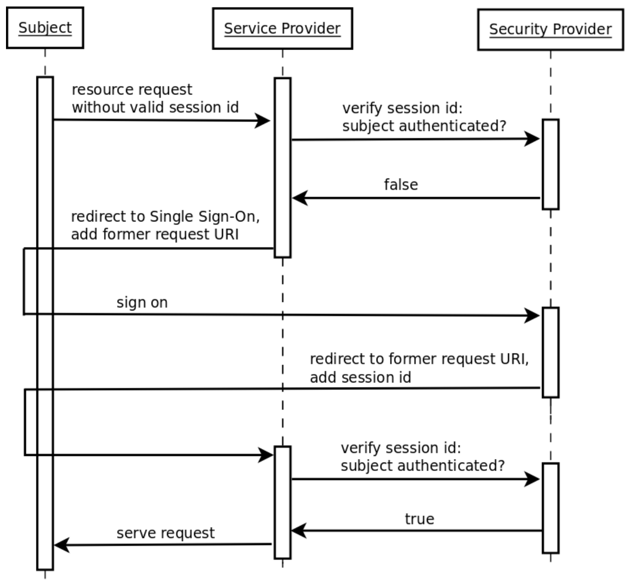
Subject
Service Provider
Security Provider
resource request without valid session id
verify session id: subject authenticated?
false
redirect to Single Sign-On, add former request URI
sign on
redirect to former request URI, add session id
verify session id: subject authenticated?
true
serve request

**Fig. 4** A yet unauthenticated subject being prompted to authenticate via Single Sign-On during a resource request

To reduce architectural complexity, we opted for delegating external credential management to a service provider of its own that makes use of the aforementioned Web service interface as opposed to integrate it within the Single Sign-On server. This service provider, labeled Credential Manager, is a small Java-based Web application using the aforementioned Single Sign-On client library to authenticate and authorize users. Once a user is authenticated, he can use the Credential Manager to create, modify and remove his external credentials.

## 4.3 Caching

Given the security validation necessary for every subject request (see figure 4), it is easy to see why this interaction pattern constitutes a potential bottle neck, since a service provider needs to verify every request on secured resources with the security provider. At the very least, a service provider has to verify that a requesting subject is authenticated. It is safe to assume, that a service provider needs to verify a subject's roles for authorization purposes. If trust delegation is employed, a service provider also has to retrieve a subject's external credential(s) from the security provider. Thus, depending on the security verification pattern required by a service provider (see figures 4 and 5), one subject request can necessitate three different request types with differently sized response data payloads to be made by a service provider to the security provider. In the context of a multi-user system where every user can make concurrent requests, it follows that the network connection between a service provider and the security provider constitutes a potential bottle neck. Hence, we decided on a client-side caching approach to reduce the amount of necessary security verification requests. To this end, each Single Sign-On client caches Client Session instances, a client-side representation of the session concept is outlined in section 4.1 of this chapter. Whenever a user is signed on to the Single Sign-On domain via a Single Sign-On client, the Single Sign-On client creates a corresponding Client Session instance containing that subject's authentication state, roles and external credentials and puts it into the cache. This cache is periodically refreshed with fresh values from the Single Sign-On server. Also, each client session is only populated with external credentials the service provider currently requires to fulfill respective subject requests requiring trust delegation towards external systems. Additionally, invalid cached sessions are detected and disposed of. The client session refresh and the cache clean-up intervals can be customized by the service provider. While this approach reduces the time spent on security verification requests, there is a disconnect between cached security state on the service provider's side and actual security state on the security provider's side. Thus, it is the service provider's concern to choose respective values to strike a good balance between performance and security state consistency.

## 5 Interactive Information Extraction

In the past two sections, we explained how the federation layer allows to access distributed data resources in a uniform way. In this section, we illustrate how the federated data can also be semantically unified using an extraction framework that provides semi-supervised extraction methods. In addition, this serves as a use case to illustrate how higher-level services can make use of the federation layer to access large heterogeneously distributed data sets.

To explore the needs of users in real application scenarios, we work together with user groups from the humanities, for which large repositories of digitized textual data became available in recent years. In an example setting, we integrated their data into our federation layer and developed an interactive knowledge extraction system to help with the analysis of these large data sets. One goal in this domain is to cross-analyze the works of, or about particular figures, like famous authors. By identifying differently formulated references to semantical identical entities underlying semantic links between documents of different authorship can be discovered. For instance, when investigating the life, works and journeys of the famous writer Johann Wolfgang von Goethe in historical texts, references to the person Goethe may vary in different texts. In bibliographical texts he may be referenced by his full or partial name. In many letters, however, he is referred to as "Herr geheimer Rat", a reference to an official position he held. Similarly, names of cities and places can change over time and even the formulations used to express certain facts can differ depending on the author's writing style, the target audience of a text and the time of its creation.

Information extraction methods can link these variations by lifting entity references and statements indicating relations between such entities onto a semantically canonicalized level, where each entity as well as relations between entities are uniquely identifiable, i.e. it allows to represent knowledge expressed in texts in an ontological form.

This helps, for instance, to categorize and investigate large document collections and make them discoverable for non-experts, e.g. allowing to search or cluster documents by the referenced entities. It also allows to link information from different research areas, e.g. to enrich places mentioned by Goethe in his writings with geographical or historical information.

While information extraction on its own aims at semantic unification of varying textual representations, our federated framework allows for easier access to heterogeneously stored source data within a community as well as across different communities. By integration into the unified security and data access model, the extraction system needs not to be aware of the concrete file system, nor does it need to support different security methods. For instance, a user of one community may have access to files within its community stored on different Grid systems, but also may cooperate with another community and thus have separate credentials to access some files of the other community. Since the federation level deals with all security issues, the extraction system only needs to know the Single Sign-On (SSO) credentials to access files across both communities. This would, for instance, allow a researcher from the humanities to have access, and thus apply the extraction machinery, to works of or about Goethe held by different research groups or even data of other disciplines. Using the federation system he would only need his personal SSO account, given that the data owners granted access permissions and integrated their data with the federation system.

In the following we briefly discuss the extraction approach, what information it needs to function and how this information can be provided in an interactive way. Finally, we also discuss how the extraction components interact with the federation layer.

### 5.1 Knowledge Extraction Approach

The extraction framework we provide supports two levels of knowledge extraction. The first level, commonly referred to as named entity recognition (NER), is usually understood as the problem to identify referenced entity types. For instance, occurrences of the strings *"London"* and *"Frankfurt"* in a text can be identified as references to locations and for occurrences like *"Einstein"* and *"Goethe"* the reference type could be a person.

However, recent work in this field allows for uniquely identifying individual entities referenced in texts, given some background knowledge about the domain entities [24,43]. This allows for identifying the string *"Goethe"* as a reference to the particular historical person named *Johann Wolfgang von Goethe*, a famous German writer. The same holds for other references, e.g. to locations. For instance, if the string *"Frankfurt"* appears in the same text about that particular writer it is probably a reference to the city *Frankfurt on the Main*, where Goethe was born, and not to *Frankfurt on the Oder*. The problem to decide which concrete entity is meant by a reference is called *disambiguation*.

To semantically unify different references to an identical entity in such a way, there are several basic requirements. First the system needs to have basic knowledge about the unique entities of a domain, i.e. it needs to know the famous writer Goethe existed. Secondly, type information is of importance, i.e. knowing that Goethe was a human, a writer and so on. Thirdly, the system needs to know which 'names' can potentially refer to which unique entities, e.g. the string "Frankfurt" could refer to at least two different cities within Germany. And finally, any additional relational information, e.g. where Goethe was born or which plays he wrote, can help to solve the disambiguation problem.

Based on this first step relations connecting recognized entities can be extracted from texts as well. Such relations might, for instance, be the birthplace of persons, the books a writer authored, or the movies an actor participated in. While there are different approaches [1,5,34], we apply an iterative pattern-based approach based on [34,13] that aims at extracting instances of a fixed set of predefined binary relations.

A pattern in the most abstract sense is a recurring construct, e.g. a word phrasing or a tabular representation, expressing the abstract relations textually. For instance, consider the text *"Goethe, who was born in Frankfurt, is one of the most famous German writers."*. It contains an instance of the textual pattern *"$\underline{X}$, who was born in $\underline{Y}$"*. Assuming the system knows that all instances of this pattern express an abstract `bornIn` relation, it can derive a matching relation instance asserting that Goethe was born in Frankfurt. In learning relation instances and the links between patterns and relations the system follows an iterative approach. Using type information on relations and entities, the system learns from given example relation instances which patterns represent which of these relations by analyzing how well the instances of an observed textual pattern match the given instances of individual relations. Once a link between pattern and relation is established, these patterns can be applied to extract more relation instances, which provides the system with more examples to learn more patterns from and so on.

The approach has the advantage that the actual patterns and their meaning can be learned on the basis of examples. Thus a user needs no detailed understanding of the extraction process, he only needs to deal with ontological knowledge in which he is interested anyway. As discussed earlier, the system needs some domain knowledge to function, namely 1) the entities of the domain, 2) the names of the entities, 3) the types of the entities and relations, 4) example relation instances and 5) as much other relational information on entities as possible. While a user may provide all this information upfront, we find that users are typically more motivated to provide this information in an interactive way, as they can observe the impact of their feedback efforts. This also ensures only information needed by the extraction system is provided. Thus, in the following section we shall briefly discuss the interaction between a user and the extraction system and especially which kind of information he can provide during the extraction process in the form of feedback.

## 5.2 Interface Interaction

Some domain specific background knowledge needs to be provided beforehand, in particular the type hierarchy, the relations of interest (with their range and domain types), and at least a basic set of entities along with some of their reference names. Additionally, the more relation instances are provided the more efficient the system can work.

Once this basic domain knowledge is provided, an interactive workflow allows to grow the knowledge base in a semi-automatic fashion.

First a set of relevant files is selected. The automatic extraction system searches for relevant information in the selected text basis and afterwards the user can inspect the recognized entity and relation instance occurrences and directly provide feedback on these occurrences. In particular the user can:

- correct referenced entities
- correct relations expressed between two entities
- add new entity references
- add new relation instance occurrences

In each of the given cases, the system implicitly learns from these corrections, e.g. by adding new reference names for an entity when an entity is corrected or directly derive relation instances when they are added or corrected. The accumulated knowledge can be applied when the extraction is re-run on the same or on a different data set. Coming back to the above-mentioned example the user needs only to indicate once that *"Frankfurt"* in a text does indeed reference *Frankfurt on the*

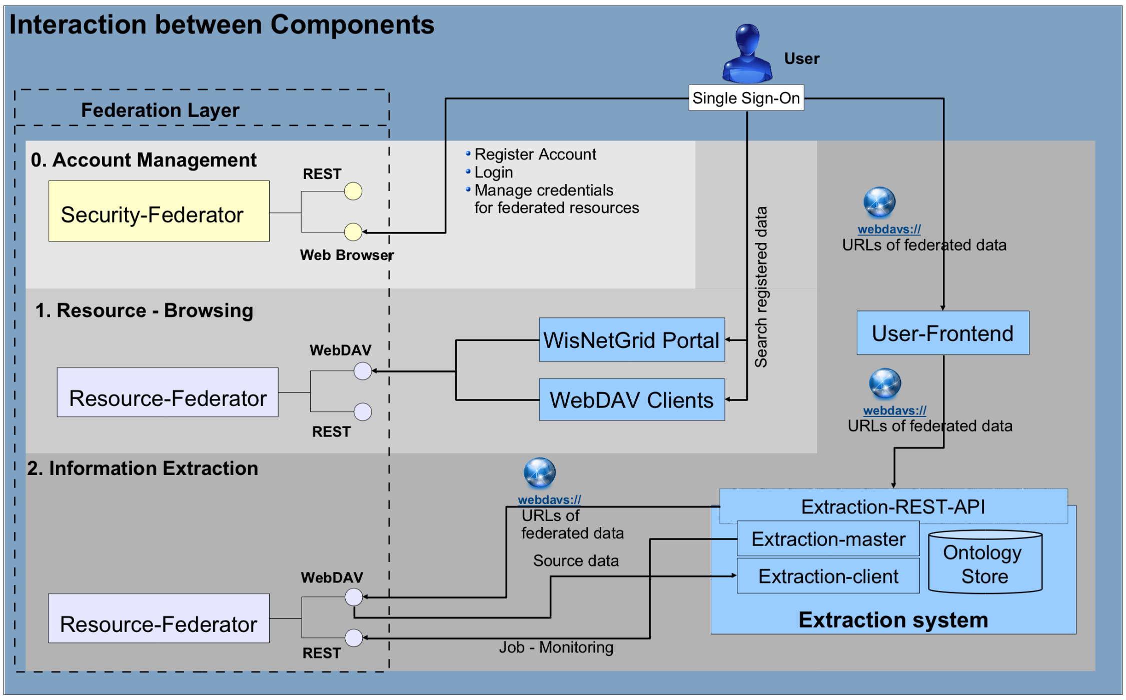

**Fig. 6** Interface interaction of the extraction system with the resource and security federation layer

*Main*, and the engine will very likely get it right in the whole text.

## 5.3 Integration into the Resource Federation Architecture

In sections 3 and 4 we explained how the federation layer allows to access distributed data sources. By invoking a Single Sign-On (SSO) client within software components, services can be realized that interact with the system on behalf of the user. We have realized the previously described interactive knowledge extraction service to extract knowledge from accessible federated data using this method. The interaction of the extraction system with the federation layer is illustrated in Fig. 6. After the user has registered the relevant credentials for the data sources in the SSO infrastructure using a Web Browser interface, the extraction system can access data sources using the central authentication and authorization provided by the Single Sign-On REST interface. All the user needs to provide are his SSO credentials and WebDAV URLs in the uniform name-space of the federation layer.

The extraction system can also make use of available computation power to distribute the extraction process over the unified job submission interface. To this end the system is split up in two main components, a master and a client part. While the master unit controls the distribution, the clients are executed as distributed jobs to parse all documents. Both parts access a central ontology store that manages all knowledge, either extracted or provided by users. This requires that the client is installed on the Grid nodes and the resource federation interface for job control is implemented for the particular Grid engine. After each iteration the extraction master presents the results to the user, who can provide feedback and re-run the extraction machinery so it can take the new feedback into account.

For the interaction with the extraction system a simple Web-based user interface is provided on top of a Web service API allowing the implementation of more sophisticated front-ends, e.g. enabling integration into a particular workflow environment of any community.

## 6 Conclusion

This paper describes an architecture providing a uniform access layer for different resources like Grid data management systems or databases. Therefore, a combination of Web server, routing engine and WebDAV environment generates a uniform namespace. The routing engine processes the client requests, forwarded by the Web server, in a defined route and directs them to the WebDAV environment. There, the requests are processed by interacting with a resource-dependent connector that mediates between the resource and the WebDAV server and the results are returned in a WebDAV-compliant response to the requesting entity. This concept is not restricted to data management systems, but also supports computing resources. An example based on the UNICORE Grid computing middleware was described in section 3.3.

We achieved our goal to provide a scalable and highly interoperable security system for use cases related to our resource federation system. The security model, realized by the Single Sign-On server and client components outlined in section 4.2, can be used by any service provider to access federated resources in a Single Sign-On manner, allowing for both Web browser and RESTful Web service access. This flexibility increases usability and paves the way for running computational studies on large interdisciplinary data sets as intended in section 1.

From a security standpoint, future work should be invested in adopting the findings outlined in [33], most of all the channel binding proposed in RFC 5929 as well as cross-domain SLSOP authentication cookies.

Both the resource federation system and the underlying security model form the basis to enable cross-organizational information extraction among distributed resources, which also benefits from uniform access to available computation resources. In addition to the uniform access achieved by the federation layer, an extraction system can provide a unified view on documents of different times, writing styles and potentially also languages by providing semantic meta-information.

Also, overall scalability must be put to the test in future work to gather significant and reliable performance data.

**Acknowledgements** The architecture described in this work is being developed as part of the German D-Grid project WisNetGrid (http://wisnetgrid.org) and is funded by the German Federal Ministry of Education and Research (BMBF).

## References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: ISWC/ASWC, pp. 11–15 (2007)
2. Barton, T., Basney, J., Freeman, T., Scavo, T., Siebenlist, F., Welch, V., Ananthakrishnan, R., Baker, B., Goode, M., Keahey, K.: Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy. In: 5th Annual PKI R&D Workshop (2006)
3. Benedyczak, K., Baa, P., van den Berghe, S., Menday, R., Schuller, B.: Key aspects of the unicore 6 security model. Future Generation Computer Systems **27**(2), 195 – 201 (2011). DOI 10.1016/j.future.2010.08.009
4. Berry, W.: 15 seconds : Sharing cookies across domains. URL `http://www.15seconds.com/issue/971108.htm`
5. Brin, S.: Extracting Patterns and Relations from the World Wide Web. In: WebDB, pp. 172–183 (1999)
6. Bryant, R.: Data-intensive scalable computing for scientific applications. Computing in Science Engineering **13**(6), 25 –33 (2011). DOI 10.1109/MCSE.2011.73
7. Cannataro, M., Talia, D., Srimani, P.K.: Parallel data intensive computing in scientific and commercial applications. Parallel Computing **28**(5), 673 – 704 (2002). DOI 10.1016/S0167-8191(02)00091-1
8. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile (2008). URL `http://tools.ietf.org/html/rfc5280`
9. Crockford, D.: The application/json Media Type for JavaScript Object Notation (JSON) (2006). URL `http://tools.ietf.org/html/rfc4627`
10. D-Grid: The German Grid Initiative. URL `http://www.d-grid-gmbh.de/index.php?id=1\&L=1`
11. EGI: European Grid Infrastructure. URL `http://www.egi.eu/`
12. EGI: Science Gateways. URL `http://www.egi.eu/services/support/science-gateways/`
13. Elbassuoni, S., Hose, K., Metzger, S., Schenkel, R.: Roxxi: Reviving witness dOcuments to eXplore eXtracted Information. In: Proceedings of the 36th International Conference on Very Large Data Bases, *Proceedings of the VLDB Endowment*, vol. 3, pp. 1589–1592. ACM, Singapore (2010)
14. EMI: European Middleware Initiative. URL `http://www.eu-emi.eu`
15. Farkas, Z., Kacsuk, P.: P-grade portal: A generic workflow system to support user communities. Future Generation Computer Systems **27**(5), 454 – 465 (2011). DOI 10.1016/j.future.2010.12.001
16. Fielding, R., Taylor, R.: Principled design of the modern web architecture. In: Software Engineering, 2000. Proceedings of the 2000 International Conference on, pp. 407 –416 (2000). DOI 10.1109/ICSE.2000.870431
17. Foster, I., Kesselman, C. (eds.): The grid: blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
18. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, 2008. GCE '08, pp. 1 –10 (2008). DOI 10.1109/GCE.2008.4738445
19. Gesing, S., Grunzke, R., Balasko, A., Birkenheuer, G., Blunk, D., Breuers, S., Brinkmann, A., Fels, G., Herres-Pawlis, S., Kacsuk, P., Kozlovszky, M., Krger, J., Packschies, L., Schfer, P., Schuller, B., Schuster, J., Steinke, T., Szikszay Fabri, A., Wewior, M., Mller-Pfefferkorn, R., Kohlbacher, O.: Granular security for a science gateway in structural bioinformatics. In: 3rd International Workshop on Science Gateways for Life Sciences (IWSG 2011), *CEUR Workshop Proceedings*, vol. 819 (2011). URL `http://ceur-ws.org/Vol-819/`
20. Guo, Z., Singh, R., Pierce, M.: Building the PolarGrid portal using Web 2.0 and OpenSocial. In: Proceedings of the 5th Grid Computing Environments Workshop, GCE '09, pp. 5:1–5:8. ACM, New York, NY, USA (2009). DOI 10.1145/1658260.1658267
21. Internet2 Middleware Initiative: Shibboleth. URL `http://shibboleth.internet2.edu/`
22. Jie, W., Arshad, J., Sinnott, R., Townend, P., Lei, Z.: A review of grid authentication and authorization technologies and support for federated access control. ACM Comput. Surv. **43**, 12:1–12:26 (2011). DOI http://doi.acm.org/10.1145/1883612.1883619

23. Kouzes, R., Anderson, G., Elbert, S., Gorton, I., Gracio, D.: The changing paradigm of data-intensive computing. Computer **42**(1), 26 –34 (2009). DOI 10.1109/MC.2009.26
24. Kulkarni, S., Singh, A., Ramakrishnan, G., Chakrabarti, S.: Collective annotation of Wikipedia entities in web text. In: KDD, pp. 457–466 (2009). DOI http://doi.acm.org/10.1145/1557019.1557073
25. MosGrid: Molecular Simulation Grid. URL `https://mosgrid.de/portal`
26. Novotny, J., Tuecke, S., Welch, V.: An online credential repository for the Grid: MyProxy. In: High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on, pp. 104 –111 (2001). DOI 10.1109/HPDC.2001.945181
27. NSF Middleware Initiative: GridShib. URL `http://gridshib.globus.org`
28. NSF Middleware Initiative: GridShib SAML Tools (2008). URL `http://gridshib.globus.org/docs/gridshib-saml-tools-0.5.0/readme.html`
29. OASIS Security Services TC: Security Assertion Markup Language (SAML) v2.0 (2005). URL `http://www.oasis-open.org/standards\#samlv2.0`
30. Plantikow, S., Peter, K., Hgqvist, M., Grimme, C., Papaspyrou, A.: Generalizing the data management of three community grids. Future Generation Computer Systems **25**(3), 281 – 289 (2009). DOI 10.1016/j.future.2008.05.001
31. PubMed: PubMed. URL `http://www.ncbi.nlm.nih.gov/pubmed/`
32. Recordon, David and Fitzpatrick, Brad: OpenID Authentification 1.1 (2006). URL `http://openid.net/specs/openid-authentication-1_1.html`
33. Schwenk, J., Kohlar, F., Amon, M.: The power of recognition: secure single sign-on using TLS channel bindings. In: Proceedings of the 7th ACM workshop on Digital identity management, DIM '11, pp. 63–72. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/2046642.2046656
34. Suchanek, F.M., Sozio, M., Weikum, G.: SOFIE: A Self-Organizing Framework for Information Extraction. In: WWW (2009)
35. Tuecke, S., Engert, D., Foster, I., Welch, V., Chicago, U., Thompson, M., Pearlman, L., Kesselman, C.: Internet X.509 Public Key Infrastructure Proxy Certificate Profile (2001). Revised July 2002
36. United Nations Environment Programme: Environmental Data Explorer . URL `http://geodata.grid.unep.ch/`
37. Venugopal, S., Buyya, R., Ramamohanarao, K.: A taxonomy of data grids for distributed data sharing, management, and processing. ACM Comput. Surv. **38** (2006). DOI http://doi.acm.org/http://doi.acm.org/10.1145/1132952.1132955
38. W3C Working Group: libcurl - the multiprotocol file transfer library (2004). URL `http://www.w3.org/TR/ws-arch/`
39. W3C Working Group: Web Services Architecture (2004). URL `http://www.w3.org/TR/ws-arch/`
40. WisNetGrid: WisNetGrid – Knowledge Networks for Grids. URL `http://wisnetgrid.org`. Grid Project within the German Grid Initiative (D-Grid)
41. XSede: Extreme Science and Engeneering Discovery Environment. URL `https://www.xsede.org/home`
42. XSede: Science Gateways via User Portal. URL `https://www.xsede.org/science-gateways`
43. Yosef, M.A., Hoffart, J., Spaniol, M., Weikum, G.: Aida: An online tool for accurate disambiguation of named entities in text and tables. In: H.V. Jagadish, J. Blakeley, J.M. Hellerstein, N. Koudas, W. Lehner, S. Sarawagi, U. Röhm (eds.) Proceedings of the 37th International Conference on Very Large Data Bases, *Proceedings of the VLDB Endowment*, vol. 4, pp. 1450–1453. VLDB Endowment, Seattle, USA (2011)