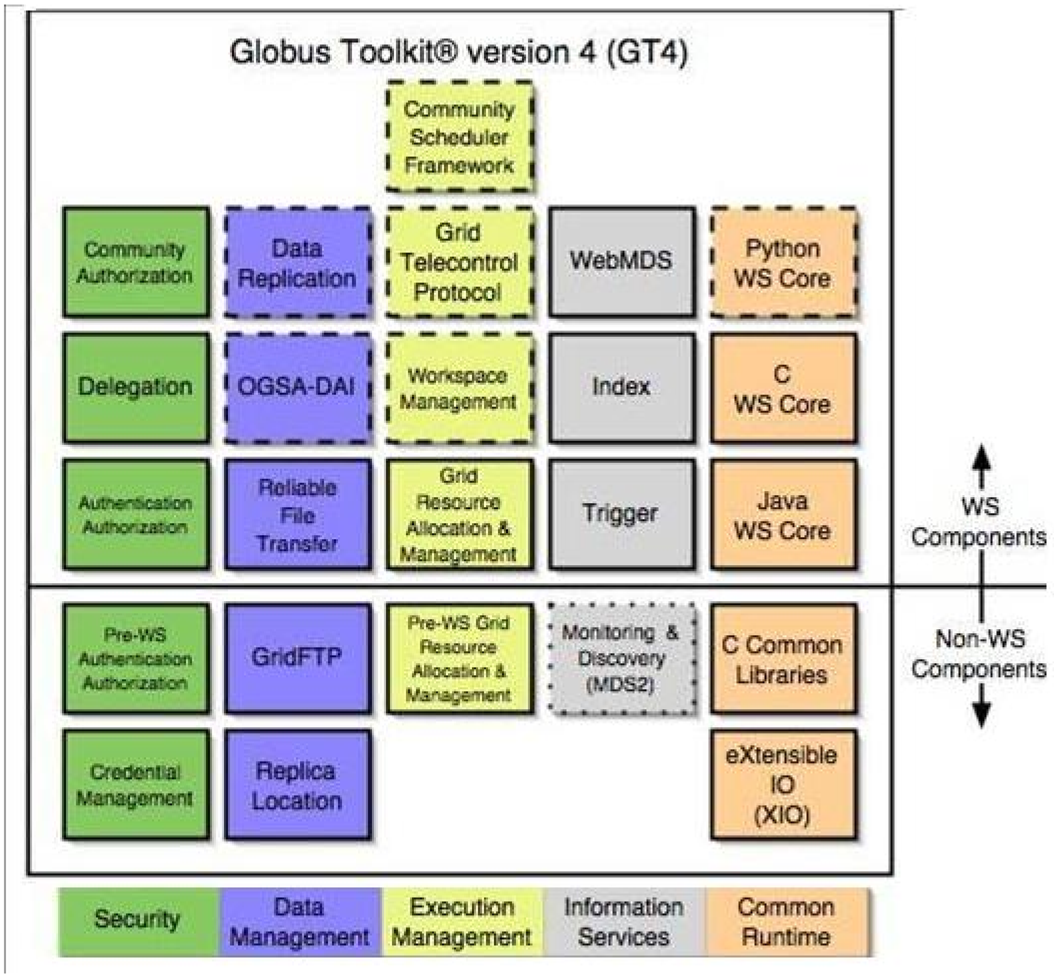
Globus Toolkit Java Security Components

Rachana Ananthakrishnan, ranantha@mcs.anl.gov
Frank Siebenlist, franks@mcs.anl.gov

The Globus Toolkit security components comprise Web service-based security infrastructure implementations in both Java and C and a set of higher-level security services including the Delegation Service and Community Authorization Service. This handout provides an overview of the supported features and highlights the new features.



The main authentication mechanism is based on a Public Key Infrastructure (PKI) enhanced with extensions such as proxy certificates and delegation of rights. The toolkit's Java WS **authentication framework** facilitates secure communication by leveraging SSL/TLS, WS-Security, WS-Trust, and WS Secure Conversation and is compliant with the Web Services Security 1.0 Specification. It supports transport-level and secure conversation as well as message-level security to provide for privacy, integrity, and replay attack protection.

The GT 4.0 distribution includes a pluggable server-side **authorization framework,** implemented in Java that can be configured with various custom Policy Information Points (PIPs) to collect attributes and Policy Decision Points (PDPs) to evaluate the policy. The framework evaluates the PDPs in the order configured using a deny-override algorithm to determine whether a client can invoke an operation on the resource. Also provided is a client-side authorization framework that allows for configuring a single authorization scheme to authorize the server.

The upcoming GT 4.2 release includes a **new, sophisticated authorization framework** that allows plugging in of different attribute-based evaluation engines. This facilitates the enforcement of complex policies, including the delegation of rights. The new framework is distributed with a default engine that

Rachana Ananthakrishnan, ranantha@mcs.anl.gov
Frank Siebenlist, franks@mcs.anl.gov

attempts to construct a delegation chain from the resource owner to the access requestor to establish whether access can be granted. Further, the engine differentiates between access rights, which allow an entity to access a resource and administration rights, which allow an entity to delegate its access rights to another.

The Java WS Security framework provides both **declarative and programmatic security**. That is, security properties can be configured just as configuration files, called security descriptors, without modifying any code. Alternatively, the properties can be incorporated as part of the source code.

On the server side, security properties can be configured as resource-, service-, and container-level granularity; the properties are picked up in that order of precedence. Typical server side security properties include the credentials to use, the authentication mechanism required to be used by the client, the authorization mechanism to use to authorize client access, and the credential to associate with the current thread. Similar declarative or programmatic security can be used on client side to configure properties such as credentials to use, the authentication scheme to use, and the authorization scheme to use to authorize the server.

The **Delegation Service** allows for the delegation of user rights to a remote resource independent of the communication protocol. Its implementation conforms to the WS-Trust specification. The service is based on the Web Services Resource Framework (WSRF) specification, with each delegated credential being a resource, and allows delegation of credentials to a hosting environment. Delegated credentials can be shared across multiple invocations and multiple services in a hosting environment. Also, the service retains the delegated credentials on disk and enables the user who delegated the credential to refresh the credential.

The **Community Authorization Service** allows for fine-grained authorization policy management for distributed resources. The authorization request protocol is based on the Simple Assertion Markup Language (SAML) specification, and the service is constructed as a WSRF-based service, while the actual policy is stored in a backend database. CAS supports both a pull model, where the resource can pull down the policy decisions from the server, and a push model, where the client can contact server to get the assertions from the CAS server and send it to the resource. The CAS service implements the GGF OGSA-AuthZ Authorization Service (using SAML OGSA Authorization, Global Grid ForumGFD.066; http://www.ggf.org/documents/GFD.66.pdf) specification to provide the pull interface. This specification allows the CAS service to be queried for specific policy decisions for resource access.

The toolkit provides callouts to use the CAS service for authorization of file access using the GridFTP component and authorization of any resource in the Java WS Core framework. The **GridFTP authorization using CAS** uses the push model, where the client pushes assertion from the CAS server in its request to the GridFTP server. The assertion can be embedded in the proxy certificate used by the client and also can be cached across multiple sessions. The server processes the assertion to evaluate and determine access to file.

In the 4.2 version of the Globus Toolkit, the Java WS authorization framework provides for pluggable PDPs and PIPs that allow for using **CAS for Web service policy management**. In order to support pull model, a PIP is provided that extracts the assertions from a client's proxy or the message header. A PDP then evaluates the extracted assertion, and the authorization decision processes it to provide a decision. In order to support the push model, a generic authorization callout is provided that talks to any OGSA AuthZ-compliant authorization service. The callout can be used to talk to a configured CAS service to pull down assertions about a particular resource access.

The Portal-based User Registration Service provides a set of tools for automating user registration and credential management, especially for portal-based systems. It provides a set of customizable components that can be used to manage the full lifecycle of Grid-based credential management. The tools provide a backend API to register a new user, issue credentials for the user, manage registered users, renew credentials for users, and so on. The tools interface with a backend database that stores the user details.

The Globus Toolkit is an open source project, and community participation is encouraged and welcomed. For more information on the toolkit and ways to contribute, please see http://dev.globus.org/ .