



# **An Evaluation of the CROWN Middleware for the Engineering Task Force**

**The UK Grid Engineering Task Force**

Tim Parkinson, Southampton e-Science Centre  
Mark Hewitt, North-East Regional e-Science Centre  
David McBride, London e-Science Centre

**30-04-2007**

## **Abstract:**

CROWN 1.0 is a Web Service hosting environment based on an earlier release of the Globus Toolkit. It allows services to be deployed, undeployed, and redeployed remotely without container restart (Hot Deployment). It contains a security system that allows fine grained authentication and authorisation control at both Node and Service levels. CROWN nodes can be federated into a Grid. There is also an Eclipse plug-in development tool.

The evaluation team installed the software on a variety of machines and successfully tested remote deployment, interoperability, and access control.

In general we conclude that CROWN is suitable for deployment on NGS machines.

## **UK e-Science Technical Report Series**

**Report UKeS-2007-02**

Available from [http://www.nesc.ac.uk/technical\\_papers/UKeS-2007-02.pdf](http://www.nesc.ac.uk/technical_papers/UKeS-2007-02.pdf)

Copyright © 2007 The University of Edinburgh. All rights reserved.

# An Evaluation of the CROWN Middleware for the Engineering Task Force

Tim Parkinson (SeSC), Mark Hewitt (NEReSC), David McBride (LeSC)

Version: 1.0 – April 2007

## Abstract

CROWN 1.0 is a Web Service hosting environment based on an earlier release of the Globus Toolkit. It allows services to be deployed, undeployed, and redeployed remotely without container restart (Hot Deployment). It contains a security system that allows fine grained authentication and authorisation control at both Node and Service levels. CROWN nodes can be federated into a Grid. There is also an Eclipse plug-in development tool.

The evaluation team installed the software on a variety of machines and successfully tested remote deployment, interoperability, and access control.

In general we conclude that CROWN is suitable for deployment on NGS machines.

## 1 Introduction

The UK's Engineering Task Force (ETF) is evaluating several Grid middleware solutions in order to better understand their deployability on the resources within the National Grid Service (NGS) and those of the wider UK e-Science community. This report presents the results of the evaluation of the CROWN middleware system from Beihang University, China.

While this evaluation will provide input into the NGS deployment, its main purpose is to examine the strengths and weaknesses of the CROWN system and identify the issues that would need to be considered before deployment in a production environment.

## 2 General Information

The CROWN (China Research & Development Environment Over Wide-area Network) middleware is a Web services based toolkit. Release 1.0 is based on version 3.9.3 of the Globus Core (but see Appendix for CROWN 2.0 details) which allows hot deployment of a Web service on a remote resource also running the CROWN software. Once this Web service has been deployed it can then be used by other CROWN hosted services or by CROWN client programs that meet the service level and node level authentication and authorisation criteria.

This evaluation report refers only to Release 1.0 plus its patches up to 22<sup>nd</sup> July 2005. Release 1.0 consists of seven components:

1. The **CROWN NodeServer**. This is the basic service container. All CROWN services must be invoked via a NodeServer. As well as service invocation, the NodeServer provides remote and hot service deployment, undeployment, and redeployment. A NodeServer instance must be installed and run on each server or cluster head node that wants to join the Grid.
2. **CROWN Resource Location and Description Service (RLDS)**. This component is a Grid Information Service that provides a service registry in which NodeServers register their services and from which client applications can discover those services. Each

RLDS also maintains a record of the system information for its NodeServers including number of processors, processor type, memory size, and current load. The RLDS instances are queried using a language called GIQL (Grid Information Query Language) which is “SQL like”.

At least one machine on the Grid must run an instance of RLDS in addition to the NodeServer. All NodeServers must register with either a local or a remote RLDS. Each RLDS either registers itself with a parent or acts as a standalone instance thus forming a forest topology.

Each RLDS instance requires a local or remote MySQL database instance for persistent storage.

All RLDS instances in the forest must have access to a NodeServer that hosts an instance of the GIMS service (Grid Information Model Service). The NodeServer that hosts the GIMS instance does not have to be part of the RLDS forest.

The RLDS persistent storage schema is initialised and updated by invoking GIMS at each NodeServer start-up.

In a simple topology, all RLDS instances use the same model from a single GIMS instance that may run on a machine outside the forest. More complex topologies may have different trees within the forest using different data models from different GIMS services.

3. **CROWN Security (and Trust).** A Secure NodeServer consists of a standard Node Server with the CROWN Security and Trust (ST). CROWN-ST adds support for message level integrity, message level confidentiality, authentication, and authorisation.

Message integrity and confidentiality are highly configurable with a large choice of digest and encryption algorithms which can be specified at service and node level. Service and application designers can choose between locally executed mechanisms and callouts to remote signing and encryption services.

All team members performed an initial installation of NodeServer without Security in order to verify the functionality.

Subsequently all members installed the Security and Trust components and from then on we only used Secure NodeServers. So for the rest of the document, the term **NodeServer** is synonymous with CROWN NodeServer plus CROWN Security and Trust.

4. **CROWN Designer** is a development tool that is shipped as an Eclipse plug-in. It allows Java programmers to turn Java classes into Web Services that can be packaged and deployed to CROWN servers from within the Eclipse IDE. Other tools allow the developer to query the state and make-up of a Grid.
5. **CROWN Client Libraries.** The server distribution contains a large number of libraries. A necessary and sufficient subset of those libraries is shipped as a client development library. This component was not evaluated by the team.
6. **CROWN Portal** is a Web portal front end that allows registered users to initiate and monitor service based jobs that are hosted by the Grid. It acts as a shop window and can be used for branding. Registered users have a single sign-on way of authenticating for the use of services, even though they may not be authorised to use them. This component was not evaluated by the team.

7. **CROWN Portal Scheduler** runs below the CROWN Portal and provides the scheduling and monitoring of jobs. This component was not evaluated by the team.

The relationship between the components is pictured in Figure 1 below. The MySQL per RLDS instance is not shown, nor is the GIMS service(s) from which all the RLDS instances obtain their data models.

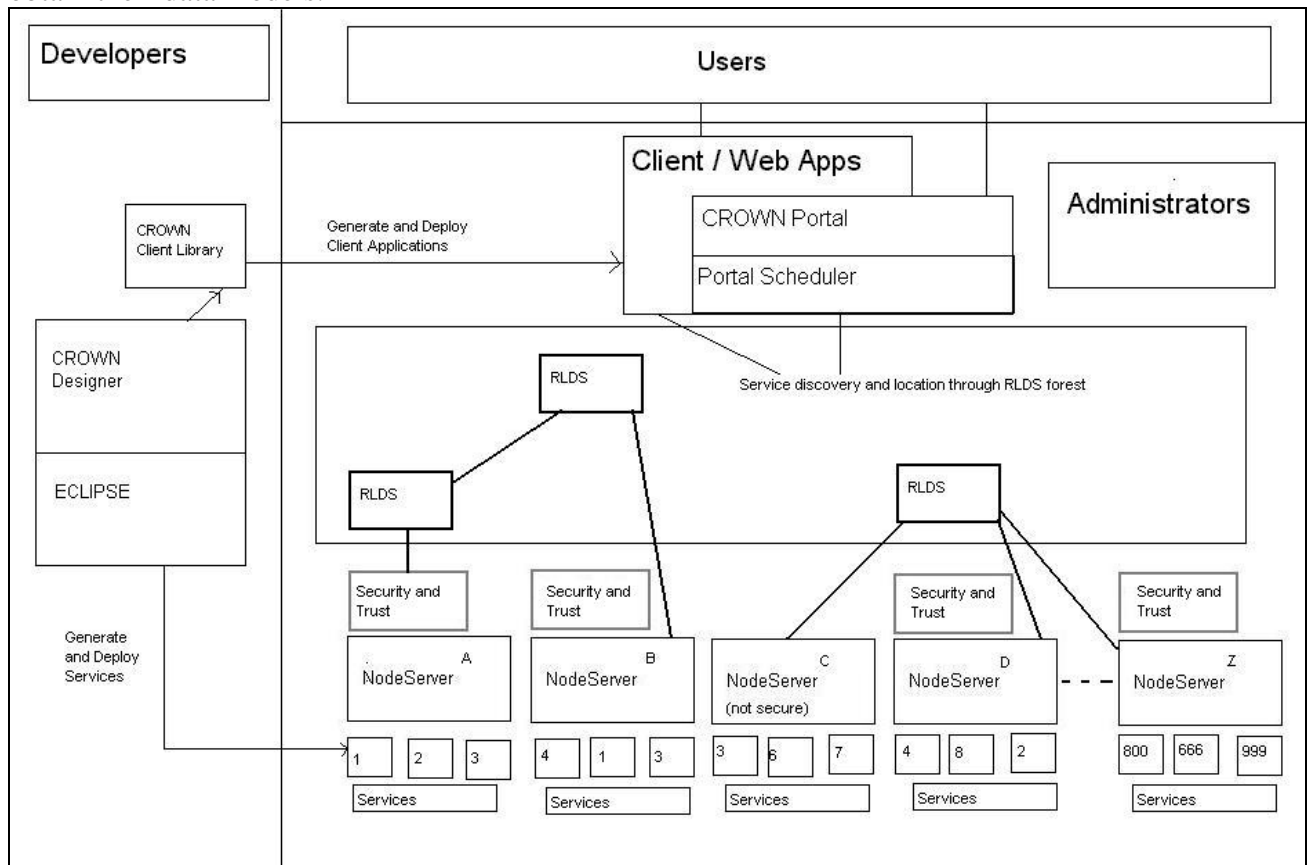


Figure 1

We only installed and evaluated four of these components, NodeServer, RLDS, Security, and Designer. The other three, Portal, Portal Scheduler, and Client, were not evaluated because the main objectives were to investigate the hot-deployment and security mechanisms.

In December 2005, CROWN 2.0 was released. The evaluation team have neither downloaded nor installed this new release.

## 2.1 Provider

The software is provided by the CROWN research group in the Institute of Advanced Computing Technology, Beihang University, Beijing, China under the supervision of Prof Jinpeng Huai. Beihang's development process is not publicly documented so we cannot comment on it.

The CROWN project is jointly-funded by the National Natural Science Foundation for China (NFSC at [www.nsf.gov.cn](http://www.nsf.gov.cn)) and the 863 Hi-Tech R&D Programme of the Ministry of Science and Technology ([www.863.org.cn](http://www.863.org.cn)). The 863 funding covers some of the project activity from July 2005 to July 2006. The NFSC grant is for four years from 2004 to 2007.

CROWN has secured longer term funding from the National 973 grant programme until 2009.

The installed base is the CNGRID National Test bed for Grid Software spread over eight campus and institute sites in cities throughout China. Several services are offered through the Beihang CROWN portal which has over three hundred registered users.

In the UK, CROWN has been installed at the University of Leeds. Work is being carried out under the direction of Prof Jie Xu to deploy computational medical remote visualisation and healthcare planning services that can be accessed by both White Rose Grid and CNGRID users (see gViz at <http://colab.crown.org.cn>).

## 2.2 Licensing

The CROWN software is distributed under the Apache Software Foundation licence version 2.0 with no more and no less limitation and restrictions than the Apache Licence imposes. This licence has been certified as an Open Source licence by the Open Source Foundation and is considered DFSG-Free by the Debian Project.

CROWN redistributes the Globus Toolkit version 3.9.3. Since this release of GT pre-dates the adoption of the Apache Licence by Globus part of the CROWN distribution is still subject to the Globus Licence.

There is no LICENCE.txt type file in the distribution nor any licence included in the documentation, however there is a small 'License' link at the bottom of the CROWN Portal Website [www.crown.org.cn/en](http://www.crown.org.cn/en).

## 2.3 Supported Platforms and Evaluation Testbed

The server bundle is largely pure Java and can run on any platform that has an implementation of the 1.4.2 or later JDK. The developers recommend the Sun JDK. The Chinese documentation has both Windows and Linux installation instructions. The CROWN Designer Eclipse IDE plug-in component requires Eclipse 3.0.1 or greater. However we found that the plug-in was not suitable for Eclipse 3.1 series and the developers have now rectified this in a later release (see Appendix 8.2 [Eclipse Plug-in Support](#), for details).

All the servers in our evaluation were 32 bit Intel Linux with Sun JDK 1.4.2 except one machine which was a 64-bit AMD processor. The 64-bit machine initially ran the server under a 64-bit Sun Java 5 JDK with one minor configuration change and without platform dependent problems, but later switched to a 32-bit 1.4.2 JDK.

Some client development work was done on a Windows XP desktop machine.

The evaluation team installed the secure server configuration on one machine at each site as follows:

- *Imperial College*, a dual 2.0Ghz Intel Pentium 4 Xeon with 1024MB RAM running Red Hat Linux 7.3 from behind a minimal firewall.
- *Southampton*, an AMD Opteron 64 bit dual 2.1GHz processor server with 2GB memory running Red Hat Enterprise Linux 3 update 3 behind the campus firewall allowing incoming TCP connections only on port 28080
- *Newcastle*, an Intel 32 bit server with a single 1GHz processor and 1GB RAM running SuSE 9.0 with restrictions on the use of ports 80 and 8080.

- *Beijing*, from time to time we connected up to a large 64-bit multi-processor Linux machine at Beihang University.

## **2.4 Support**

The software is currently supported directly by the CROWN developers who have been very responsive to our support requests. Emails are answered promptly, allowing for time zone and language differences, and they have regularly taken part in teleconference meetings. There is no commercial support available. The CROWN Portal [1] has a *User Center*. Registered users can obtain free support from the developers. The developers do not have specific specialised support roles. The authors of the individual components tend to respond to queries in their own area of expertise.

We are not aware of community support. Any such community support would undoubtedly originate from the installed base in China with the consequent language and time zone gaps. There is a pocket of expertise at Leeds University but no formal BBS or mailing list.

## **3 Systems Management**

We consider here the issues that would impact on the administrators of the particular software product and on the administrators of the resources upon which the product will be deployed.

### **3.1 Documentation for System Managers**

The documentation consists of three English translations of original Chinese documents in PDF format: an Overview; an Installation and Admin Guide; and a User Guide. The English version of the User Guide incorporates translations of the separate Chinese documents that describe the authentication and authorisation service. None of these PDFs are indexed but keyword search is possible via Acrobat Reader's Search facility.

Installation documentation is available via a series of supplementary Microsoft Word documents including a Minimal Steps (Quick Start) Guide. These were very comprehensive and went through the installation process step by step detailing each task which needed to be completed. A few typos were quickly identified by the evaluation team and were confirmed and corrected by the authors overnight.

### **3.2 Server Deployment**

#### **3.2.1 Server Installation and Configuration**

No changes were needed to the installation environment. The CROWN software was easy to install in its default configuration with minimum changes.

The Secure NodeServer only requires a JDK 1.4.2. If local versions of RLDS or GIMS are to be run, then a running MySQL is necessary (either on the local machine or accessible over the network). Otherwise the installation is self-contained. The well known Xalan bug that affects some Sun JDK's is worked around by CROWN shipping a correct set of Xalan libraries in a local 'endorsed' directory so there is no dependency on a particular build of Java 1.4.2.

The team ran the Secure NodeServer on a mixture of SuSE 9.0, Red Hat 7 and Red Hat Enterprise Linux without problems.

The NodeServer runs as a user process, normally under the *crown* userid (or other site-defined name). Only `JAVA_HOME` and `GLOBUS_LOCATION` environment variables need to be set plus `$JAVA_HOME/bin` on the `PATH`.

### 3.2.1.1 Administration Tasks

The main administration tasks are:

- a) Starting and stopping the server. This is performed by a simple command line script with a few options.

On Linux/Unix platforms the startup and shutdown commands themselves could easily be incorporated into a normal `/etc/init.d` script to fix these operations into the boot and shutdown sequence. Writing such a script would not be trivial as both the startup and shutdown process can take a minute or two and the script would have to wait for the operations to complete.

On a Windows Platform, running the NodeServer as a Windows Service would have to be configured by the system administrator, possibly with the aid of third party utilities such as the Tanuki Java Service Wrapper.

On both platforms the enhancement of the installation procedure to integrate these operations into the boot sequence would be a welcome improvement. (See [Appendix 8.3 Auto-start and auto-stop](#), for improvements in CROWN 2.0)

- b) Deploying, Undeploying, and Redeploying services. This is covered in greater detail in Section 3.2.2.
- c) Configuration of Authentication. Adding trust to (and removing trust from) a node or some of its services requires familiarity with X.509 certificates and knowledge of Java *keytool*. The administrator may also need to know how to use `openssl` or equivalent utilities to convert certificates from their delivery format into different formats, `.pem` in particular.

However, a NodeServer can be configured to use an authentication service from another node at a different site so the specialised knowledge can be concentrated on a single site. This requires all services on a node to either be pre-configured to reference the central authentication service prior to deployment or to have their policies modified with a text editor post-deployment, which risks the changes being overwritten on redeployment.

The Overview refers to authentication via Kerberos in one of its diagrams. There was no detailed documentation on how to achieve this and we did not attempt to investigate this feature further.

Modifying authorisation security policies for the node or some of its services, requires knowledge of the XACML Language [2] that CROWN uses to specify policies. The site administrator would need to learn this syntax and where to find the service level policy files. Like the authentication service, all the services on a group of NodeServers can reference a remote authorisation service, allowing this knowledge to be concentrated on one site.

### 3.2.1.2 Port Usage

The Node Server requires only one port open for incoming TCP connections and this is configurable on the server startup command line. The software as originally supplied performed all communication on port 8080, which is blocked by some sites, but the documentation to describe how to run on a different port has been added.

Once a server was running on a non-default port, all client scripts needed an extra command line option to specify the invoked service's entire endpoint URL including the new port.

A weakness is that all services deployed on the NodeServer that refer to other services on the same server, or indeed on other servers running on a non-default port, must have their configuration files modified to explicitly reference the new port number.

Southampton and Imperial have a formal procedure for opening ports to the greater Internet. This was followed without problems or challenge from the network administrators and was made very easy by the fact that only one port opening was required. In addition to the campus firewall, the Southampton server runs an `iptables` local firewall that only allows incoming connections on a few ports.

### 3.2.1.3 System Stability

The NodeServers at Southampton and Imperial running on 24x7 machines stayed up for weeks at a time. The only reason they ever terminated was by administrator action to restart the NodeServer or the machine.

## 3.2.2 Service Deployment

The simplest CROWN services are basic Web services. In order to be deployed, the Java classes, support jars, WSDL and other files must be packaged up in a special `tar` archive called a `gar` file, similar to the WAR files used to deploy Web applications. The GAR format is defined and described in the Globus Toolkit 4 documentation

### 3.2.2.1 Service Packaging

By far the easiest way to package a newly developed CROWN service is to develop it within Eclipse using CROWN Designer. The necessary directory structure will be created by the tool and when the developer is ready, use the `Make Gar` utility from the Project Context Menu. If the developer prefers not to use Eclipse and therefore cannot use CROWN Designer then it is still possible to create a deployable CROWN `gar` file by creating the deployment directory structure manually from the command line if using JWSDP or from NetBeans or other IDE, placing the class and WSDL files into the correct place, then using the normal `jar` command to package up the service.

A service developer that does not use an IDE may obtain the `ant makeGAR` extension task from the `.jars` in the CROWN lib directory or from the Globus 4.x toolkit.

### 3.2.2.2 Deployment

Among CROWN's particular unique features are Hot Deployment, Hot Undeployment, and Hot Redeployment, which refer respectively to the ability to add, remove, and update services without restarting the NodeServer. There are two ways to do this, Local and Remote.

Local deployment is typically used during development where a programmer has access to a personal copy of the NodeServer on their own machine. The only way to tell if the deployment succeeds is to check the tail of the log file.

If an administrator or developer has local login access to a NodeServer machine, secure or non-secure, then the quick way to deploy a service for the first time is to copy the service `gar` file to the special `auto-deploy` directory of the running server. The NodeServer will then detect the presence of the new service, unpack it, and register it with the governing RLDS instance. Likewise to undeploy a service, simply delete the `gar` file from the `auto-deploy` directory of the running server. The NodeServer will detect the change and will de-register the service and delete its files. To redeploy a service, copy the new `gar` file to the `auto-deploy` directory to overwrite the original. The NodeServer detects the change, de-registers and deletes the original service and deploys the new. All of these local methods were used by the evaluation team without problems.

By extension, if remote copy (via say `scp`, `rcp`, or even `ftp`) is available to NodeServer machines then remote deployments could be executed in this manner too. However, to allow such a pseudo-local deployment would be undesirable as it bypasses the CROWN authentication and authorisation methods that protect the true remote deployment service.

Remote deployment is the invocation of a NodeServer's remote deployment service, passing a `.gar` file.

CROWN ships with a command line tool called `deployclient.sh` that allows anyone to remotely deploy, undeploy, and redeploy `gar` files on any NodeServer provided that they have access and the authority to do so.

Essentially, the tool invokes the specified NodeServer's remote deployment service and the NodeServer authenticates the user's identity, checks the authority to deploy and then performs the copy or deletion of the `gar` file to or from the NodeServer's `auto-deploy` directory as described in the last section. The same operations can be carried out much more easily from menu options within CROWN Designer and these operations were also performed successfully by the evaluation team. These operations were all executed by the evaluation team without problems.

### **3.3 Client Deployment**

The server distribution contains some simple example / test scripts such as `HelloWorld` and the `CounterClient` scripts. To run these scripts on other machines required us to unpack the full server distribution to supply the necessary libraries. Then, provided that the machine had a Java 1.4.2 `JAVA_HOME` available and that the `GLOBUS_LOCATION` was set, the client program would work. The example scripts are POSIX shell scripts so their use is possible on Linux/Unix machines or Windows machines plus Cygwin.

As mentioned earlier, the Overview document mentions a Client library distribution but we did not evaluate it. The client distribution would contain sufficient libraries to run client programs without installing the whole server distribution as described above. A true standalone client script distribution would include the client libraries.

To invoke a client program, the server name, port, and path to the service endpoint must be specified to the scripts and the client machine must be able to connect out to the active port on the server machine and receive traffic back over that connection.

This requirement for local access to the server libraries and explicit port specification is not a criticism of the supplied examples. Clearly a more sophisticated client program that consumes real services would be less cumbersome and would be shipped with the minimal client libraries and would obtain service:port URL's through a Discovery process.

### **3.4 Account Management**

The default authentication operation of the Secure NodeServer is to trust X509 Certificates issued by CROWN. Site owners may change the keystores to trust other CA's. The management interface for modifying certificate trust is Java *keytool*. So any clean production installation that requires locked down access would have to remove the trust for CROWN certificates and replace it with trust for its own CA. (In the NGS case that would be the UK Grid CA.)

Authorisation – allow and deny policy – for the server or its deployed services is achieved through editing XACML policy files. The management interface for authorisation policy is a text editor plus the manager's knowledge of XACML. However these service level XACML files would be overwritten by a redeployment, so care should be taken to back up and re-implement local changes.

Virtual Organisations can be supported by writing arbitrarily complex XACML policy rules on a service by service basis. In principle, every distinct user of a CROWN Grid (i.e. every unique DN of a trusted certificate) could have different permissions for every service on every node. Policy rules are written using a text editor.

The XACML 1.0 and 2.0 specifications on the OASIS Web site contain functions for calculating times and dates within rules. So provided that CROWN has a full implementation of at least XACML 1.0 then date and time based access to services should be configurable.

The MySQL database attached to each RLDS instance accumulates tables of information about service invocation. However we have not found any tools to report on this information other than the RLDS's own GIQL language. The only way to export usage information is via a database GIQL query. There is no resource accounting or quota functionality currently included in CROWN 1.0. (But see in Appendix 8.4 CROWN Monitor for the change in CROWN 2.0)

### **3.5 Reliability**

The software has not been tested under high loads or failure conditions. However, when a service on one of our nodes was inadvertently relying on a service provided by one of the Beihang machines and that remote machine was taken down, our node's service failed mysteriously, as described in Section 4.7.

### **3.6 Distributed management**

The CROWN usage model is that a NodeServer acts as the gateway to applications offered on that particular machine or on a local cluster attached to that machine. An application can only be run by a client as a CROWN Web Service. The NodeServer offers its services to the wider community either by running its own Resource Location and Description Service

(RLDS) or by subordinating itself to another node's RLDS. RLDS services themselves can be federated in a forest topology. A new node that subordinates itself to another node's RLDS joins the Grid. The Node Server does not in itself offer an arbitrary execution service for arbitrary programs in the manner of the Globus GRAM or GridSAM JDSL services. (But see in Appendix 8.5 CROWN Scheduler supports Legacy Applications, for changes in this area for CROWN 2.0)

Provided that the hot deployment services on each node are configured to allow it, a centralised service manager could deploy, undeploy, or upgrade application services on the other nodes in the Grid.

The RLDS service allows query and monitoring (by repeated query) of the Grid status. An example of a raw system information query from one of the CROWN Designer panels is shown in Figure 2.

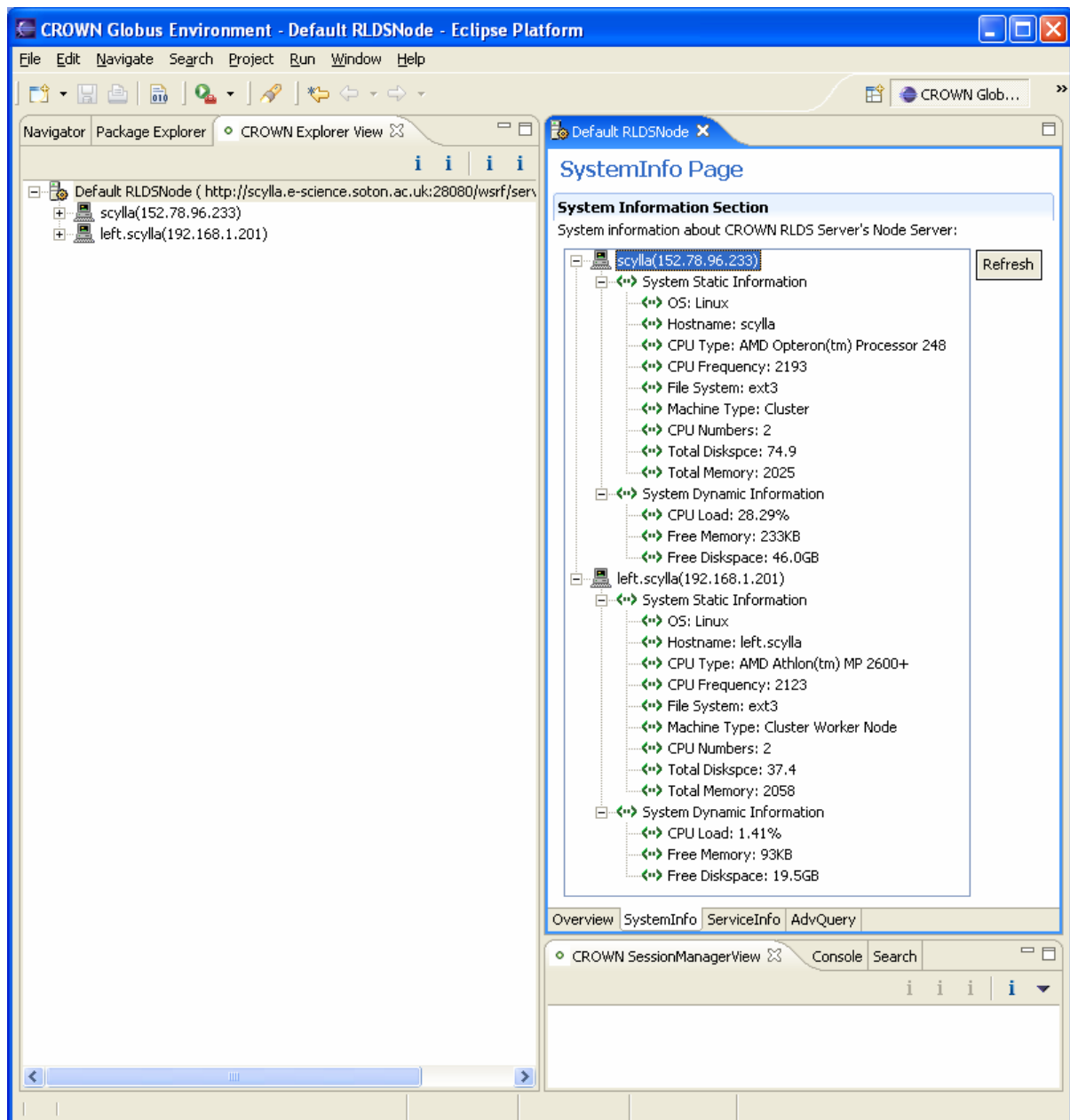


Figure 2

The Beihang CROWN Portal continuously displays the output of the CNGRID status. An example is shown in Figure 3.

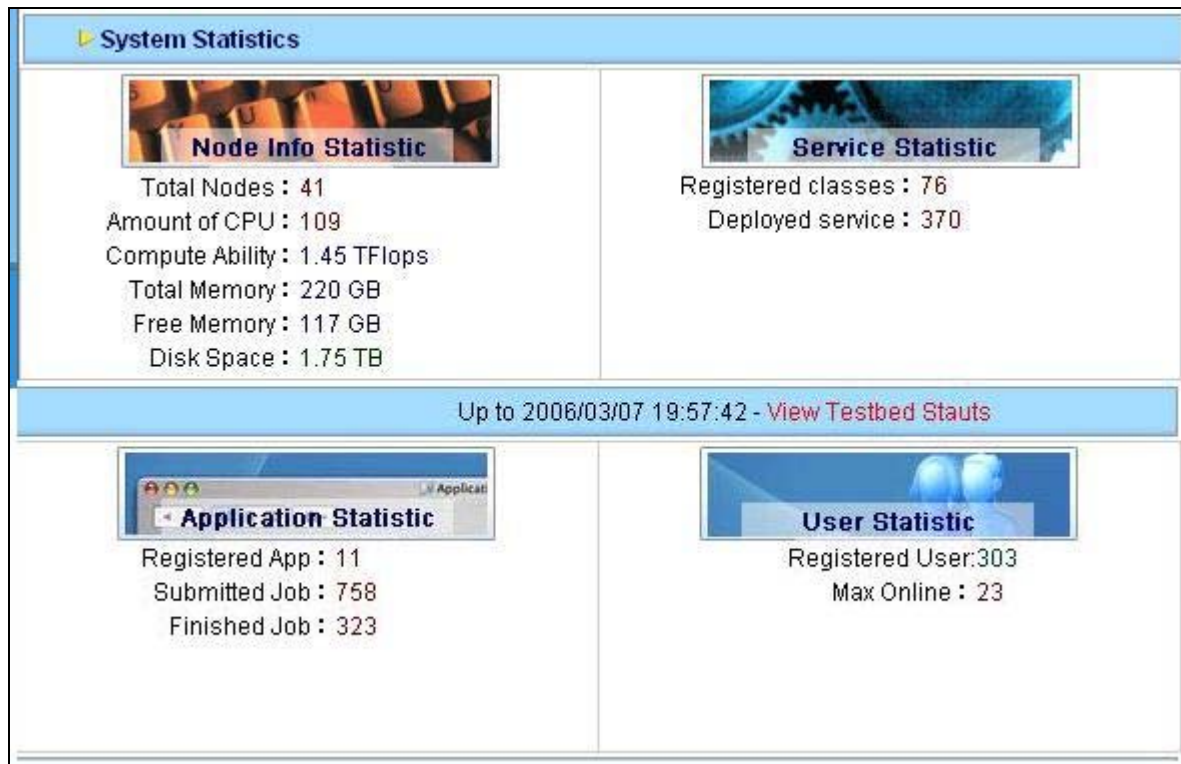


Figure 3: The screenshot was cut up and re-arranged for layout readability but the content is preserved

Service and Node configuration and policy can be configured either by specialised XML editors within CROWN Designer or by using a text editor. These methods cannot really be called management tools.

Upgrading the software was difficult, there appears to be no defined upgrade process apart from deleting the current installation and reinstalling a new installation from scratch. The version number attached to later versions of the software was not updated in some cases, which made it more difficult to track version changes. Also on one occasion after one of our nodes were patched, our node servers refused to communicate with each other and the situation was only resolved when all nodes re-installed NodeServers with an identical set of patches. This suggests that there may be some compatibility issues, although we did not establish the root cause of that problem

### 3.7 External Services

We believe that a third-party Web service could not be easily deployed into a CROWN Secure Node Server without extra coding and repackaging as a .gar.

To write a client program that uses CROWN based and other services and also requires third-party client libraries would be possible using the CROWN Designer tool within Eclipse. It would only be necessary to define the third party libraries as external libraries within the Eclipse project. The CROWN Client Libraries would also need to be packaged with existing client libraries before distribution.

Modifying existing client environments to use secure CROWN services would require re-engineering.

### **3.8 Scalability**

The machines can be arranged in a distributed topology. Each machine reports to an RLDS instance and this RLDS can itself report to another above it, thus a forest structure can be developed. The member nodes in this forest could be at arbitrary physical distances apart with consequent differing hop counts and transmission times.

The test environment only consisted of a handful of machines so we did not test the effect of numbers of machines. We did at one stage add a machine in Beijing to our test Grid. Initially we encountered difficulties that were traced to the increased latency of packet transmission between the UK and Beijing caused by more hops and a slow link along one of the hops. This shows that for any installation on the NGS machines, such connectivity should be taken into account. Connection to the JANET backbone should be more than adequate. The production Grid in China which is spread over a much larger area than the UK does not have these problems.

A new server can join the Grid simply by configuring itself to have an RLDS parent that is already within the Grid. There does not seem to be any way to prevent a machine's RLDS instance from declaring itself a child of another so in principle a Grid could grow without restriction. A consequence of that would be that the query time for service lookup could increase without limit. [The CROWN developers note that it would be possible to configure the RLDS service as a security RLDS service, and then use the access control policy to allow and deny registration to other machines.]

It is also possible that all nodes on a Grid would refer to a single authentication or authorisation service leading to performance bottlenecks. The solution to overload would be to duplicate such services on several nodes and partition the clients. Care should of course be taken to keep the authentication and authorisation services in synch, perhaps by having those nodes share a filesystem. Other aspects have not been investigated by this team.

## **4 User Experience**

### **4.1 Documentation for Users**

The CROWN architecture means that User Documentation is on a service by service basis and therefore provided by the service authors, not the CROWN developers. A client program's developers must provide the user documentation for that program. Likewise a service developer must provide consumer documentation.

A running CROWN Portal describes the services that it hosts. Again, the description must be provided by the service authors, not the CROWN developers and the common 'internal' services such as remote deployment or authentication are not described on the Portal but in the Users Guide

### **4.2 Joining the Grid**

The primary means of authentication to a CROWN Secure NodeServer is by GSI X.509 certificates. The Overview document refers briefly to a bridge between X.509 and Kerberos Ticket authentication domains but the evaluation team did not explore the Kerberos method.

A NodeServer client program can either assert identity by a pre-packaged and trusted Client Certificate, or it can programmatically obtain a User certificate from the User.

The Secure NodeServer can have its keystores modified by keytool to trust one or more Certificate Authority (CA's). A user that is calling services must obtain a certificate from a trusted CA.

Users of services hosted by a CROWN Portal must have a valid user id and password. We speculate that this registration process wraps the certificate creation process within the Portal.

### **4.3 Legacy Application Integration**

CROWN is not an arbitrary job execution engine. A legacy application must be wrapped as a Web Service using CROWN Designer and must include methods to transfer user's data to the machine that runs the program if necessary, or to specify existing files to the program. This Web Service can be packaged as a `.gar` file and remotely deployed to one or more of the Grid nodes provided that this is permitted by the Grid administrators. Otherwise a Grid node administrator must deploy the service.

More likely, the developer can start a new Node Server on the machine that supports the legacy application, join it to the Grid, and deploy the new service on the local machine. This new node's administrator may determine its own authentication and authorisation policy or alternatively the authentication and authorisation mechanisms can be delegated to a remote node that trusts the new node which requires administrator intervention at the other end.

(See in Appendix [8.5 CROWN Scheduler supports Legacy Applications](#), for updates in CROWN 2.0)

### **4.4 Migration between platforms**

A simple Web Service can be ported to CROWN without much difficulty apart from packaging it into a `gar` file with Designer. Likewise a service written for GT 3.9.3 and packaged as a `gar` file could be deployed to a NodeServer without much difficulty.

However a WSRF service written for GT 4.0 / GT 4.0.1 may not be able to run under CROWN 1.0 as the Globus API changed between the unstable GT 3.9.3 release and the production GT 4.0 release. CROWN services that rely on CROWN Security – message level encryption, authentication, , and authorisation – cannot run under Globus Toolkit 3.9.3 or 4.0.x. [The CROWN developers point out that any WSRF service written for GT4.0 or GT4.0.1 will run under the newer release CROWN 2.0]

A CROWN service could probably invoke vanilla Web Services hosted on a Tomcat/Axis or WebSphere machine and a CROWN service could invoke OMII hosted services. However if a vanilla or OMII hosted Web Service invoked a CROWN service on a Secure Node it would have to be authored to do so with access to CROWN client libraries.

## 4.5 Usability

We only used CROWN from an administrator or developer perspective. We cannot comment on any real users' views.

The *CROWN Portal User Center* displays general current statistics about the number of Submitted and Finished Jobs together with numbers of registered and online users. Registered users can log into the Portal and view the status of their own jobs.

## 4.6 Security from the User's Perspective

In general a portal user or client program must present a valid X509 certificate that is issued by a CA that is trusted by the controlling authentication service for the particular CROWN Grid (domain). To use a service on particular node, the authenticated user's credentials must be allowed by the policy rules for that server and service. The CROWN Portal conceals the X.509 details from the users and merely requires a user/password.

The Chinese user community generally submit jobs on known registered services via the CROWN Portal [1] which handles all the job control and notification for them. Portal users just need a registered user name and a browser.

The example client programs allow free substitution of identity by swapping .PEM certificate files without asking for pass phrases so these simple programs at least are vulnerable to certificate sharing or theft. A more sophisticated client program would need to include customised programmatic procedures before presenting a user certificate to a CROWN node as part of a service request and this is the responsibility of the program developers.

An experiment was carried out using the test installation. A certificate for each of the evaluation team sites was provided by the developers. The authorisation rules for the Counter Service (an example shipped with the distribution) were modified to deny authorisation for that service to Imperial College's certificate. A client program was executed that submitted a site certificate specified on the command line. The Southampton and Newcastle invocations were granted and the counter service gave the normal output. When the Imperial College invocation was executed, the counter service was denied by the NodeServer.

## 4.7 Verification

The end-user is reliant on the client program handling server errors correctly. The client program in turn is reliant on the Portal reporting status correctly.

We know that there is no mechanism (at least by default) for verifying the remote resource requirements because of an incident that occurred during the evaluation. The Southampton server underwent a re-installation of the CROWN Node Server but a step in the reconfiguration was omitted that left the server relying on an authorisation service that ran on one of the Beihang University machines. Every so often the Southampton server would be unable to execute any services and issued an apparently unrelated error message about missing encryption algorithms. With hindsight these periods coincided with the Beihang machine being taken down or its authorisation service being stopped. But the error message and stack trace gave us no way to identify that an off-site Web service was not available.

## **5 Developer Experience**

### **5.1 Documentation for Developers**

The CROWN team assume that all services will be developed in Java using the Eclipse IDE with the CROWN Designer plug-in. The developer documentation is the CROWN Designer section of the English User Guide PDF file and can be searched with Acrobat Reader's Search facility. The document accurately describes all the features of CROWN Designer and gives tutorials through worked examples of non-secure and secure service development. The non-secure service development example is a suitable quick start tutorial.

## 5.2 Languages and Tool Support

The CROWN Designer Eclipse plug-in is provided as part of the suite. It is possible to develop and package CROWN Java Web Services without Designer using command line compilation, ANT, or Sun JWSDP. However if the developer intends to write services that rely on CROWN's security features then the use of the security configuration wizards in Designer is highly desirable. The options available from the Designer plug-in can be seen in Figure 4.

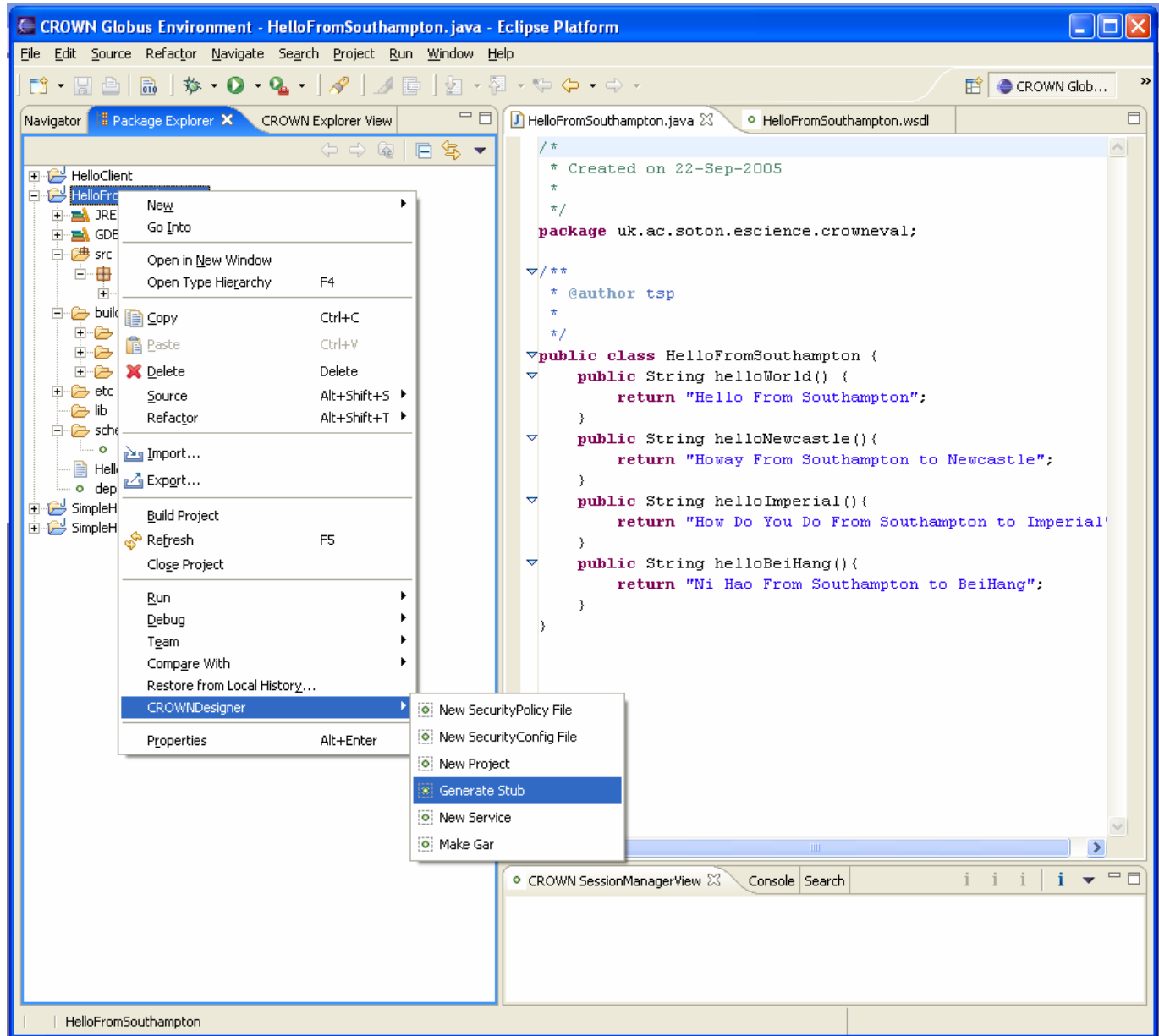


Figure 4

### 5.3 New Services

A Java Web Services developer that is familiar with Eclipse should find CROWN non-secure service development trivial with Designer. Designer wraps the Java2WSDL tool with an Ant script so it is easy to go from class to wrapped Web service. Additional tools allow the developer to package the new services in a .GAR file and to deploy it as shown in Figures 5 and 6.

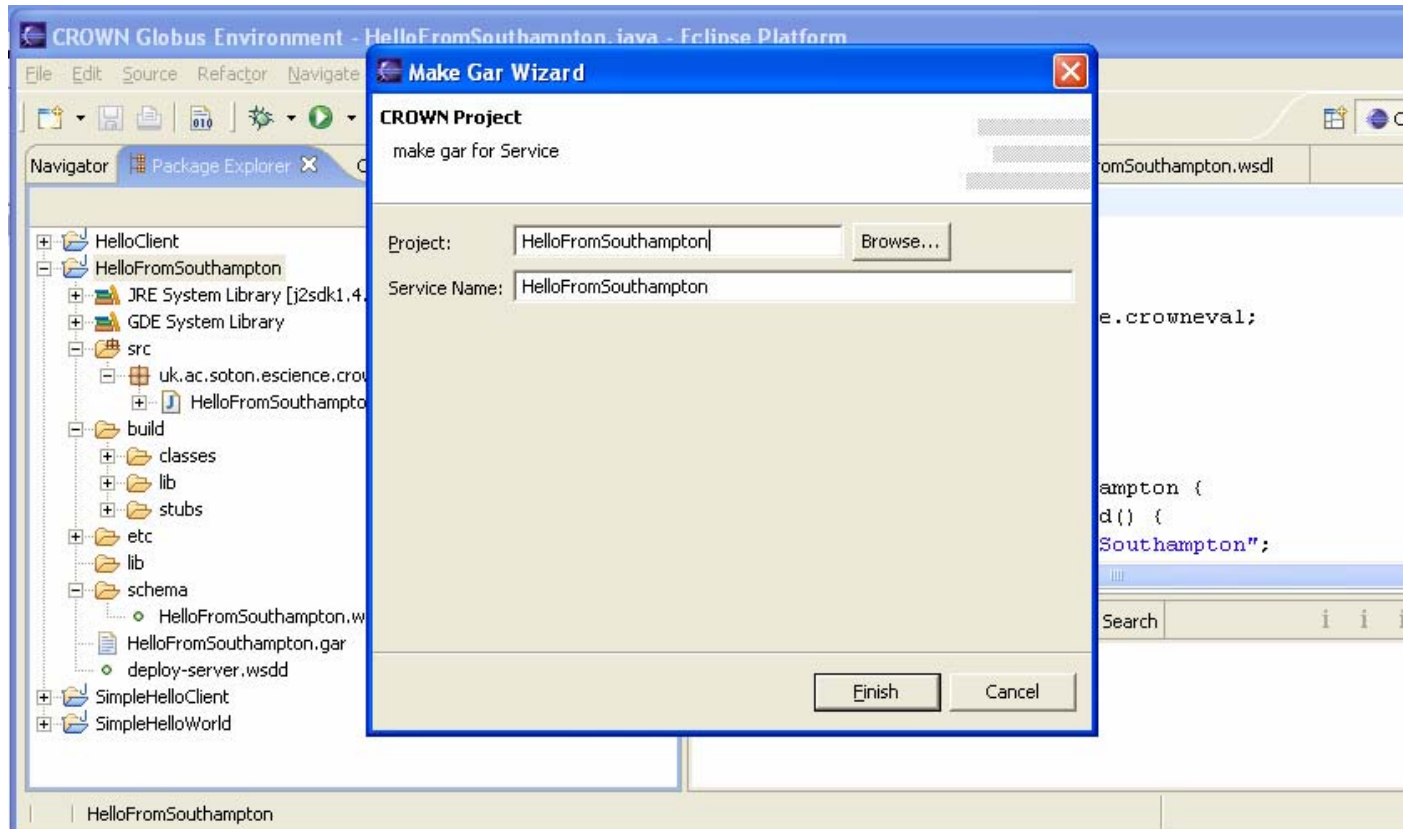


Figure 5

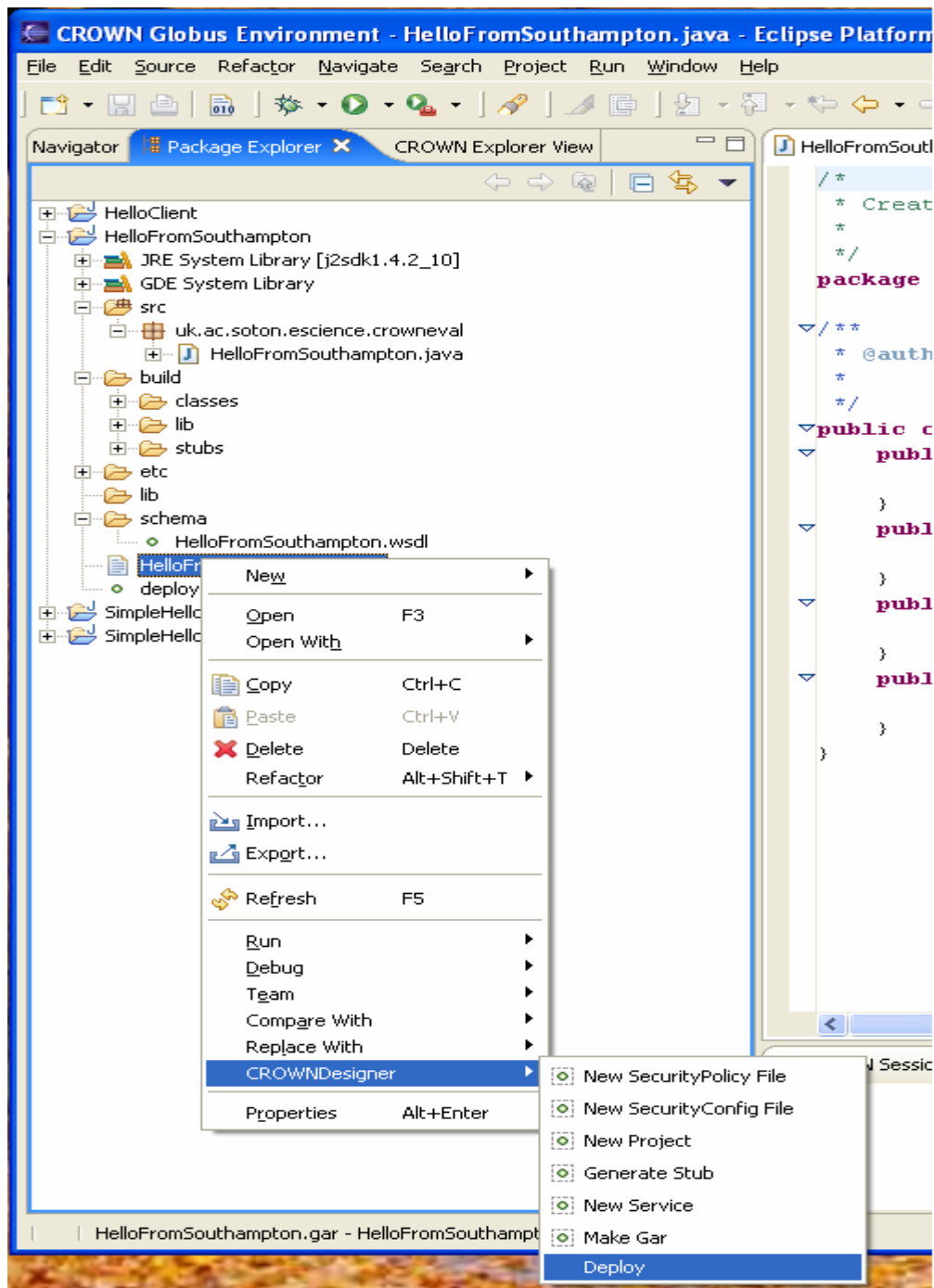


Figure 6

Once the wrapper has been generated it is hard to go back and change the service. If the developer wants either to expose extra methods or to remove / alter method signatures then the best course of action is to delete the wrapper and start again. (See in Appendix 8.6 [CROWN 2.0 Designer Improves Round-Trip Engineering](#) – this is no longer true in CROWN 2.0)

As mentioned earlier, the Overview document refers to a Client library distribution but we did not evaluate it.

Secure service development requires a much deeper understanding of the CROWN security handling mechanism in order to configure the policy files correctly. This is a lot harder to understand and would easily consume 80% of the developer's learning effort initially until the necessary expertise has been acquired. CROWN Designer provides some help with the

configuration tasks by offering an editing tool which avoids writing raw XML files, as shown in Figure 7.

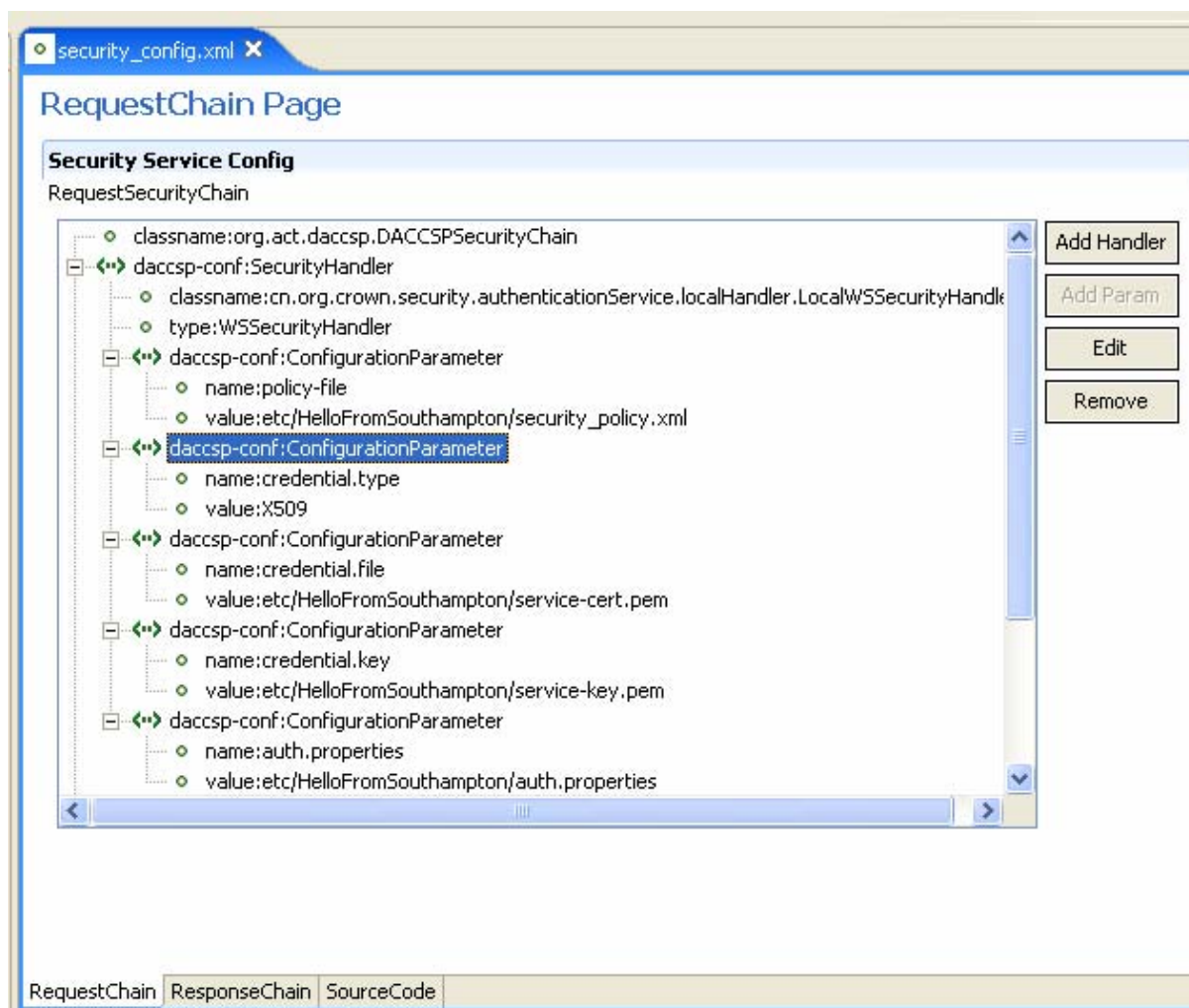


Figure 7

## 6 Technical

### 6.1 Architecture

#### 6.1.1 Derivation

CROWN 1.0 is built on and redistributes parts of the Globus Toolkit version 3.9.3. The 3.9.3 release was a development version which has since been superseded by the production releases GT 4.0 and 4.0.1.

Figure 8 shows the timeline of Globus and CROWN releases. The Globus boxes list features that were frozen in each particular release and were taken from an e-mail circulated to the ETF by Jennifer Schopf which summarise the Change Logs in the Release Notes in the Documentation Archive on [www.globus.org](http://www.globus.org). The CROWN boxes list the features that the CROWN developers consider improvements to GT 3.9.3 and are taken from a document circulated to the evaluation team by Tianyu Wo.

Other differences between GT 3.9.3 and GT 4.0 that we have been able to identify from the Globus change logs include:

- Improvements to the WS-Security, WS-Trust and WS-SecureConversation implementation so as to better comply with the relevant specifications.

- Alteration of the server's default behaviour to use transport-layer security, rather than GSI Secure Messaging, by default
- Implementation of later WSRF and WS-Notification draft specifications in the Java WS core.

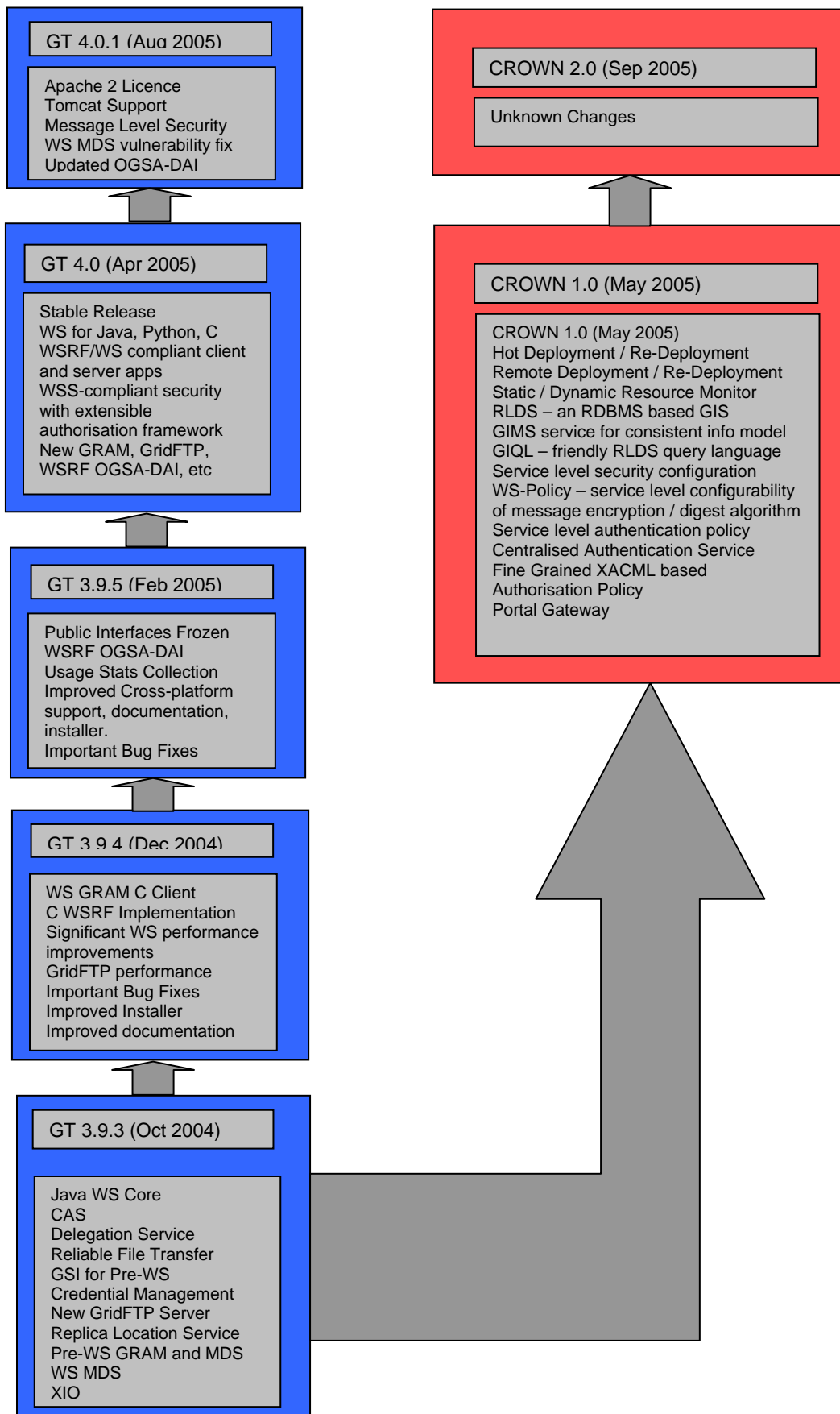


Figure 8

### **6.1.2 CROWN and SOA**

As CROWN is built on GT 3.9.3 and GT 3.9.3 is built on Web Services the system has all the SOA properties listed in the ETF criteria with the possible exception of Granularity which is a property of the hosted services rather than the hosting environment. CROWN is based on the Globus Toolkit Web Services alpha implementation and as such is clearly intended to be an SOA type environment. CROWN services are described by WSDL and are invoked and delivered across network connections using platform-neutral XML messages. CROWN is an environment that hosts services so Service Granularity is really up to the service designer.

## **6.2 Standards & Specifications**

CROWN uses the same standards as GT 3.9.3 (including SOAP, WSDL, and WSRF). In addition, it uses XACML for authorisation policy rules which the developers state to be an implementation of WS-Policy. The CROWN developers do not provide a list of standards that they aim to comply with.

A forest of CROWN NodeServers will by default interoperate with each other and not recognise any other Grid software such as Globus 2.4. We did not install the CROWN scheduler so we cannot tell whether this is a standalone job / service scheduler or whether it could be integrated with a cluster-management system, such as Maui.

CROWN uses a MySQL database to store historical data, one local MySQL instance per RLDS. We do not believe that it could be reconfigured to use the Oracle databases on the NGS machines.

A forest of CROWN node servers and their services could be configured to honour UK CA e-Science certificates that could be verified at invocation time by a single National authentication service node. Alternatively, an NGS CROWN Portal could be installed to grant registered users access to installed services by userid and password in the style of CNGRID.

## **6.3 Security**

CROWN's authentication technology is primarily X.509 certificate based. The Overview refers to Kerberos authentication domains but we did not investigate that feature. CROWN's authorisation service is configured using their implementation of the SAML based XACML language [2]. The documentation does not specify if they implement XACML 1.0 or 2.0.

CROWN implements message-level integrity through a configurable choice of digest algorithms. Message-level confidentiality is implemented through a configurable choice of encryption methods.

The authentication and authorisation frameworks can be made as complex as the Node Administrators choose to make it. The configuration of authorisation in particular is a specialised task that requires knowledge of XACML and this knowledge would tend to be restricted to one or two people. A large number of services on many nodes with different authorisation rule requirements may conceivably overwhelm the administrators' ability to implement them in a timely or accurate fashion.

CROWN does not seem to have a formal auditing framework. By default, the NodeServer logs all invocations includes security operations and these will include authorisation denials.

The logging can be made more comprehensive by changing property files but there is no auditing tool other than reading the server log file.

The default installation of the secure server is not completely secure. The most significant weakness lies with the default installation of the hot service-deployment capability. Whilst certainly functional and apparently robust, the CROWN software does not run services in robust compartments; any installed service will operate with the same credentials and capabilities as the CROWN server instance itself. As a result, this facility should not be exposed to untrusted end-users or service providers. To lock it down requires several keystores to be edited.

We do not know of any security audits or any publicised security breaches.

#### **6.4 Industrial Support**

The CROWN suite can only leverage industrial software development practices and technologies to a limited extent. This is due to the nature of the CROWN Designer tool as an Eclipse plug-in. A commercial Web Services development environment such as IBM's *Rational Application Developer 6.0*, which is based on Eclipse, could accommodate the CROWN Designer plug-in. However other IDEs such as Oracle Jdeveloper or BEA WebLogic Workshop that are not based on Eclipse would not benefit from Designer.

CROWN will run on an industrial strength host machine. We have already run CROWN on a 64-bit multi-processor Red Hat Enterprise Linux system without problems. In fact, the production installation at Beihang used by CNGRID is a large 64-bit multi-processor RHEL system.

Even if CROWN can only store its system information in a MySQL database rather than Oracle this is good enough. MySQL is used in many production environments and we consider it to be robust.

## 7 Conclusions

CROWN is an extension of GT 3.9.3 that hosts Java Web Services and adds remote hot-deployment, an RDBMS GIS (Grid information system), service level security processing, service level configurable encryption and signature algorithms, service level authentication policy, centralised authentication service, and fine-grained, XACML/SAML described, authorisation policy.

CROWN's strengths include:

- portability (mainly pure Java)
- Hot service deployment capability (local and remote)
- Ability to configure service level security
- Ease of creating a forest of related Node Servers
- Ability to centralise authentication
- XACML based authorisation policy
- A bundled service development tool
- A track record of successful deployment in the CNGRID project.

CROWN's weaknesses include:

- an apparent sensitivity to different versions of the server software
- a complex file hierarchy containing at least one keystore per service
- an access policy deep down this hierarchy that requires knowledge of a rich XML format (XACML) which is sensitive to loss through redeployment
- the development tools force service authors to use Eclipse
- Legacy applications must be wrapped as services by a developer that has Java Web Services skills and who understands the CROWN architecture.
- CROWN is based on an unstable version of Globus Toolkit (although later releases may be based on stable GT 4.x )

There is an uninvestigated component – the Portal Scheduler – that may or may not interoperate with PBSPro.

The Secure Node Server could clearly be deployed on NGS Red Hat IA32 or IA64 machines. Given that JDK's exist for Solaris, HP-UX, AIX and Irix it should also be deployable on those architectures.

The hard part would be to develop Web Service wrappers for any existing NGS services and to develop the topology and authorisation rules. With the installation of a CROWN Portal then the NGS could advertise services in the same way as CNGRID through a single gateway.

We conclude that secure CROWN NodeServers would be suitable for deployment on the NGS provided that the administrators obtain sufficient training to implement and maintain the Node level access control. User experience could probably be made simpler by implementing the Portal component as a single gateway to services.

## References

- [1] The CROWN Portal, <http://www.crown.org.cn/en> .
- [2] XACML, [www.oasis-open.org/specs/index.php#xacmlv2.0](http://www.oasis-open.org/specs/index.php#xacmlv2.0)
- [3] COLAB: Leeds / Beihang Universities CROWN Grid research collaboration, <http://www.wrgrid.org.uk/leaflets/colab.pdf>

## **8 Appendix**

Since the Evaluation took place, there have been subsequent releases of CROWN The CROWN team have given us the following information:

### **8.1 Base Level of Globus Toolkit**

CROWN 2.0 is based on the Production version of the Globus Toolkit version 4.0.1 Core

### **8.2 Eclipse Plug-in Support**

As of CROWN 2.0, CROWN Designer runs under Eclipse 3.0 and later versions.

### **8.3 Auto-start and auto-stop**

As of CROWN 2.0, the integration of auto-start and auto-stop into the boot / start-up sequence has been introduced.

### **8.4 CROWN Monitor**

In CROWN 2.0, the Monitor can collect the service invocation information and present it to a GUI client.

### **8.5 CROWN Scheduler supports Legacy Applications**

In CROWN 2.0, the Scheduler supports POSIX application type for legacy programs. This is similar to JSDL / GridSAM.

### **8.6 CROWN 2.0 Designer Improves Round-Trip Engineering**

As of CROWN 2.0 there is a synchronisation function in Designer. When service implementation classes are updated or otherwise modified, the wrappers do not have to be re-created.