NORDUGRID

THE LOGGER SERVICE

Functionality Description and Installation Manual

A.Konstantinov[*]

A.Nazarov[†]

[*]aleks@fys.uio.no
[†]moby@ut.ee

Contents

1   Introduction

This document describes NorduGrid Logger service and NorduGrid Usage Record. Goals of the service are storing information about submitted jobs and providing that information and statistics. The Usage Record represents single grid job processed by the Logger Service.

2   NorduGrid Usage Record

2.1   General

The Usage Record can be considered as composition of properties or parameters that describes a single submitted grid job. These parameters provide job identification information, information about job status, different timing characteristics etc.

2.2   Usage Record properties

Full list of the Usage Record properties and short descriptions present below. "Support Status" column contains information about implementation status of the corresponding property.

Property	Description	Support Status
charge	total charge of the job	not clear how to collect
cluster	subject of cluster certificate	yes
downloadtime	time it took to gather all input files requested for job (stage in)	not yet collected
endtime	timestamp when computing resource finished processing job	yes
exitcode	numerical code returned by job's main executable	yes
failurestring	textual description of failure which happened during processing of the job	yes
globaljobid	global identifier of an ARC Grid job	yes
globaluserid	subject of user's certificate	yes
jobdescription	description of a job as passed to computing resource	yes
jobname	name of a job specified by user	yes
localjobid	id of a job in LRMS	yes
localuserid	name/id of a user used to execute job on computing resource	yes
lrms	type of batch system used at computing resource (aka LRMS)	yes
lrmsendtime	time LRMS finished processing job	not yet collected
lrmssubmissiontime	time job was passed to LRMS	not yet collected
network	network (volume) used by the job during execution	not clear how to collect
nodecount	number of computing nodes allocated for the job	yes
nodename *	host names of computers which were executing job	yes
processid *	process ID of the job	not clear how to collect
processors	number of processors allocated for the job	not yet collected

projectname	name of the project	not yet collected
queue	name of a queue on computing resource where aplicable	yes
requestedcputime	amount of CPU time requested in job's description (minutes)	yes
requesteddisk	amount of disk space requested in job's description	yes
requestedmemory	amount of memory requested in job's description	yes
requestedwalltime	amount of CPU wall clock time requested in job's description	yes
runtimeenvironment*	description of runtime environment used by the job	yes
servicelevel	quality of service provided while executing the job	not clear how to collect
stageindata	amount of staged in data	not yet collected
stageoutdata	amount of staged out data	not yet collected
status	completions status of the job (finished/failed/killed/not completed)	yes
submissiontime	time when job reached computing resource	yes
submithost	hostname or/and certificate from which user submited a job	yes
uploadtime	time it took to distribute all output files produced by job (stage out)	not yet collected
usedcputime	CPU time consumed by job during executing	yes
useddisk	amount of disk space taken by job while executing	not clear how to collect
usedmemory	maximal amount of memory used by job while executing	not clean how to collect[‡]
usedswap	amount of the swap memory space used by the job	not clear how to collect
usedwalltime	time it took for job to finish executing	yes

Multivalued properties indicated with *. Nodename is colon-separated, runtimeenvironment is space-separated.

3   The Logger Service

3.1   General

The Logger service is one of the services implemented in HTTP(S,G) and SOAP Framework of ARC[1] and run in *httpsd* server. The Logger service is simple frontend to underlying MySQL database to store and retrieve an information about grid jobs. It provides two SOAP methods for adding and querying Usage Records stored in the database. Corresponding WSDL is avaialble in Appendix B4. It also defines the Usage Record format.

[‡]At the moment we have problems collecting this value for most of LRMS systems.
But some LRMS systems allow to collect it (for example, Condor).
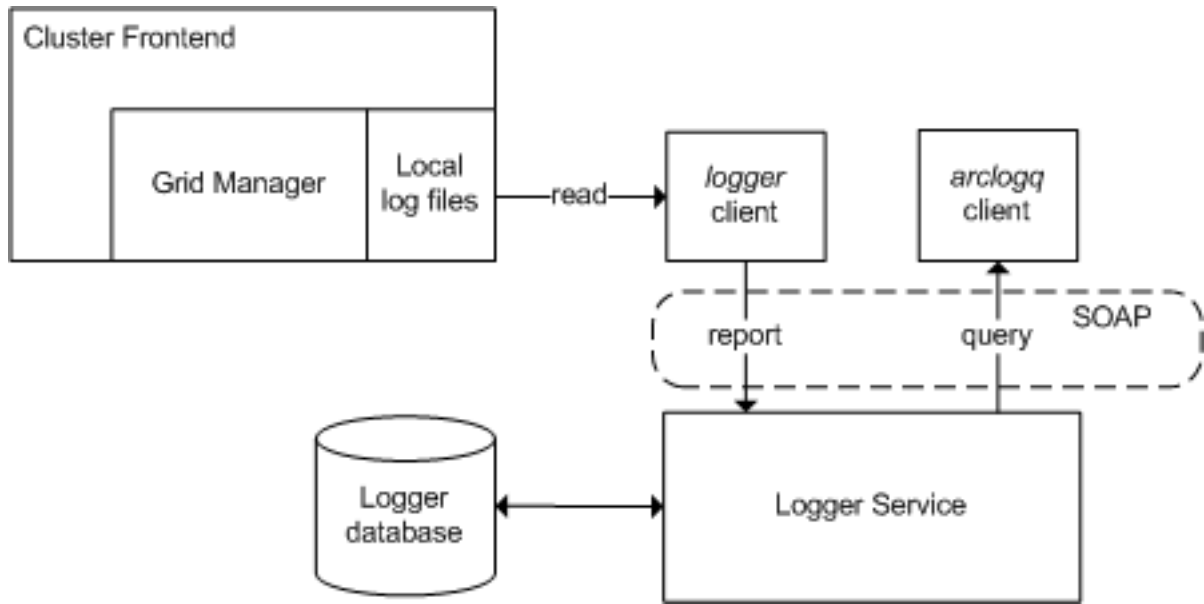
Workflow schema.

The Logger service and the clients is part of the ARC middleware [2]. *logger* client is usually deployed together with the Grid Manager on the computing resources. The Usage Records are sent to the Logger service by a *logger* client using SOAP. The service stores information present by Usage Records in a database. Jobs information stored in database can also be queried by the *arclogq* client.

3.2 Installation

3.2.1 Setting up database

Logger service uses MySQL database version 5.0 or greater. So it is essential to configure database properly. Currently You will either need file grid-manager/https/logger/create.sql or use content of Appendix A. It is important that You look into that file and modify first 3 lines (especially password). You can keep them as is, but at least make sure they fit Your intentions.

You have to feed that file to MySQL server. Preferably using *myslq* utility:

mysql -p -u root < create.sql

Please, check manual of *mysql* and configuration of the server before You use that.

If command execution is succeeded server should have new database called *nglogger* and new user also called *nglogger* allowed to do everything inside that database.

If You modify *create.sql* consider that view *v_jobs* and stored procedure *add_job()* must have same signatures.

That's it. You have configured database.

3.2.2 Database schema description

The Logger service database has following schema:
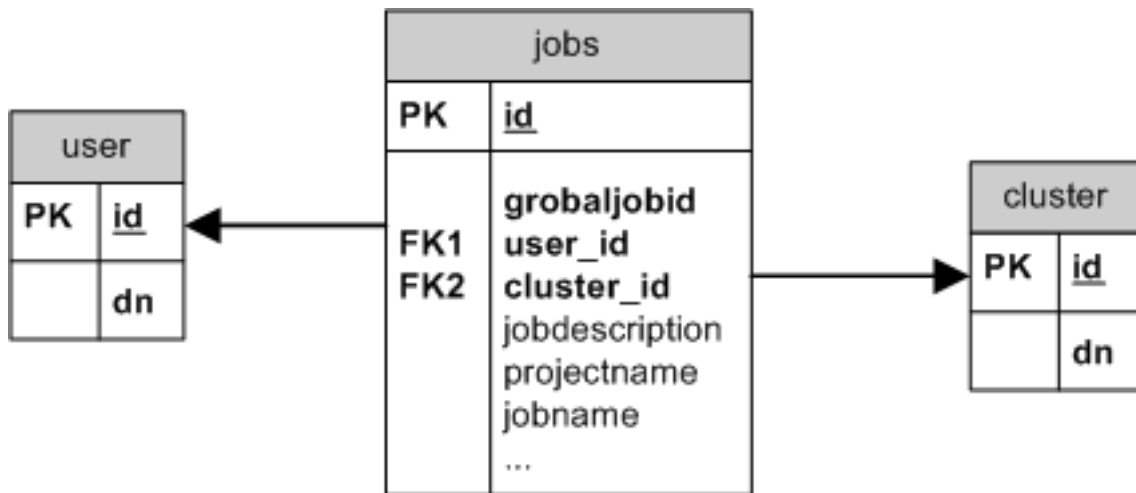
Logger database schema.

For more details about database schema see grid-manager/https/logger/create.sql script or Appendix A.

3.3 Configuration

The Logger service is configured through the [httpsd/logger] block of the central *arc.conf* file[3]. It supports following service-specific commands.

- *acl_read [group [group [...]]]*

- *acl_write [group [group [...]]]*

- *acl_query [group [group [...]]]*

If client belongs to one of groups listed in *acl_read* command it is allowed to retrieve information about job records. Those clients are still restricted to an information about jobs which they are owners of. And those listed in *acl_write* can add new records. Usually those groups match against host credentials of a clusters' frontends. Groups in *acl_query* are allowed to retrieve information about *all* jobs.

Other supported commands are:

- *sqlcontact [user [password]]*

- *sqlcontactsource path*

They provide username and password used to access MySQL database. *sqlcontactsource* points to file which contains username and password in first line. It is better to use *sqlcontactsource* and make it readable only for user who starts the server.

3.4 Startup and shutdown

As long as Logger service is part of HTTP(S,G) follow instruction in [1] to start and stop server. Look into log file for line which looks like "Added service logger at /some/path". If You do not see it try to look through file for anything suspicious.

3.5 Usage of the Logger Client

Logger service can be accesed with the *logger* utility to send data and with the *arclogq* utility to query data. These utilities are installed into ${NORDUGRID_LOCATION}/libexec by default. *logger* client can be used by a grid-manager to automatically send Usage Records to selected service(s). The target service can be specified in the *arc.conf* (*jobreport* configuration parameter) or can be given in the grid job description file(*jobreport* xrsl attribute).

Use following command to send jobs data to the Logger service:

```
logger [-h] [-d level] [-u url] [-E expiration_period_days] control_dir ...
```

Use following command to query jobs data from the Logger service:

```
arclogq [-h] [-d level] [-u url] [-x] [-o offset] [-s size] [-e element, element, ...] [query]
```

Here possible options are

-h   show reminder;

-d   set debug output to *level*;

-u   URL of central logging server;

-E   expiration period of gathered jobs information

-x   print obtained information in XML,

-o   from which record to start,

-s   how many records to show,

-e   specify which elements are desired in output.
     Posible values are: *start,end,cluster,user,id,name,failure,lrms,queue,rsl,ui,usedcpu,usedmem*.

control_dir is location directory of grid-manager log files with jobs information

query is SQL conditional expression used to select jobs' records.

Query examples:

"submissiontime > '10.12.2005'"

"submissiontime >= '10.12.2005' AND endtime <= '17.12.2005'"

"user LIKE '%John Doe%'"

4 Privacy issues[§]

The Logger service collects some information that can broke user's privacy. You should take to account that UR atributes present in the list below can be accessed by other users:

- failurestring

- globaluserid

[§]based on comments from Arto Teräs, CSC

- jobdescription

- jobname

- localuserid

Appendix A

DDL script to create Logger service database:

```
CREATE DATABASE IF NOT EXISTS nglogger;
USE nglogger;
CREATE TABLE IF NOT EXISTS jobs (
        id INT AUTO_INCREMENT PRIMARY KEY,
        globaljobid VARCHAR(255) NOT NULL,
        user_id INT UNSIGNED NOT NULL,
        cluster_id INT UNSIGNED NOT NULL,
        jobdescription BLOB NULL DEFAULT NULL,
        projectname VARCHAR(255) NULL DEFAULT NULL,
        jobname VARCHAR(255) NULL DEFAULT NULL,
        submithost VARCHAR(255) NULL DEFAULT NULL,
        requestedcputime INT NULL DEFAULT NULL,
        requestedwalltime INT NULL DEFAULT NULL,
        requestedmemory INT NULL DEFAULT NULL,
        requesteddisk INT NULL DEFAULT NULL,
        submissiontime DATETIME NULL DEFAULT NULL,
        localuserid VARCHAR(255) NULL DEFAULT NULL,
        queue VARCHAR(32) NULL DEFAULT NULL,
        lrms VARCHAR(32) NULL DEFAULT NULL,
        localjobid VARCHAR(255) NULL DEFAULT NULL,
        lrmssubmissiontime DATETIME NULL DEFAULT NULL,
        lrmsendtime DATETIME NULL DEFAULT NULL,
        nodename BLOB NULL DEFAULT NULL,
        nodecount INT DEFAULT 1,
        processors INT DEFAULT 1,
        exitcode INT NULL DEFAULT NULL,
        failurestring BLOB NULL DEFAULT NULL,
        usedcputime INT NULL DEFAULT NULL,
        usedwalltime INT NULL DEFAULT NULL,
        usedmemory INT NULL DEFAULT NULL,
        useddisk INT NULL DEFAULT NULL,
        status VARCHAR(255) NULL DEFAULT NULL,
        endtime DATETIME NULL DEFAULT NULL,
        downloadtime INT NULL DEFAULT NULL,
        uploadtime INT NULL DEFAULT NULL,
        processid INT NULL DEFAULT NULL,
        charge INT NULL DEFAULT NULL,
        network VARCHAR(255) NULL DEFAULT NULL,
        stageindata INT NULL DEFAULT NULL,
        stageoutdata INT NULL DEFAULT NULL,
        usedswap INT NULL DEFAULT NULL,
        servicelevel VARCHAR(255) NULL DEFAULT NULL,
        runtimeenvironment BLOB NULL DEFAULT NULL,
        INDEX start_failure(globaljobid,submissiontime,endtime,status),
        INDEX cluster(cluster_id),
        INDEX user(user_id)
) ENGINE = MyISAM;
CREATE TABLE IF NOT EXISTS user (
        id INT UNSIGNED AUTO_INCREMENT,
        dn VARCHAR(255) CHARACTER SET utf8 NOT NULL,
        KEY id(id)
) ENGINE = InnoDB;
CREATE TABLE IF NOT EXISTS cluster (
        id INT UNSIGNED AUTO_INCREMENT,
        dn VARCHAR(255) CHARACTER SET utf8 NOT NULL,
        KEY id(id)
```

```
) ENGINE = InnoDB;
CREATE VIEW v_jobs AS
SELECT
        jobs.id,
        jobs.globaljobid,
        user.dn as usersn,
        cluster.dn as cluster,
        jobs.jobdescription,
        jobs.projectname,
        jobs.jobname,
        jobs.submithost,
        jobs.requestedcputime,
        jobs.requestedwalltime,
        jobs.requestedmemory,
        jobs.requesteddisk,
        jobs.submissiontime,
        jobs.localuserid,
        jobs.queue,
        jobs.lrms,
        jobs.localjobid,
        jobs.lrmssubmissiontime,
        jobs.lrmsendtime,
        jobs.nodename,
        jobs.nodecount,
        jobs.processors,
        jobs.exitcode,
        jobs.failurestring,
        jobs.usedcputime,
        jobs.usedwalltime,
        jobs.usedmemory,
        jobs.useddisk,
        jobs.status,
        jobs.endtime,
        jobs.downloadtime,
        jobs.uploadtime,
        jobs.processid,
        jobs.charge,
        jobs.network,
        jobs.stageindata,
        jobs.stageoutdata,
        jobs.usedswap,
        jobs.servicelevel,
        jobs.runtimeenvironment
FROM jobs, user, cluster
WHERE jobs.user_id = user.id AND jobs.cluster_id = cluster.id;
delimiter //
CREATE PROCEDURE add_job(
        p_globaljobid VARCHAR(255),
        p_globaluserid VARCHAR(255) CHARACTER SET utf8,
        p_cluster VARCHAR(255) CHARACTER SET utf8,
        p_jobdescription BLOB,
        p_projectname VARCHAR(255),
        p_jobname VARCHAR(255),
        p_submithost VARCHAR(255),
        p_requestedcputime INT,
        p_requestedwalltime INT,
        p_requestedmemory INT,
        p_requesteddisk INT,
        p_submissiontime DATETIME,
        p_localuserid VARCHAR(255),
        p_queue VARCHAR(32),
        p_lrms VARCHAR(32),
        p_localjobid VARCHAR(255),
        p_lrmssubmissiontime DATETIME,
        p_lrmsendtime DATETIME,
        p_nodename BLOB,
        p_nodecount INT,
        p_processors INT,
        p_exitcode INT,
        p_failurestring BLOB,
```

```
            p_usedcputime INT,
            p_usedwalltime INT,
            p_usedmemory INT,
            p_useddisk INT,
            p_status VARCHAR(255),
            p_endtime DATETIME,
            p_downloadtime INT,
            p_uploadtime INT,
            p_processid INT,
            p_charge INT,
            p_network VARCHAR(255),
            p_stageindata INT,
            p_stageoutdata INT,
            p_usedswap INT,
            p_servicelevel VARCHAR(255),
            p_runtimeenvironment BLOB
            )
    BEGIN
            DECLARE v_cluster_id INT DEFAULT NULL;
            DECLARE v_user_id INT DEFAULT NULL;
            DECLARE v_entry_count INT DEFAULT 0;
            DECLARE s VARCHAR(200) DEFAULT ";
            DECLARE s_length INT;
            -- 1. try to get cluster
            SELECT id INTO v_cluster_id FROM cluster WHERE dn = p_cluster;
            IF( v_cluster_id IS NULL) THEN
                    INSERT INTO cluster(dn) values (p_cluster);
                    SELECT LAST_INSERT_ID() INTO v_cluster_id;
            END IF;

            -- 2. try to get user
            SELECT id INTO v_user_id FROM user WHERE dn = p_globaluserid;
            IF( v_user_id IS NULL) THEN
                    INSERT INTO user(dn) values (p_globaluserid);
                    SELECT LAST_INSERT_ID() INTO v_user_id;
            END IF;
            -- 3. create set statement
            IF(p_jobdescription IS NOT NULL) THEN
                    SET s = CONCAT(s, ' jobdescription = \", p_jobdescription, '\',');
            END IF;
            IF(p_projectname IS NOT NULL) THEN
                    SET s = CONCAT(s, ' projectname = \", p_projectname, '\',');
            END IF;
            IF(p_jobname IS NOT NULL) THEN
                    SET s = CONCAT(s, ' jobname = \", p_jobname, '\',');
            END IF;
            IF(p_submithost IS NOT NULL) THEN
                    SET s = CONCAT(s, ' submithost = \", p_submithost, '\',');
            END IF;
            IF(p_requestedcputime IS NOT NULL) THEN
                    SET s = CONCAT(s, ' requestedcputime = ', p_requestedcputime, ',');
            END IF;
            IF(p_requestedwalltime IS NOT NULL) THEN
                    SET s = CONCAT(s, ' requestedwalltime = ', p_requestedwalltime, ',');
            END IF;
            IF(p_requestedmemory IS NOT NULL) THEN
                    SET s = CONCAT(s, ' requestedmemory = ', p_requestedmemory, ',');
            END IF;
            IF(p_requesteddisk IS NOT NULL) THEN
                    SET s = CONCAT(s, ' requesteddisk = ', p_requesteddisk, ',');
            END IF;
            IF(p_submissiontime IS NOT NULL) THEN
                    SET s = CONCAT(s, ' submissiontime = \", p_submissiontime, '\',');
            END IF;
            IF(p_localuserid IS NOT NULL) THEN
                    SET s = CONCAT(s, ' localuserid = \", p_localuserid, '\',');
            END IF;
            IF(p_queue IS NOT NULL) THEN
                    SET s = CONCAT(s, ' queue = \", p_queue, '\',');
            END IF;
```

10

```
IF(p_lrms IS NOT NULL) THEN
        SET s = CONCAT(s, ' lrms = \", p_lrms, '\',');
END IF;
IF(p_localjobid IS NOT NULL) THEN
        SET s = CONCAT(s, ' localjobid = \", p_localjobid, '\',');
END IF;
IF(p_lrmssubmissiontime IS NOT NULL) THEN
        SET s = CONCAT(s, ' lrmssubmissiontime = \", p_lrmssubmissiontime, '\',');
END IF;
IF(p_lrmsendtime IS NOT NULL) THEN
        SET s = CONCAT(s, ' lrmsendtime = \", p_lrmsendtime, '\',');
END IF;
IF(p_nodename IS NOT NULL) THEN
        SET s = CONCAT(s, ' nodename = \", p_nodename, '\',');
END IF;
IF(p_nodecount IS NOT NULL) THEN
        SET s = CONCAT(s, ' nodecount = ', p_nodecount, ',');
END IF;
IF(p_processors IS NOT NULL) THEN
        SET s = CONCAT(s, ' processors = ', p_processors, ',');
END IF;
IF(p_exitcode IS NOT NULL) THEN
        SET s = CONCAT(s, ' exitcode = ', p_exitcode, ',');
END IF;
IF(p_failurestring IS NOT NULL) THEN
        SET s = CONCAT(s, ' failurestring = \", p_failurestring, '\',');
END IF;
IF(p_usedcputime IS NOT NULL) THEN
        SET s = CONCAT(s, ' usedcputime = ', p_usedcputime, ',');
END IF;
IF(p_usedwalltime IS NOT NULL) THEN
        SET s = CONCAT(s, ' usedwalltime = ', p_usedwalltime, ',');
END IF;
IF(p_usedmemory IS NOT NULL) THEN
        SET s = CONCAT(s, ' usedmemory = ', p_usedmemory, ',');
END IF;
IF(p_useddisk IS NOT NULL) THEN
        SET s = CONCAT(s, ' useddisk = ', p_useddisk, ',');
END IF;
IF(p_status IS NOT NULL) THEN
        SET s = CONCAT(s, ' status = \", p_status, '\',');
END IF;
IF(p_endtime IS NOT NULL) THEN
        SET s = CONCAT(s, ' endtime = \", p_endtime, '\',');
END IF;
IF(p_downloadtime IS NOT NULL) THEN
        SET s = CONCAT(s, ' downloadtime = \", p_downloadtime, '\',');
END IF;
IF(p_uploadtime IS NOT NULL) THEN
        SET s = CONCAT(s, ' endtime = \", p_uploadtime, '\',');
END IF;
IF(p_processid IS NOT NULL) THEN
        SET s = CONCAT(s, ' processid = \", p_processid, '\',');
END IF;
IF(p_charge IS NOT NULL) THEN
        SET s = CONCAT(s, ' charge = \", p_charge, '\',');
END IF;
IF(p_network IS NOT NULL) THEN
        SET s = CONCAT(s, ' network = \", p_network, '\',');
END IF;
IF(p_stageindata IS NOT NULL) THEN
        SET s = CONCAT(s, ' stageindata = \", p_stageindata, '\',');
END IF;
IF(p_stageoutdata IS NOT NULL) THEN
        SET s = CONCAT(s, ' stageoutdata = \", p_stageoutdata, '\',');
END IF;
IF(p_usedswap IS NOT NULL) THEN
        SET s = CONCAT(s, ' usedswap = \", p_usedswap, '\',');
END IF;
IF(p_servicelevel IS NOT NULL) THEN
```

```
                    SET s = CONCAT(s, ' servicelevel = \", p_servicelevel, '\',');
            END IF;
            IF(p_runtimeenvironment IS NOT NULL) THEN
                    SET s = CONCAT(s, ' runtimeenvironment = \", p_runtimeenvironment, '\',');
            END IF;

            SET s_length = CHAR_LENGTH(s);

            -- 4. check if exist jobs with the same attributes
            SELECT count(*) INTO v_entry_count FROM jobs WHERE cluster_id = v_cluster_id AND user_id = v_user_id AND globaljobid = p_globaljobid;
            -- 5. update/insert data
            IF(v_entry_count > 0 AND s_length > 0) THEN
                    SET @q = CONCAT('UPDATE jobs SET', SUBSTRING(s, 1, s_length-1), ' WHERE cluster_id = ', v_cluster_id, ' AND user_id = ', v_user_
                    PREPARE stmt FROM @q;
                    EXECUTE stmt;
                    DEALLOCATE PREPARE stmt;
            ELSEIF(v_entry_count = 0) THEN
                    SET s = CONCAT(s, ' cluster_id = ', v_cluster_id, ',');
                    SET s = CONCAT(s, ' user_id = ', v_user_id, ',');
                    SET s = CONCAT(s, ' globaljobid = \", p_globaljobid, '\");
                    SET @q = CONCAT('INSERT INTO jobs SET', s);
                    PREPARE stmt FROM @q;
                    EXECUTE stmt;
                    DEALLOCATE PREPARE stmt;
            END IF;
    END
    //
    delimiter ;
    GRANT ALL ON nglogger.* TO 'nglogger'@'localhost' IDENTIFIED BY 'pass';
```

Appendix B

WSDL document describing Logger Web Service interface:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="http://www.nordugrid.org/ws/schemas/ARCLoggerV2"
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:nl2="http://www.nordugrid.org/ws/schemas/ARCLoggerV2"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
 <schema targetNamespace="http://www.nordugrid.org/ws/schemas/ARCLoggerV2"
  xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
  <simpleType name="ResultCode">
   <restriction base="xsd:string">
    <enumeration value="NoError"/>
    <enumeration value="UndefinedError"/>
   </restriction>
  </simpleType>
  <complexType name="Result">
   <sequence>
     <element name="Code" type="nl2:ResultCode"/>
     <element name="Description" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
    </sequence>
  </complexType>
  <complexType name="UsageRecord">
   <sequence>
     <element name="globaljobid" type="xsd:string"/>
     <element name="globaluserid" type="xsd:string"/>
     <element name="cluster" type="xsd:string"/>
     <element name="jobdescription" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
```

12

```xml
        <element name="projectname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="jobname" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="submithost" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="requestedcputime" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="requestedwalltime" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="requestedmemory" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="requesteddisk" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="submissiontime" type="xsd:dateTime" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="localuserid" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="queue" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="lrms" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="localjobid" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="lrmssubmissiontime" type="xsd:dateTime" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="lrmsendtime" type="xsd:dateTime" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="nodename" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="nodecount" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="processors" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="exitcode" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="failurestring" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="usedcputime" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="usedwalltime" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="usedmemory" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="useddisk" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="status" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="endtime" type="xsd:dateTime" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="downloadtime" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
        <element name="uploadtime" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="processid" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="charge" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="network" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="stageindata" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="stageoutdata" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="usedswap" type="xsd:int" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="servicelevel" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
            <element name="runtimeenvironment" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
        <any namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
   </complexType>

   <complexType name="addRequest">
    <sequence>
     <element name="job" type="nl2:UsageRecord" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
   </complexType>
   <complexType name="addResponse">
    <sequence>
     <element name="result" type="nl2:Result"/>
    </sequence>
   </complexType>
   <complexType name="getRequest">
    <sequence>
     <element name="query" type="xsd:string" minOccurs="0" maxOccurs="1" nillable="true"/>
     <element name="offset" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
     <element name="size" type="xsd:unsignedInt" minOccurs="1" maxOccurs="1"/>
    </sequence>
   </complexType>
   <complexType name="getResponse">
    <sequence>
     <element name="result" type="nl2:Result"/>
     <element name="job" type="nl2:UsageRecord" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
   </complexType>

  <element name="add" type="nl2:addRequest"/>
  <element name="addResponse" type="nl2:addResponse"/>
  <element name="get" type="nl2:getRequest"/>
  <element name="getResponse" type="nl2:getResponse"/>
 </schema>
</wsdl:types>
<wsdl:message name="addRequest">
```

```
  <wsdl:part name="addRequest" element="nl2:add"/>
</wsdl:message>
<wsdl:message name="addResponse">
  <wsdl:part name="addResponse" element="nl2:addResponse"/>
</wsdl:message>
<wsdl:message name="getRequest">
  <wsdl:part name="getRequest" element="nl2:get"/>
</wsdl:message>
<wsdl:message name="getResponse">
  <wsdl:part name="getResponse" element="nl2:getResponse"/>
</wsdl:message>
<wsdl:portType name="ARCLoggerV2PortType">
  <wsdl:operation name="add">
    <wsdl:documentation>Add new information into database</wsdl:documentation>
    <wsdl:input name="addRequest" message="nl2:addRequest"/>
    <wsdl:output name="addResponse" message="nl2:addResponse"/>
  </wsdl:operation>
  <wsdl:operation name="get">
    <wsdl:documentation>Query information in database</wsdl:documentation>
    <wsdl:input name="getRequest" message="nl2:getRequest"/>
    <wsdl:output name="getResponse" message="nl2:getResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ARCLoggerV2" type="nl2:ARCLoggerV2PortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="add">
    <soap:operation soapAction=""/>
    <wsdl:input name="addRequest">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="addResponse">
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="get">
    <soap:operation soapAction=""/>
    <wsdl:input name="getRequest">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getResponse">
      <soap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="ARCLoggerV2">
  <wsdl:documentation>Database to keep and search information about jobs executed at NG-enabled GRID sites</wsdl:documentation>
  <wsdl:port name="ARCLoggerV2" binding="nl2:ARCLoggerV2">
    <soap:address location="http://localhost:80"/>
  </wsdl:port>
</wsdl:service>
</definitions>
```

Appendix C

Mappings between properties in local log files and UR properties.

```
Local log files      Usage Record
charge               -
cluster              cluster
downloadtime         -
endtime              endtime
exitcode             exitcode
failurestring        failurestring
globaljobid          ngjobid
globaluserid         usersn
jobdescription       description
```

```
jobname              jobname
localjobid           localjobid
localuserid          localuser
lrms                 lrms
lrmsendtime          -
lrmssubmissiontime   -
network              -
nodecount            nodecount
nodename             nodename
processid            -
processors           -
projectname          -
queue                queue
requestedcputime     requestedcputime
requesteddisk        requesteddisk
requestedmemory      requestedmemory
requestedwalltime    requestedwalltime
runtimeenvironment   runtimeenvironment
servicelevel         -
stageindata          -
stageoutdata         -
status               status
submissiontime       submissiontime
submithost           clienthost
uploadtime           -
usedcputime          usedcputime
useddisk             -
usedmemory           usedmemory
usedswap             -
usedwalltime         usedwalltime
```

References

[1] HTTP(S,G) and SOAP Server/Framework.

[2] NorduGrid middleware, the Advanced Resource Connector. `http://www.nordugrid.org/middleware/`

[3] Configuration and Authorisation of ARC (NorduGrid) Services.