NORDUGRID-TECH-14
21/12/2010

ARC Computational Job Management Component – A-REX

Description and Administrator's Manual

A. Konstantinov*

*aleks@fys.uio.no

Contents

1 Introduction

The A-REX is an ARC middleware component that implements functions of the so-called *Computing Element* (CE). Here Computing Element is a service accepting requests containing a description of generic computational jobs and executing it in the underlying local batch system.

It takes care of job pre- and post-processing, i.e. stage-in of files containing input data or program modules from a wide range of sources and transfer or storing of output results.

In previous versions of ARC, computational tasks (jobs) were submitted to a resource through a GridFTP service running on the CE, and processed by a Grid Manager (GM). The A-REX replaces both these components, by implementing a Web Service (WS) interface which provides a way to submit jobs, and a revised version of the GM to process jobs and interact with the underlying Local Resource Management System. The A-REX can also be set up to process jobs submitted through the traditional ARC GridFTP service, as well as or instead of those using the WS interface. Unless stated otherwise, the remainder of this document assumes running only the WS interface.

> You should use this document for advanced configuration purposes and understanding of the internals of the aforementioned tools. For general installation and configuration of the whole system please refer to other documents available at `http://www.nordugrid.org/papers.html`.

2 Main concepts

On the computing element a job is described as a set of input files (which may include executables), a main executable and a set of output files. The process of gathering input files, executing a job, and transferring/storing output files is called a *session*.

Each job gets a directory on the CE called the *session directory* (SD). Input files are gathered in the SD. The job may also produce new data files in the SD. The A-REX does not guarantee the availability of any other places accessible by the job other than SD (unless such a place is part of a requested Runtime Environment). The SD is also the only place which is controlled by the A-REX. It is accessible by the user from outside through the HTTP(S) protocol. Any file created outside the SD is not controlled by the A-REX. Any exchange of data between client and A-REX (including also program modules) is performed via HTTP(S). A URL for accessing input/output files is obtained through the WS Local Information Description Interface (LIDI) of A-REX.

Each job gets an identifier (*jobid*). This is a handle – WS-Addressing [12] XML document – which identifies the job in the A-REX and in the Information Interface.

Jobs are initiated and controlled through the WS interface. Complete job descriptions (JD) are passed to the A-REX through WS in JSDL [4] coded description. Input data files and job executables are transferred separately through the same interface, as described in Section 3.

3 Input/output data

One of the most important tasks of the A-REX is to take care of processing of the input and output data (files) of the job. Input files are gathered in the SD or in the associated cache area. There are two ways to put a file into the SD:

- Download is initiated by the A-REX – This is the case for files defined in the JD (with name and source). The A-REX alone is responsible to ensure that all required files will be available in the SD. The supported protocols for sources at the moment are (in case of full installation): GridFTP, FTP, HTTP, HTTPS (HTTP over SSLv3). Some less standard sources and *Indexing Services* are also supported, these are described in Section 5. In the case where a file in an indexing service resolves to multiple physical replicas, some selection of the "best" replica may be performed based on any available information on the access latency of those replicas. The A-REX fully relies on the HED framework [10]

for data transferring capabilities and so the actual set of supported protocols depends on the installed Data Management Components of the HED.

- Upload is initiated by the user directly or through the User Interface (UI). Because the SD becomes available immediately at the time of submission of the JD, the UI can (and should) use that to upload data files which are not otherwise accessible by the A-REX. Examples of such files are the main executable of the job, the job's input files, etc. These files can (and should) also be specified in the JD.

There is no other reliable way for a job to obtain input data on the CE based on the A-REX. Access to AFS, NFS, FTP, HTTP and any other remote data transport during execution of a job is not guaranteed (at least not by A-REX).

At the start of a download initiated by the A-REX, a dummy file with the same name and size as the source file is created in the SD to check that enough space is available. As the transfer proceeds this dummy file is over-written with the real data. If the file is to be cached (see Section 7) then the pre-allocation of space is not done, as cache space is managed by the A-REX. In the case of caching two extra validation checks are also performed: firstly, if the source is an Indexing Service then metadata (file size, checksum) reported by the Indexing Service and the service hosting the physical replica are compared, if available. If they differ then that replica is not downloaded. Secondly, on completion of the download the file size is compared to that reported by the source - if they differ then the download is failed.

Jobs should store output files in their SD. Like input files, output files belong into two groups:

- Files which are supposed to be moved to a *Storage Element* (SE) and optionally registered in some Indexing Service like the Globus *Replica Location Service* (RLS) – The A-REX takes care of these files. They have to be specified in the JD. If the job fails during any stage of processing, no attempt is made to transfer those files to their final destination, unless the option *preserve=yes* is specified in their URLs.
- Files which are supposed to be fetched by the user – The user has to use a tool like the UI to obtain these files. They **must** also be specified in the JD.

All files not specified in the JD are deleted after job execution finished. If job execution fails for any reason (if exit code of main executable is not 0) all files from first group are transferred to second one.

4 Job flow

From the point of view of the A-REX a job passes through various states. Figure 1 presents a diagram of the possible states of a job.

A user can examine the state of a job by querying the dedicated Local Information Description Interface of A-REX using the UI or any other suitable tool or through query method of WS interface.

Configuration can put limits on the amount of simultaneous jobs in some states. If such a limit is reached, a job ready to enter into the state in question will stay in it's current state waiting for a free slot. This situation is presented by additional state mark **PENDING** to the current state name in the job's status description.

Below is the description of all actions taken by the A-REX at every state:

- **Accepted** – In this state the job has been submitted to a CE but is not processed yet. The A-REX will analyze the JD and move to the next stage. If the JD can not be processed the job will be canceled and moved to the state **Finishing**.
- **Preparing** – The input data is being gathered in the SD (stage-in). The A-REX is downloading the files specified in the JD and is waiting for files which are supposed to be uploaded by the UI. If all files are successfully gathered the job moves to the next state. If **any** file can't be downloaded or it takes the UI too long to upload a file, the job moves to **Finishing** state. It is possible to put a limit on the number of simultaneous **Preparing** jobs. If this limit is exceeded, jobs ready to enter the

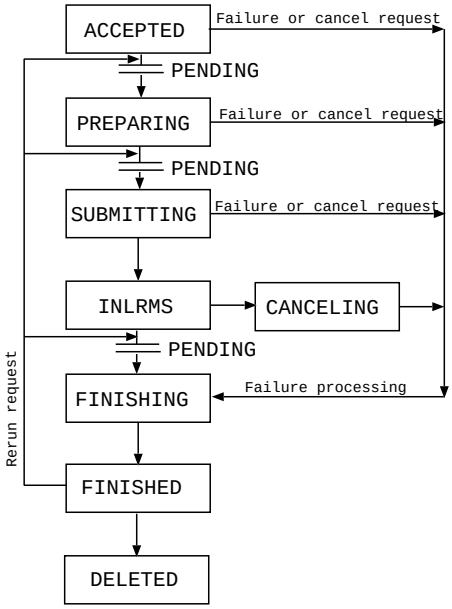

Figure 1: Job states

Preparing state will stay in the **Accepted** state, but prefixed with the PENDING: mark. Exceptions are jobs which have no files to be downloaded. These are processed out of limits. If the A-REX fails to download a file due to a temporary error with a remote service, the job will be moved back to the **Accepted** state, but prefixed with the PENDING: mark, and will wait in that state for some time before being retried. The waiting time increases exponentially with each attempt.

- **Submitting** – The job is being passed for execution to the *Local Resource Management System* (LRMS). The corresponding backends for many LRMSs are provided with the default installation. If the local job submission is successful the job moves to the **Executing** state. Otherwise it moves to **Finishing**. It is possible to limit the aggregate number of jobs in **Submitting** and **Executing** states.
- **Executing (InLRMS)** – The job is queued or being executed in the LRMS. The A-REX takes no actions except waiting until the job finishes.
- **Killing (Canceling)** – Necessary action to cancel the job in the LRMS is being taken.
- **Finishing** – The output data is being processed (stage-out). Specified data files are moved to the specified SEs and are optionally registered at an Indexing Service. The user can download data files from the SD by using the UI or other adequate tool. All the files not specified as output files are removed from the SD at very beginning of this state. It is possible to limit the number of simultaneous jobs in this state. If the A-REX fails to upload a file due to a temporary error with a remote service, the job will be moved back to the **Executing** state, but prefixed with the PENDING: mark, and will wait in that state for some time before being retried. The waiting time increases exponentially with each attempt.
- **Finished** – No more processing is performed by the A-REX. The user can continue to download data files from the SD. The SD is kept available for some time (default is 1 week). After that the job is moved to the state **Deleted**. The 'deletion' time can be obtained by querying the Information Interface of the A-REX. If a job was moved to **Finished** because of failure, it may be restarted on request of a client. When restarted, a job is moved to the state previous to the one in which it failed and is assigned mark PENDING. This is needed in order to not break the configuration limits. Exception is a job failed in **Executing** state and lacking input files specified in JD. Such a job is treated like failed in **Preparing** state.
- **Deleted** – The job is moved to this state if the user have not requested job to be cleaned before the SD's lifetime expires. Only minimal subset of information about such job is kept. The SD is not available anymore.

In case of failure, special processing is applied to output files. All specified output files are treated as **downloadable by the user**. No files will be moved to their destination SE.

5 URLs

In a full installation, the A-REX and its components support the following data transfer protocols and corresponding URLs: *ftp, gsiftp, http, https, lfc, rls* and *srm.* For more information please see "The Hosting Environment of the Advanced Resource Connector middleware" document [10].

6 Internals

6.1 Internal Files of the A-REX

For each local UNIX user listed in the A-REX configuration – including a generic one which covers all local user identities – a *control directory* exists. In this directory the A-REX stores information about jobs belonging to that user. Multiple users can share the same *control directory.* In the most common configuration case, the A-REX serves all users defined by the Operating System and stores their control files in the same directory. To make it easier to recover in case of failure, the A-REX stores most information in files rather than in memory. All files belonging to the same job have names starting with **job.ID.,** where ID is the job identifier.

The files in the control directory and their formats are described below:

- *job.ID.status* – current state of the job. This is a plain text file containing a single word representing the internal name of current state of the job. Possible values and corresponding external job states are:
  - ACCEPTED
  - PREPARING
  - SUBMIT
  - INLRMS
  - FINISHING
  - FINISHED
  - CANCELING
  - DELETED

See Section 4 for a description of the various states. Additionally each value can be prepended the prefix "PENDING:" (like PENDING:ACCEPTED, see Section 4). This is used to show that a job is *ready* to be moved to the next state but it has to stay in it's current state *only* because otherwise some limits set in the configuration would be exceeded.

- *job.ID.description* – contains the description of the job (JD).
- *job.ID.local* – information about the job used by the A-REX. It consists of lines of format *"name = value"*. Not all of them are always available. The following names are defined:
  - *subject* – user certificate's subject, also known as the distinguished name (DN)
  - *starttime* – GMT time when the job was accepted represented in the Generalized Time format of LDAP
  - *lifetime* – time period to preserve the SD after the job has finished in seconds
  - *cleanuptime* – GMT time when the job should be removed from the cluster and it's SD deleted in Generalized Time format
  - *notify* – email addresses and flags to send mail to about the job specified status changes

- *processtime* – GMT time when to start processing the job in Generalized Time format
- *exectime* – GMT time when to start job execution in Generalized Time format
- *expiretime* – GMT time when the credentials delegated to the job expire in Generalized Time format
- *rerun* – number of retries left to rerun the job
- *jobname* – name of the job as supplied by the user
- *projectname* – name of the project as supplied by the user
- *lrms* – name of the LRMS backend to be used for local submission
- *queue* – name of the queue to run the job at
- *localid* – job id in LRMS (appears only after the job has reached state **InLRMS**)
- *globalid* – BES ActivityIdentifier XML tree, the global identifier of the job
- *args* – executable name followed by a list of command-line arguments
- *downloads* – number of files to download into the SD before execution
- *uploads* – number of files to upload from the SD after execution
- *gmlog* – directory name which holds files containing information about the job when accessed through GridFTP interface
- *clientname* – name (as provided by the user interface) and IP address:port of the submitting client machine
- *clientsoftware* – version of software used to submit the job
- *sessiondir* – the job's SD
- *failedstate* – state in which job failed (available only if it is possible to restart the job)
- *jobreport* – URL of a user requested *logger service*. The A-REX will also send job records to this service in addition to the default logger service configured in the configuration.
- *activityid* – Job-id of previous job in case the job has been resubmitted or migrated. This value can appear multiple times if a job has been resubmitted or migrate more than once.
- *forcemigration* – This boolean is only used for migration of jobs. It determines whether the job should persist if the termination of the previous job fails.
- *transfershare* – name of share used in **Preparing** and **Finishing** states.

This file is filled partially during job submission and fully when the job moves from the **Accepted** to the **Preparing** state.

- *job.ID.input* – list of input files. Each line contains 2 values separated by a space. First value contains name of the file relative to the SD. Second value is a URL or a file description. Example:

  input.dat gsiftp://grid.domain.org/dir/input_12378.dat

  A URL represents a location from which a file can be downloaded. Each URL can contain additional options.

  A file description refers to a file uploaded from the UI and consists of [size][.checksum] where

  size - size of the file in bytes.

  checksum - checksum of the file identical to the one produced by ***cksum*** (1).

  These values are used to verify the transfer of the uploaded file. Both size and checksum can be left out. A special kind of file description *.* is used to specify files which are **not** required to exist.

  This file is used by the '***downloader***' utility. Files with *URL* will be downloaded to the SD and files with 'file description' will simply be checked to exist. Each time a new **valid** file appears in the SD it is removed from the list and *job.ID.input* is updated. Any external tool can thus track the process of collecting input files by checking *job.ID.input*.

- *job.ID.output* – list of output files. Each line contains 1 or 2 values separated by a space. First value is the name of the file relative to the SD. The second value, if present, is a URL. Supported URLs are the same as those supported by job.ID.input.

  This file is used by the '***uploader***' utility. Files with *URL* will be uploaded to SE and remaining files will be left in the SD. Each time a file is uploaded it is removed from the list and *job.ID.output* is updated. Files not mentioned as output files are removed from the SD at the the beginning of the **Finishing** state.

- *job.ID.failed* – the existence of this file marks the failure of the job. It can also contain one or more lines of text describing the reason of failure. Failure includes the return code different from zero of the job itself.

- *job.ID.errors* – this file contains the output produced by external utilities like ***downloader***, ***uploader***, script for job submission to LRMS, etc on their stderr handle. Those are not necessarily errors, but can be just useful information about actions taken during the job processing. In case of problem include content of that file while asking for help.

- *job.ID.diag* – information about resources used during execution of job and other information suitable for diagnostics and statistics. It's format is similar to that of *job.ID.local*. The following names are at least defined:

  - *nodename* – name of computing node which was used to execute job,
  - *runtimeenvironments* – used runtime environments separated by ';',
  - *exitcode* – numerical exit code of job,
  - *frontend_distribution* – name and version of operating system distribution on frontend computer,
  - *frontend_system* – name of operating on frontend computer,
  - *frontend_subject* – subject (DN) of certificate representing frontend computer,
  - *frontend_ca* – subject (DN) of issuer of certificate representing frontend computer,

  and other information provided by GNU *time* utility. Note that some implementations of *time* insert unrequested information in their output. Hence some lines can have broken format.

- *job.ID.proxy* – delegated X509 credentials.

- *job.ID.proxy.tmp* – temporary X509 credentials with different UNIX ownership used by processes run with effective *user id* different from job owner's *id*.

There are other files with names like job.ID.* which are created and used by different parts of the A-REX. Their presence in the *control directory* can not be guaranteed and can change depending on changes in the A-REX code.

6.2 Web Service Interface

A-REX Web Service Interface provides means to submit a description of a computational job to a computing resource, to stage-in additional data, to monitor and control processing of jobs, and obtain data produced during the execution of a job. The WS Interface is built and deployed inside the Hosting Environment Daemon (HED) infrastructure [10].

6.2.1 Basic Execution Service Interface

The job submission and control interface is based on a document produced by the OGF OGSA Basic Execution Services (BES) Working Group [7].

The exchange of SOAP messages is performed via HTTP(S). The BES interface is represented by two port-types – BES-Management and BES-Factory. The former is made to control the A-REX service itself and thus defines operations to start and stop the functionality of the BES service. The A-REX does not implement remote control of service functionality. Hence the BES-Management port-type is not functional. The BES-Factory port-type provides operations to submit new jobs (to create an activity in terms of BES) and to

monitor its state. It also has an ability to provide information about the service. A-REX fully implements the functionality of this port-type.

For job descriptions A-REX accepts the Job Submission Description Language (JSDL) [4] documents as defined by the OGF Job Submission Description Language Working Group. Supported elements and extensions are described below.

6.2.2 Extensions to OGSA BES interface

A-REX introduces two new operations in addition to those provided by BES. It does that by defining its own port-type with new operations *ChangeActivityStatus* and *MigrateActivity*(see Appendix C).

The *ChangeActivityStatus* operation provides a way to request simple transfers between states of jobs and corresponding actions.

- *ChangeActivityStatus*
  - Input
    - *ActivityStatusType OldStatus*: Description of the state the job is supposed to be in during execution of this request. If the current state of the job is different from the one having been given, the operation is aborted and a fault is returned. This parameter is optional.
    - *ActivityStatusType NewStatus*: Description of the state the job is to be put into.
  - Output
    - *ActivityStatusType NewStatus*: Description of the current state of the job.
  - Fault(s)
    - *NotAuthorizedFault*: Indicates that the client is not allowed to do this operation.
    - *InvalidActivityIdentifierFault*: There is no such job/activity.
    - *CantApplyOperationToCurrentStateFault*: The requested transition is not possible.

  On result of this command, the job should be put into the requested state. If such a procedure cannot be performed immediately then the corresponding sequence is initiated and fault OperationWillBeAppliedEventuallyFault will be returned.

  Since BES allows implementations to extend their initial activity states with additional sub-states, A-REX defines a set of sub-states of activity processing in addition to those defined by the BES, as listed in Table 1. Their meaning is described in Section 4.

  The *MigrateActivity* operation generates a request to migrate a grid job from another A-REX, i.e. the operation will get input files and possibly job description from the cluster currently holding the job and create the job as a new activity at the present cluster. Currently only migration of queuing jobs is supported.

- *MigrateActivity*
  - Input
    - *wsa:EndpointReferenceType ActivityIdentifier*: This element should contain the *wsa:EndpointReference* of the job to be migrated.
    - *ActivityDocument*: JSDL document of the job to be migrated. This element is optional.
    - *Boolean ForceMigration*: Boolean that determines whether the job will persist on the new cluster if the termination of the previous job fails.
  - Output
    - *wsa:EndpointReferenceType ActivityIdentifier*: This element should contain the *wsa:EndpointReference* of the new activity.
    - *ActivityDocument*: Contains the JSDL document of the new activity.
  - Fault(s)
    - *NotAuthorizedFault*: Indicates that the client is not allowed to do this operation.
    - *NotAcceptingNewActivitiesFault*: A fault that indicates that A-REX currently is not accepting new activities.

        * *UnsupportedFeatureFault*: This fault indicates that an sub-element in the JDSL document is not supported or the ActivityDocument has not been recognised as JSDL.
        * *InvalidRequestMessageFault*: This fault indicates that an element in the request is either missing or has an invalid format. Typically this would mean that the job-id cannot be located in the ActivityIdentifier of the old job.

The *ActiviterIdentifier* specifies the URL of the job which will be migrated. In case the *ActivityDocument* is filled this document will be used to create a new activity otherwise an attempt will be made to retrieve the job description through the BES operation *GetActivityDocument*.

Once the input files have been downloaded from the other cluster, a request will be send to terminate the old job. If this request fails the new activity at the present cluster will be terminate unless the *ForceMigration* is true. This is to prevent the job from being executed at two different places at the same time.

6.2.3 Delegation Interface

The A-REX also supports the Delegation Interface (see Appendix D). This is a common purpose interface to be used by ARC services which accepts delegated credentials from clients. The Delegation Interface implements two operations: initialization of credentials delegation (DelegateCredentialsInit) and update/renewal of credentials (UpdateCredentials).

- *DelegateCredentialsInit* operation – this operation performs the first half of the credentials delegation sequence.
    - Input
        * None. On this request the service generates a pair of *public* and private keys. The public key is then sent to the client in response.
    - Output(s)
        * *TokenRequestType TokenRequest*: Contains the public key generated by the service as a Value element. It also provides an identifier in the Id element which should be used to refer to the corresponding private key.
    - Fault(s)
        * *UnsupportedFault*: Indicates that the service does not support this operation despite supporting the port-type.
        * *ProcessingFault*: Internal problems during generation of the token.
- *UpdateCredentials* operation – this operation makes it possible to update the content of delegated credentials (like in the case of credentials being renewed) unrelated to other operations of the service.
    - Input
        * *DelegatedTokenType DelegatedToken*: Contains an X509 proxy certificate based on the public key from the DelegateCredentialsInit signed by the user's proxy certificate. Also includes the Id element which identifies the private key stored at the service side associated with these credentials. The reference element refers to the object to which these credentials should be applied in a way specific to the service. The same element must also be used for delegating credentials as part of other operations on service.
    - Output(s)
        * None.
    - Fault(s)
        * *UnsupportedFault*: Indicates that service does not support this operation despite supporting the port-type.
        * *ProcessingFault*: Internal problems during generation of the token.

Additionally, A-REX Web Service Interface allows delegation to be performed as part of the *CreateActivity* operation of the BES-Factory port-type. For this it accepts the element *DelegatedCredentials* inside the *CreateActivity* element. The *Id* element of *DelegatedCredentials* must contain an identifier obtained in response to the previous *DelegateCredentialsInit* operation. For more information about delegations and delegation interface refer to [9].

Table 1: Job states definitions and mappings

Applicable BES State	ARC Sub-state	A-REX internal state	Description
Pending	Accepting	ACCEPTED	Job is in the process of being submitted. This state is not recognised by the A-REX yet. *Accepted* is first reported state
	Accepted	ACCEPTED	Job was submitted
Running	Preparing	PREPARING	Stage-in process is going on
	Prepared	PREPARING + PENDING	Stage-in process has finished
	Submitting	SUBMIT	Communication with local batch system is in process
	Queued	INLRMS	Job entered local batch system but is not runnning now. This state is not recognised by the A-REX yet. *Executing* is reported instead
	Executing	INLRMS	Job is being executed in local batch system
	Executed	INLRMS, INLRMS + PENDING	Job execution in local batch system has finished. The A-REX dos not detect job states inside local batch system yet. As result this state is reported only if job is *Pending*.
	Killing	CANCELING	Communication with local batch system to terminate execution is in process
	Finishing	FINISHING	Stage-out process is going on
Cancelled	Killed	FINISHED	Job was stopped by explicit user request. The A-REX currently does not remember this request. *Failed* is reported instead.
Failed	Failed	FINISHED	There was a failure during execution
Finished	Finished	FINISHED	Job finished successfully
Finished	Deleted	DELETED	Job finished and was left in A-REX too long
All	Pending	PENDING	Job is prevented from going to the next state due to some internal limits; this sub-state appears in parallel with other sub-states
All	Held		Job processing is suspended on client request; this sub-state appears in parallel with other sub-states. This state is reserved for future and is not implemented yet.

6.2.4 Local Information Description Interface

The A-REX implements the Local Information Description Interface (LIDI) interface common for all ARC services. This interface is based on OASIS Web Services Resource Properties specification [9]. Information about resources and maintained activities/jobs are represented in a *WS-Resource Properties* informational XML document. The document type is defined in the A-REX WSDL as a *ResourceInformationDocument-Type*. It contains the following elements/resources:

nordugrid – description of computing resource that uses NorudGrid LDAP schema [5] converted to XML document.

Domains – description of a computation resource that uses Glue2 schema.

All information can be accessed either through requests on particular resources or through XPath queries using WS-Resource Properties operations.

6.2.5 Supported JSDL elements

A-REX supports the following elements from the JSDL version 1.0 specification [4] including POSIX Applications extension and JSDL HPC Profile Application Extension [8]:

JobName – name of the job as assigned by the user.

Executable (POSIX,HPC) – name of the executable file.

Argument (POSIX,HPC) – arguments the executable will be launched with.

DataStaging

Filename – name of the data file on the executing node.

Source – source where the file will be taken from before execution.

Target – destination the file will be delivered to after execution.

Input (POSIX,HPC) – file to be used as standard input for the executable.

Output (POSIX,HPC) – file to be used as standard output for the executable.

Error (POSIX,HPC) – file to be used as standard error for the executable.

MemoryLimit (POSIX) – amount of physical memory needed for execution.

TotalPhysicalMemory – same as MemoryLimit.

IndividualPhysicalMemory – same as MemoryLimit.

CPUTimeLimit (POSIX) – maximal amount of CPU time needed for execution.

TotalCPUTime – same as CPUTimeLimit.

IndividualCPUTime – same as CPUTimeLimit.

WallTimeLimit (POSIX) – amount of clock time needed for execution.

TotalCPUCount – number of CPUs needed for execution.

IndividualCPUCount – same as *TotalCPUCount*.

6.2.6 ARC-specific JSDL Extensions

A-REX accepts JSDL documents having the following additional elements (see Appendix E):

IsExecutable – marks file to become executable after being delivered to the computing resource.

RunTimeEnvironment – specifies the name of the Runtime Environment needed for job execution.

Middleware – request for specific middleware on the computing resource frontend.

RemoteLogging – destination for the usage record report of the executed job.

LocalLogging – name for the virtual directory available through job interface and containing various debug information about job execution.

AccessControl – ACL expression which describes the identities of those clients who are allowed to perform operations on this job.

Notify – Email destination for notification of job state changes.

SessionLifeTime – duration for the directory containing job-related files to exist after the job finished executing.

JoinOutputs – specifies if standard output and standard error channels must be merged.

Reruns – defines how many times a job is allowed to rerun in case of failure.

CredentialServer – URL of MyProxy service which may be used for renewing the expired delegated job credentials.

CandidateTarget – specifies host name and queue of a computing resource.

OldJobID – specifies the previous job-ids in case the job has been resubmitted or migrated.

7 Cache

The A-REX can cache input files, so that subsequent jobs requiring the same files do not have to download them again. Caching is enabled if one or more cache directories are specified in the configuration file. All input files except files uploaded by the user during job submission are cached by default. This includes executable files downloaded by the A-REX. Caching can be explicitly turned off by the user in the job description (see [11]). The disk space occupied by the cache is controlled by removing files in the order of least recent access. For more information on configuration see Section 9.1.

7.1 Structure

Cached files are stored in sub-directories under the *data* directory in each main cache directory. Filenames are constructed from an SHA-1 hash of the URL of the file and split into subdirectories based on the two initial characters of the hash. In the extremely unlikely event of a collision between two URLs having the same SHA-1 hash, caching will not be used for the second file.

When multiple caches are used, a new cache file goes to a randomly selected cache, where each cache is weighted according to the size of the file system on which it is located. For example: if there are two caches of 1TB and 9TB then on average 10% of input files will go to the first cache and 90% will go to the second cache.

Some associated metadata including the corresponding URL and an expiry time, if available, are stored in a file with the same name as the cache file, with a *.meta* suffix.

For example, with a cache directory */cache*, the file

lfc://atlaslfc.nordugrid.org//grid/atlas/file1
is mapped to
/cache/data/78/f607405ab1df6b647fac7aa97dfb6089c19fb3

and the file */cache/data/78/f607405ab1df6b647fac7aa97dfb6089c19fb3.meta* contains the original URL and an expiry time if one is available.

At the start of a file download, the cache file is locked, so that it cannot be deleted and so that another download process cannot write the same file simultaneously. This is done by creating a file with the same name as the cache filename but with a *.lock* suffix. This file contains the process ID of the process and the hostname of the host holding the lock. If this file is present, another process cannot do anything with the cache file and must wait until the cache file is unlocked (i.e. the *.lock* file no longer exists). The lock has a timeout of one day, so that stale locks left behind by a download process exiting abnormally will eventually be cleaned up. Also, if the process corresponding to the process ID stored inside the lock is no longer running on the host specified in the lock, it is safe to assume that the lock file can be deleted.

7.2 How it works

If a job requests an input file which can be cached or is allowed to be cached, it is stored in the selected cache directory, and depending on the configuration, either the file is copied to the SD or a hard link is created in a per-job directory and a soft link is created in the SD to there. The per-job directories are in the *joblinks* subdirectory of the main cache directory. The former option is advised if the cache is on a file system which will suffer poor performance from a large number of jobs reading files on it, or the file system containing the cache is not accessible from worker nodes. The latter option is the default option. Files marked as executable in the job will be stored in the cache without executable permissions, but they will be copied to the SD and the appropriate permissions applied to the copy.

The per-job directory is only readable by the local user running the job, and the cache directory is readable only by the A-REX user. This means that the local user cannot access any other users' cache files. It also means that cache files can be removed without needing to know whether they are in use by a currently running job. However, as deleting a file which has hard links does not free space on the disk, cache files are not deleted until all per-job hard links are deleted. **IMPORTANT:** If a cache is mounted from an NFS server and the A-REX is run by the root user, the server must have the *no_root_squash* option set for the A-REX host in the */etc/exports* file, otherwise the A-REX will not be able to create the required directories. Note that when running A-REX under a non-privileged user account, all cache files will be owned and accessible by the same user, and therefore modifyable by running jobs. This is potentially dangerous and so cacheing should be used with caution in this case.

If the file system containing the cache is full and it is impossible to free any space, the download fails and is retried without using cacheing.

Before giving access to a file already in the cache, the A-REX contacts the initial file source to check if the user has read permission on the file. In order to prevent repeated checks on source files, this authentication information is cached for a limited time. On passing the check for a cached file, the user's DN is stored in the *.meta* file, with an expiry time equivalent to the lifetime remaining for the user's proxy certificate. This means that the permission check is not performed for this user for this file until this time is up (usually several hours). File creation and validity times from the original source are also checked to make sure the cached file is fresh enough. If the modification time of the source is later than that of the cached file, the file will be downloaded again. The file will also be downloaded again if the modification date of the source is not available, as it is assumed the cache file is out of date. These checks are not performed if the DN is cached and is still valid.

The A-REX checks the cache periodically. If the used space on the file system containing the cache exceeds the high water-mark given in the configuration file it tries to remove the least-recently accessed files to reduce size to the low water-mark.

7.3 Remote Caches

If a site has multiple A-REXs running, an A-REX can be configured to have its own caches and have read-only access to caches under the control of other A-REXs (remote caches). An efficient way to reduce network traffic within a site is to configure A-REXs to be under control of caches on their local disks and have caches on other hosts as remote caches. If an A-REX wishes to cache a file and it is not available on the local cache, it searches for the file in remote caches. If the file is found in a remote cache, the actions the A-REX takes depends on the policy for the remote cache. The file may be replicated to the local cache to decrease the

load on the remote file system caused by many jobs accessing the file. However, this will decrease the total number of cache files that can be stored. The other policy is to use the file in the remote cache, creating a per-job directory for the hard link in the remote cache. Then the link is created from the session dir to that directory, bypassing the local cache completely. The usual permission and validity checks are performed for the remote file. Note that no creation or deletion of remote cache data is done - cache cleaning is only performed on local caches.

7.4 Cache Administration

The cache is cleaned automatically periodically (every 2 minutes) by the A-REX to keep the size of each cache within the configured limits. Files are removed from the cache if the total size of the cache is greater than the configured limit. Files which are not locked are removed in order of access time, starting with the earliest, until the size is lower than the configured lower limit. If the lower limit cannot be reached (because too many files are locked, or other files outside the cache are taking up space on the file system), the cleaning will stop before the lower limit is reached.

Since the limits on cache size are given as a percentage of space used on the filesystem on which the cache is located, it is recommended that each cache has its own dedicated file system. If the cache shares space with other data on a file system, changes in the amount of non-cache data will result in changes in the available cache space.

With large caches mounted over NFS and an A-REX heavily loaded with data transfer processes, cache cleaning can become slow, leading to caches filling up beyond their configured limits. For performance reasons it may be advantageous to disable cache cleaning by the A-REX, and run the *cache-clean* tool independently on the machine hosting the file system.

Caches can be added to and removed from the configuration as required without affecting any cached data, but after changing the configuration file, the A-REX should be restarted. If a cache is to be removed and all data erased, it is recommended that the cache be put in a *draining* state until all currently running jobs possibly accessing files in this cache have finished. In this state the cache will not be used by any new jobs, but the hard links in the *joblinks* directory will be cleaned up as each job finishes. Once this directory is empty it is safe to delete the entire cache. See the *cachedir* option in Section 9.1 for how to set a cache to a draining state.

The following tools (installed in *$ARC_LOCATION/libexec/arc*) exist to help with administration of the cache:

- *cache-clean* - This tool is used periodically by the A-REX to keep the size of each cache within the configured limits.
  cache-clean -h gives a list of options. The most useful option for administrators is *-s*, which does not delete anything, but gives summary information on the files in the cache, including information on the ages of the files in the cache.
  It is not recommended to run *cache-clean* manually to clean up the cache, unless it is desired to temporarily clean up the cache with different size limits to those specified in the configuration, or to improve performance by running it on the file system's local node as mentioned above.
- *cache-list* - This tool is used to list all files present in each cache or, given a list of URLs as arguments, shows the location of each URL in the cache if present. In the first case it simply reads through all the *.meta* files and prints to stdout a list of all URLs stored in each cache and their corresponding cache filename, one per line. In the second case the cache filename of each URL is calculated and then each cache is checked for the existence of the file.

8 Files and directories

8.1 Modules

The A-REX consists of several separate modules. These are:

- *libarex.so* – The main module providing main functionality and web interface. It is implemented as HTTP and SOAP service inside HED. It is responsible for processing jobs, moving them through states and running other modules.
- *downloader* – This is a module responsible for gathering input files in the SD. It processes the *job.ID.input* file and updates it.
- *uploader* – This module is responsible for delivering output files to the specified SEs and registration at an Indexing Service (like RLS) as needed. It processes and updates the *job.ID.output* file.
- *gm-kick* – Sends a signal to the A-REX though a FIFO file to wake it up. It's used to increase responsiveness of A-REX.
- *CEinfo.pl* – Collects and generates information about computing resource as XML document in NorduGrid and Glue 2 format.

The following modules are always run under the Unix account to which a Grid user is mapped.

- *smtp-send.sh* and *smtp-send* – These are the modules responsible for sending e-mail notifications to the user. The format of the mail messages can be easily changed by editing the simple shell script *smtp-send.sh*.
- *submit-*-job* – Here * stands for the name of the LRMS. Currently supported LRMS are PBS/Torque, Condor, LoadLeveler, LSF, SLURM, and SGE. Also *fork* pseudo-LRMS is supported for testing purposes. This module is responsible for job submission to the LRMS.
- *cancel-*-job* – This script is for canceling jobs which have been already submitted to the LRMS.
- *scan-*-job* -This shell script is responsible for notifying the A-REX about completion of jobs. It's implementation for PBS uses server logs to extract information about jobs. If logs are not available it uses the less reliable *qstat* command for that. Other backends use different techniques.

In addition, there is also an administration utility:

- *gm-jobs* – prints a list of jobs available on the cluster and the number of jobs in each state.
  gm-jobs [-h] [-s] [-l] [-u uid] [-U name] [-c conf_file] [-d control_dir]
  -h – print short help,
  -s – print summary of jobs in each transfer share,
  -l – print more information about each job,
  -u – pretend utility is run by user with id *uid*,
  -U – pretend utility is run by user with name *name*,
  -c – use specified configuration file,
  -d – read information from specified control dir.

8.2 Directories

The A-REX is installed into a single installation point referred as $ARC_LOCATION and the following sub-directories are used:

$ARC_LOCATION/bin – tools

$ARC_LOCATION/libexec – program modules used by A-REX

/etc – central configuration file – location used by default

$ARC_LOCATION/lib/arc – service module

The A-REX also uses following directories:

- *session root directory* – This is the directory in which a user's SDs are created. It's location is configurable per UNIX user. Several (or even all) users may share the same session root directory.
  The A-REX needs to have permission to create new files and directories in the session root directory. If A-REX is run under a dedicated user account, that account needs full permissions in the *session root directory*.
  If A-REX is run under the *root* account, make sure *session root directory* resides on a file system which does not limit the capabilities of the *root* user (as does for example NFS with *root_squash* option).
  If there is a need to run A-REX under the *root* account (to be able to run jobs in LRMS under different users' accounts, for example) but there is no way to provide a suitable *session root directory,* use the *norootpower* command in configuration file. In that case A-REX will use the identity of the local user to which a Grid identity is mapped to access the *session root directory*. Hence those users will need full access there.
  The A-REX creates SDs with proper ownership and permissions for the local identity used to run a job. Some file systems require users to have *execute* permission on the *session root directory* in order to access any file or subdirectory there.
  In order for jobs to access their input files, session root directories should be shared across cluster nodes. Otherwise, LRMS-specific methods must be used to transfer files to execution nodes.

- *control directory* – In this directory A-REX stores information about accepted jobs. It must have full permissions there.
  A subdirectory called *logs* is created there. It is used to accumulate information about started and finished jobs. This information is periodically sent to the desired *logger service*(s). For each job start and stop event, and for each logger service where that event must be sent, a separate file is written. Once an event is sent, the corresponding file is deleted.
  Some utility files, which contain for example cached information on remote services, may also be found in the control directory. These files have a *.conf* suffix.

9 Configuration

9.1 Configuration of the A-REX

Due to historical reasons, configuration of the A-REX is split into 2 parts: HED configuration for the WS interface and a legacy configuration file for the rest. For more information on the HED configuration see instructions in Section 10.2, and Appendix B for a full schema and description of supported elements. The HED configuration refers to the legacy file, which is identical to the configuration file used for the GridFTP/Grid Manager services in previous versions of ARC. The default location of this legacy file is */etc/arc.conf*. If the A-REX is run without the WS interface, then only the legacy file is needed (to configure the GridFTP interface see [2]). This section describes the content of the legacy file relating to running the A-REX.

The configuration file can contain empty lines and comments in lines starting with #. It is separated into sections. Each section starts with a string containing

```
[section name/subsection name/subsubsection name]
```

Each section continues until the next section or until the end of the file. The configuration file can have commands for multiple services/modules/programs. Each service has its own section named after it. The A-REX uses the *[grid-manager]* section. Some services can make use of multiple subsections to reflect their internal modular structure. Commands in section *[common]* apply to all services. Command lines have the format

```
name=‘‘arguments string’’
```

The following commands are defined:

Commands affecting the A-REX process and logging:

- ***pidfile***=*path* – specifies file where process id of A-REX process will be stored. Defaults to */var/log/arched.pid* if running as root and *$HOME/arched.pid* otherwise.
- ***logfile***=*path* – specifies name of file for logging debug/informational output. Defaults to */var/log/arc/grid-manager.log*. If installed from packages, the default log is managed by logrotate.
- ***logsize***=*size number* – restricts log file size to *size* and keeps *number* archived log files.
- ***debug***=*number* – specifies level of debug information. More information is printed for higher levels. Currently the highest effective number is 5 (DEBUG) and lowest 0 (FATAL). Defaults to 2 (WARNING).
- ***user***=*username* – specifies username to which the A-REX must switch after reading configuration. Defaults to *not switch*.

Commands setting limits and options for how the A-REX handles jobs and files:

- ***joblog***=*path* – specifies where to store log file containing information about started and finished jobs.
- ***jobreport***=*URL ... number* – specifies that A-REX has to report information about jobs being processed (started, finished) to a centralized service running at the given *URL*. Multiple entries and multiple URLs are allowed. *number* specifies how long (in days) old records have to be kept if failed to be reported. The last specified value becomes effective.
- ***jobreport_credentials***=*key_file [cert_file [ca_dir]]* – specifies the credentials for accessing the accounting service.
- ***jobreport_options***=*options* – specifies additional options for Usage Reporter and/or accounting service. The *options* string is interpreted by Usage Reporter, its format is described in the corresponding technical document.
- ***securetransfer***=*yes—no* – specifies whether to use encryption while transferring data. Currently works for GridFTP only. Default is *no*. It is overridden by values specified in URL options.
- ***passivetransfer***=*yes—no* – specifies whether GridFTP transfers are passive. Setting this option to yes can solve transfer problems caused by firewalls. Default is no.
- ***localtransfer***=*yes—no* – specifies whether to pass file downloading/uploading task to computing node. If set to yes the A-REX will not download/upload files but compose script submitted to the LRMS in order that the LRMS can execute file tranfer. This requires installation of A-REX and all related software to be accessible from computing nodes and environment variable ARC_LOCATION to be set accordingly. Default is *no*.
- ***maxjobs***=*[max_processed_jobs [max_running_jobs [max_jobs_per_dn [max_jobs_total]]]]* – specifies maximum number of jobs being processed by the A-REX at different stages:
  max_processed_jobs – maximum number of concurrent jobs processed by A-REX. This does not limit the amount of jobs which can be submitted to the cluster.
  max_running_jobs – maximum number of jobs passed to Local Resource Management System
  max_jobs_per_dn – maximum number of concurrent jobs processed by A-REX per user DN. If this option is used the total maximum number of jobs processed is still *max_processed_jobs*.
  max_jobs_total – total maximum number of jobs associated with service. It is advised to use this limit only inexceptional case because it also accounts for finished jobs.

  Missing value or -1 means no limit.
- ***maxload***=*[max_frontend_jobs [emergency_frontend_jobs [max_transferred_files]]]* – specifies maximum load caused by jobs being processed on frontend:
  max_frontend_jobs – maximum number of jobs in PREPARING and FINISHING states (downloading and uploading files). Jobs in these states can cause a heavy load on the A-REX host. This limit is applied before moving jobs to PREPARING and FINISHING states.
  emergency_frontend_jobs – if limit of *max_frontend_jobs* is used only by PREPARING or by FINISHING jobs, aforementioned number of jobs can be moved to another state. This is used to avoid the case

where jobs cannot finish due to a large number of recently submitted jobs.
max_transferred_files – maximum number of files being transferred in parallel by every job. Used to decrease load on not so powerful frontends.
Missing value or -1 means no limit.

- ***maxloadshare****=max_share share_type* – specifies a sharing mechanism for data transfer. *max_share* is the maximum number of processes that can run per transfer share and *share_type* is the scheme used to assign jobs to transfer shares. See Section 9.2 for possible values and more details.

- ***share_limit****=name limit* – specifies a transfer share that has a number of processes different from the default value in maxloadshare. *name* is the name of the share and *limit* is the number of processes for this share. In the configuration should appear after maxloadshare. Can be repeated several times for different shares. See Section 9.2 for how to compose shares' names and more details.

- ***wakeupperiod****=time* – specifies how often the A-REX checks for job state changes (like new arrived job, job finished in LRMS, etc.). *time* is a minimal time period specified in seconds. Default is *3 minutes*. The A-REX may also be woken up by external processes such as LRMS scripts before this time period expires.

- ***authplugin****=state options plugin* – specifies *plugin* (external executable) to be run every time job is about to switch to *state*. The following states are allowed: ACCEPTED, PREPARING, SUBMIT, FINISHING, FINISHED and DELETED. If exit code is not 0 job is canceled by default. *Options* consists of *name=value* pairs separated by commas. The following *name*s are supported:
timeout – specifies how long in seconds execution of the plugin allowed to last (mandatory, "*timeout=*" can be skipped for backward compatibility).
onsuccess, *onfailure* and *ontimeout* – defines action taken in each case (*onsuccess* happens if exit code is 0). Possible actions are:
pass – continue execution,
log – write information about result into log file and continue execution,
fail – write information about result into log file and cancel job.

- ***localcred****=timeout plugin* – specifies *plugin* (external executable or function in shared library) to be run every time job has to do something on behalf of local user. Execution of *plugin* may not last longer than *timeout* seconds. If *plugin* looks like *function@path* then function *int function(char*,char*,char*,...)* from shared library *path* is called (*timeout* is not functional in that case). If exit code is not 0 current operation will fail.

- ***norootpower****=yes/no* – if set to *yes* all processes involved in job management will use local identity of a user to which Grid identity is mapped in order to access file system at path specified in ***session*** command (see below). Sometimes this may involve running temporary external process.

- ***speedcontrol****=min_speed min_time min_average_speed max_inactivity* – specifies how long/slow data transfer is allowed to take place. Transfer is canceled if transfer rate (bytes per second) is lower than *min_speed* for at least *min_time* seconds, or if average rate is lower than *min_average_speed*, or no data is received for longer than *max_inactivity* seconds. To allow statistics to build up, no transfers will be stopped within the first 3 minutes.

- ***preferredpattern****=pattern* - specifies a preferred pattern on which to sort multiple replicas of an input file. It consists of one or more patterns separated by a pipe character (|) listed in order of preference. Replicas will be ordered by the earliest match. If the dollar character ($) is used at the end of a pattern, the pattern will be matched to the end of the hostname of the replica.

- ***copyurl****=template replacement* – specifies that URLs starting from *template* should be accessed in a different way (most probably Unix open). The *template* part of the URL will be replaced with *replacement*. *replacement* can be either URL or local path starting from '/'. It is advisable to end template with '/'.

- ***linkurl****=template replacement [node_path]* – mostly identical to *copyurl* but file will not be copied. Instead a soft-link will be created. *replacement* specifies the way to access the file from the frontend, and is used to check permissions. The *node_path* specifies how the file can be accessed from computing nodes, and will be used for soft-link creation. If *node_path* is missing, *local_path* will be used instead. Neither *node_path* nor *replacement* should be URLs.

NOTE: URLs which fit into *copyurl* or *linkurl* are treated as more easily accessible than other URLs. That means if A-REX has to choose between several URLs from which should it download input file, these will be tried first.

Per UNIX user commands

- ***mail***=*e-mail_address* – specifies an email address **from** which notification mails are sent.
- ***defaultttl***=*ttl [ttr]* – specifies the time in seconds for the SD to be available after job finishes (*ttl*) and after job was deleted (*ttr*) due to *ttl*. Defaults are 7 days for *ttl* and 30 days for *ttr*. The minumum value for both parameters is 2 hours.
- ***lrms***=*default_lrms_name default_queue_name* – specifies names for the LRMS and queue. Queue name can also be specified in the JD (currently it is not allowed to override LRMS by using the JD).
- ***sessiondir***=*path [drain]* - specifies the path to the directory in which the SD is created. Multiple session directories may be specified by specifying multiple *sessiondir* commands. In this case jobs are spread evenly over the session directories. If the path is * the default sessiondir is used - *$HOME/.jobs*. When adding a new session directory, ensure to restart the A-REX so that jobs assigned there are processed. A session directory can be drained prior to removal by adding the *"drain"* option (no restart is required in this case). No new jobs will be assigned to this session directory but running jobs will still be accessible. When all jobs are processed and the session directory is empty, it can be removed and the A-REX should be restarted.
- ***cachedir***=*path [link_path]* – specifies a directory to store cached data (see section 7). Multiple cache directories may be specified by specifying multiple *cachedir* commands. Cached data will be distributed over multiple caches according to free space in each. Specifying no *cachedir* command or commands with an empty path disables caching. The optional *link_path* specifies the path at which *path* is accessible on computing nodes, if it is different from the path on the A-REX host. If *link_path* is set to '.' files are not soft-linked, nor are per-job links created, but files are copied to the session directory. If a cache directory needs to be drained, then *cachedir* should specify *"drain"* as the *link_path*.
- ***remotecachedir***=*path [link_path]* - specifies caches which are under the control of other A-REXs, but which this A-REX can have read-only access to (see Section 7.3). Multiple remote cache directories may be specified by specifying multiple *remotecachedir* commands. If a file is not available in paths specified by cachedir, the A-REX looks in remote caches. *link_path* has the same meaning as in *cachedir*, but the special path "replicate" means files will be replicated from remote caches to local caches when they are requested.
- ***cachesize***=*high_mark [low_mark]* – specifies high and low watermarks for space used on the file system on which the cache directory is located, as a percentage of total file system capacity. When the max is exceeded, files will be deleted to bring the used space down to the min level. It is a good idea to have each cache on its own separate file system. To turn off this feature, "cachesize" without parameters can be specified. These cache settings apply to all caches specified by *cachedir* commands.
- ***cachelifetime***=*lifetime* - if cache cleaning is enabled, files accessed less recently than the *lifetime* time period will be deleted. Example values of this option are 1800, 90s, 24h, 30d. When no suffix is given the unit is seconds.
- ***cacheloglevel***=*number* – specified the level of logging by the *cache-clean* tool, between 0 (FATAL) and 5 (DEBUG). Defaults to 3 (INFO).
- ***maxrerun***=*number* – specifies maximal number of times job will be allowed to rerun after it failed at any stage. Default value is *5*. This only specifies a upper limit. The actual number is provided in job description and defaults to 0.
- ***maxtransfertries***=*number* – specifies the maximum number of times download and upload will be attempted per job (retries are only performed if an error is judged to be temporary). This number must be greater than 0 and defaults to 10.

All per-user commands should be put before the *control* command which initiates serviced user.

- ***control****=path username [username [...]]* – This option initiates UNIX user as being serviced by the A-REX. The *path* refers to the control directory (see Section 6 for the description of control directory). If the path is * the default one is used – `$HOME/.jobstatus`. The *username* stands for UNIX name of the local user. Multiple names can be specified. If the name starts from @ rest is treated as path to file containing list serviced users. Usernames are specified one per line and may be optionally prepended with Grid identity of user - last one is ignored. That is done for compatibility with so-called grid-mapfile (for more information please see the description of Globus project [6]). Also the special name '.'(dot) can be used. Corresponding control directory will be used for **any** user. This option should be the last one in the configuration file. There is also command ***controldir****=path*. It presumes special username '.' and is always executed last independent of its placement in file.
- ***helper****=username command [argument [argument [...]]]* – associates an external program with the local UNIX user. This program will be kept running under account of the user specified by *username*. Special names can be used: '*' – all names from /etc/grid-security/grid-mapfile, '.' - root user. The user should be already configured with *control* option (except root, who is always configured). *command* is an executable and *argument*s are passed as arguments to it.

The following are global commands specific to communication with the underlying LRMS.

- ***gnu_time****=path* – path to *time* utility.
- ***tmpdir****=path* – path to directory for temporary files.
- ***runtimedir****=path* – path to directory which contains *runtimenvironment* scripts.
- ***shared_filesystem****=yes—no* – if computing nodes have an access to session directory through a shared file system like NFS.
- ***nodename****=command* – command to obtain hostname of computing node.
- ***scratchdir****=path* – path on computing node where to move session directory before execution.
- ***shared_scratch****=path* – path on frontend where ***scratchdir*** can be found.

In the command arguments (paths, executables, ...) following substitutions can be used:

%R – session root – see command *sessiondir*

%C – control dir – see command *control*

%U – username (as specified in configuration, hence empty for '.' control directories)

%u – userid – numerical

%g – groupid – numerical

%H – home dir – home of username as specified in /etc/passwd

%Q – default queue – see command *lrms*

%L – default lrms – ses command *lrms*

%W – installation path – ${ARC_LOCATION}

%c – list of all control directories

%F – path to configuration file of this instance

%I – job ID (for plugins only, substituted in runtime)

%S – job state (for *authplugin* plugins only, substituted in runtime)

%O – reason (for *localcred* plugins only, substituted in runtime). Possible reasons are:

new – new job, new credentials

renew – old job, new credentials

write – write/delete file, create/delete directory

read – read file, directory, etc.

extern – call external program

9.2 Transfer shares

For many jobs, large amounts of input and output data can mean significant time is spent in the PREPARING and FINISHING states gathering input data and writing output data. With FIFO processing, this can lead to one user or group of users blocking the queue for others. The A-REX implements a sharing system to avoid this problem, by assigning each user or group of users to a "transfer share" and specifying a limit on the number of data transfer processes per share. If one user's jobs' transfer share is using the maximum number of processes and another user submits jobs which are assigned to a different share, the second user's jobs can immediately go to PREPARING, up to the same maximum limit of processes. This means that no matter how many jobs the first user submits, the second user's jobs are not blocked. Assuming the bandwidth from the sources of input data for both users' jobs is similar, the available throughput will then be split evenly between the two users' jobs.

If a limit on the total number of data transfer processes is set in the *maxload* option, the maximum number of processes per transfer share is set by splitting the total maximum evenly among all the shares with jobs in data transfer states, up to the maximum allowed per share.

The scheme used to assign jobs to transfer shares can be set in the *maxloadshare* option. Possible values are:

- *dn* - each job is assigned to a share based on the DN of the user sumbitting the job.
- *voms:vo* - if the user's proxy is a VOMS [3] proxy the job is assigned to a share based on the VO specified in the proxy. If the proxy is not a VOMS proxy a default share is used.
- *voms:role* - if the user's proxy is a VOMS proxy the job is assigned to a share based on the role specified in the first attribute found in the proxy. If the proxy is not a VOMS proxy a default share is used.
- *voms:group* - if the user's proxy is a VOMS proxy the job is assigned to a share based on the group specified in the first attribute found in the proxy. If the proxy is not a VOMS proxy a default share is used.

It's possible to distinguish some transfer shares and assign them a limit different from what's specified in *maxloadshare*. It's done by *share_limit* option. *share_limit* can only be used if *maxloadshare* has been already set before. Depending on the sharing mechanism used by *maxloadshare*, the proper name for the share should be specified, as illustrated by the following examples (note, that in dn case spaces are allowed, the configuration parser will take care of them):

- *dn*: /O=Grid/O=NorduGrid/OU=domainname.com/CN=Jane Doe
- *voms:vo*: voname
- *voms:role*: voname:rolename
- *voms:group*: /voname/groupname

The specific shares, specified in *share_limit*, are processed differently from the other shares. A-REX reserves an indicated number of processes for each specific share. The number of unreserved processes is then split evenly between the ordinary shares, as determined by *maxloadshare*. So the specific shares have a strict, non-decreaseable limit, unlike all the ordinary shares, whose limit can be decreased while A-REX tries to split the load evenly. However, A-REX reserves processes only for active specific shares, i.e. shares to which at least one active job on the resource belongs to. If the share is not active, its slots are used in overall splitting between ordinary transfer shares.

A particular case is when A-REX reserves more processes than specified in *maxjobs*. A-REX will process jobs from specific share at FIFO-basis and stop at reaching *maxjobs* number of processes, even if some specific shares haven't reached their limits. Also in this situation each ordinary share is allowed to launch only one upload and download process.

If VOMS is not supported, the *dn* scheme is the only option that should be used, as using a VOMS-based scheme will lead to all jobs being assigned to the default share. The current number of jobs processing and pending processing for each share can be seen with the command *gm-jobs -s*.

Important: If a sharing mechanism based on VOMS is used, server certificates for each supported VO must be installed. It is possible to either download the public key of each VOMS server, or create a special file for each VO containing the server's DN and its CA DN. Instructions are given on NorduGrid's web site at `http://www.nordugrid.org/documents/voms-notes.html`.

When XML file only is used to configure the A-REX, the transfer shares can be implemented by defining *maxLoadShare* (the limit itself) and *loadShareType* (the scheme used) elements inside *loadLimits* block. For defining the specific shares, *shareLimit* sub-blocks with *name* and *limit* elements can be used after *maxLoadShare*.

9.3 Authorization

Authorization is performed by generic means provided by HED framework. Currently A-REX does not implement any internal authorization techniques except those imposed by Access Policy assigned to jobs through AccessControl element of assigned JSDL.

9.4 LRMS support

For information about supported LRMSes and their specific features and configuration options please read dedicated documentation [1].

9.5 Runtime environment

The A-REX can run specially prepared *BASH* scripts prior to creation of the job's script, before and after executing job's main executable. Those scripts are requested by the user through the *runtimeenvironment* attribute in JSDL and are run with the only argument set either equal to '0', '1' or '2' during creation of the job's script, before execution of the main executable and after main the executable is finished, respectively. They all are run through BASH's 'source' command, and hence can manipulate shell variables. With argument '0' scripts are run by the A-REX on the frontend. Some environment variables are defined in that case and can be changed to influence job's execution later:

- joboption_directory – session directory.
- joboption_arg_# – command with arguments to be executed as specified in the JD (**not** bash array).
- joboption_env_# – array of 'NAME=VALUE' environment variables (**not** bash array).
- joboption_runtime_# – array of requested *runtimeenvironment* names (**not** bash array).
- joboption_num – *runtimeenvironment* currently beeing processed (number starting from 0).
- joboption_stdin – name of file to be attached to stdin handle.
- joboption_stdout – same for stdout.
- joboption_stderr – same for stderr.
- joboption_cputime – amount of CPU time requested (minutes).
- joboption_memory – amount of memory requested (megabytes).
- joboption_count – number of processors requested.

- joboption_lrms – LRMS to be used to run job.
- joboption_queue – name of a queue of LRMS to put job into.
- joboption_nodeproperty_# – array of properties of computing nodes (LRMS specific, **not** bash array).
- joboption_jobname – name of the job as given by user.

For example *joboption_arg_#* could be changed to wrap the main executable. Or *joboption_runtime* could be expanded if current one depends on others.

With argument '1' scripts are run just before the main executable is run. They are executed on the computing node. Such a script can prepare environment for some third-party software package. A current directory in that case is the one which would be used for execution of the job. Variable `$HOME` also points to that directory.

With argument '2' scripts are executed after main executable finished. Main purpose is to clean possible changes done by scripts run with '1' (like removing temporary files). Execution of scripts at that stage also happens on computing node and is not reliable. If the job is killed by LRMS they most probably won't be executed.

For publicly available runtime environments please see the RTE repository at `http://gridrer.csc.fi/`.

10 Installation

The A-REX is installed as a component of the ARC middleware and packages for various distributions are available from NorduGrid repositories, or directly from the download area of the NorduGrid website. Source code ready for compilation is available too.

10.1 Requirements

When installed from binary packages, all the dependencies are handled automatically. For compilation from source code please read included README files.

10.2 Setup of the A-REX with WS Interface

The A-REX service is a pluggable module of the HED, so it is first required to set up HED configuration, and add the A-REX elements. HED configuration is in an XML format, however a special comand *a-rex* has been written which automatically creates a HED configuration for running the A-REX from an existing arc.conf file used by the Grid Manager, and so no manual XML configuration editing needs to be done. To enable the WS interface the following line must be added to the arc.conf configuration file:

- ***arex_mount_point***=*path*

where *path* is the A-REX service endpoint, for example `https://your.host:60000/arex`. Then the A-REX can be started

```
$ARC_LOCATION/etc/init.d/a-rex start
```

It is also possible to write configuration directly in XML. To add A-REX to an existing HED configuration, add a new `<Name>` element inside `<Plugins>` containing the string *arex*. This will make HED load the libarex plugin library.

Then add a new `<Service>` element with attribute `name="a-rex"`. That will instantiate the A-REX service. Now to make service accessible extend the `<Plexer>` element with new `<next>` referring to an id of the service. Take care to write the `<Service>` element carefully.

Here is an example of the full HED configuration file including the A-REX service definition:

```
<?xml version="1.0"?>
<ArcConfig
  xmlns="http://www.nordugrid.org/schemas/arcconfig/2009/08"
  xmlns:loader="http://www.nordugrid.org/schemas/loader/2009/08"
  xmlns:arex="http://www.nordugrid.org/schemas/a-rex/2009/08"
  xmlns:tcp="http://www.nordugrid.org/schemas/tcp/2009/08"
  xmlns:tls="http://www.nordugrid.org/schemas/tls/2009/08"
  xmlns:authz="http://www.nordugrid.org/schemas/arcauthz/2009/08"
  xmlns:idmap="http://www.nordugrid.org/schemas/identitymap/2009/10"
>
  <!-- Common configuration of the daemon -->
  <Server>
    <PidFile>/var/run/arched.pid</PidFile>
    <Logger>
      <Level>VERBOSE</Level>
      <File>/var/log/arc/arched.log</File>
      <MaxSize>100000000</MaxSize>
      <Backups>10</Backups>
    </Logger>
  </Server>
  <!-- Where to find plugins -->
  <loader:ModuleManager>
    <loader:Path>/usr/local/lib/arc/</loader:Path>
  </loader:ModuleManager>
  <!-- Simply load all needed plugins -->
  <loader:Plugins>
    <loader:Name>mcctcp</loader:Name>
    <loader:Name>mcctls</loader:Name>
    <loader:Name>mcchttp</loader:Name>
    <loader:Name>mccsoap</loader:Name>
    <loader:Name>arcpdc</loader:Name>
    <loader:Name>identitymap</loader:Name>
    <loader:Name>arex</loader:Name>
  </loader:Plugins>
  <!-- Create a chain -->
  <loader:Chain>
    <!-- TCP listening socket -->
    <loader:Component name="tcp.service" id="tcp">
      <loader:next id="tls"/>
      <tcp:Listen><tcp:Port>60000</tcp:Port></tcp:Listen>
    </loader:Component>
    <!-- Transport-level security -->
    <loader:Component name="tls.service" id="tls">
      <loader:next id="http"/>
      <!-- Location of server's security keys -->
      <tls:KeyPath>/etc/grid-security/hostkey.pem</tls:KeyPath>
      <tls:CertificatePath>/etc/grid-security/hostcert.pem</tls:CertificatePath>
      <tls:CACertificatesDir>/etc/grid-security/certificates</tls:CACertificatesDir>
      <!-- Evaluate requestor's identity into local identity.
      Comment it if no user mapping is needed. -->
      <loader:SecHandler name="identity.map" id="map" event="incoming">
        <!-- Safe choice if all other rules failed -->
        <idmap:PDP name="allow.pdp"><idmap:LocalName>nobody</idmap:LocalName></idmap:PDP>
      </loader:SecHandler>
    </loader:Component>
    <!-- HTTP processing is done here -->
    <loader:Component name="http.service" id="http">
```

```
      <loader:next id="soap">POST</loader:next>
      <loader:next id="plexer">GET</loader:next>
      <loader:next id="plexer">PUT</loader:next>
    </loader:Component>
    <!-- This one parses content into XML tree -->
    <loader:Component name="soap.service" id="soap">
      <loader:next id="plexer"/>
    </loader:Component>
    <!-- Directing messages to proper service -->
    <loader:Plexer name="plexer.service" id="plexer">
      <!-- RegExp pattern is matched to path part of endpoint.
      Unmatched part of path is propagated to service in
      PLEXER:EXTENSION attribute. -->
      <loader:next id="a-rex">^/arex</loader:next>
    </loader:Plexer>
    <!-- A-Rex service -->
    <loader:Service name="a-rex" id="a-rex">
      <!-- Optional endpoint element is advised in case of multiple IP adresses -->
      <arex:endpoint>https://localhost:60000/arex</arex:endpoint>
      <!-- Use information generated by identity.map plugin or default provided below -->
      <arex:usermap><arex:defaultLocalName>nobody</arex:defaultLocalName></arex:usermap>
      <!-- grid-manager part of a-rex requires legacy configuration file.
      Use arc.conf example or write own. -->
      <arex:gmconfig>/etc/arc.conf</arex:gmconfig>
    </loader:Service>
  </loader:Chain>
</ArcConfig>
```

For in-depth information about available elements see Appendix B.

For A-REX configuration, either use a template arc.conf or write a new A-REX configuration file. For information about format and available configuration commands see Section 9.1. It is also possible to specify the A-REX configuration in XML format in the HED configuration file, instead of a separate file. For more information on this see the examples included in the documentation bundled with the release.

For a quick start, simply run

```
$ARC_LOCATION/sbin/arched -c <HED configuration file>
```

For more information please read the *User Guide* [11].

10.3 Setup of the A-REX with GridFTP Interface

The A-REX can process jobs submitted through a traditional GridFTP interface instead of, or as well as, the WS interface. The A-REX version of the ARC GridFTP server (`a-rex-gridftpd`) must be used instead of the Grid Manager server. However, the GridFTP/Grid Manager server packages can be installed alongside A-REX packages, so it is easy to switch between the two. The configuration of the GridFTP server is in the same legacy *arc.conf* configuration file as is used by the A-REX, and configuration instructions are given in [**?**]. In general no configuration changes are necessary to use a Grid Manager-configured GridFTP server with the A-REX.

If it is desired to run both WS and GridFTP interfaces at the same time, simply setup and start the A-REX as described in the previous section, then start the GridFTP server. If only the GridFTP interface is to be used, then the special *a-rex* command should be used (rather than *arched*) to start A-REX. The GridFTP server should then be started.

IMPORTANT: Do not run the A-REX and Grid Manager services at the same time, as they will interfere with each other with unpredictable results.

10.4 Running as non-root

The A-REX is primarily designed to be run by the *root* UNIX account and serve multiple global Grid identities mapped to several UNIX accounts. Nevertheless it is possible to use *non-root* accounts to run that service at the cost of some functionality loss as described below.

There are no drawbacks of running A-REX under a *non-root* account as long as the only UNIX identity used is that of the user who runs the services and all served files and directories are owned by the server's account. Because A-REX has to impersonate a user's local account while communicating with the LRMS, it can serve only the account it is run under (unless it is run under the *root* account, of course).

A Session directory access through HTTP(S) interface

In addition to the BES interface A-REX provides access to the SD through pure HTTP(S) interface. This functionality is used for uploading user-stageable files during job submission and for staging out result files produced by job. It can also be used to monitor job execution by checking content of application dependent files in SD.

The BES defines job identifier as WS Addressing [12] Endpoint Reference (EPR) – XML document. The EPR is extendable and the A-REX adds it own element `JobSessionDir` belonging to the namespace `http://www.nordugrid.org/schemas/a-rex` as a direct child of ReferenceParameters element. This new element contains the URL of SD.

Obtained URL should be extended with file names relative to SD and HTTP methods PUT and GET may be used to upload/download content of those files. For directories – including SD itself – GET method is supported which returns HTML encoded non-recursive list of files and directories. The files and subdirectories have their URLs inside HTML element `<A>`.

B Configuration schema of A-REX

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.nordugrid.org/schemas/ArcConfig/2007/arex"
xmlns:arc="http://www.nordugrid.org/schemas/ArcConfig/2007/arex"
targetNamespace="http://www.nordugrid.org/schemas/ArcConfig/2007/arex"
elementFormDefault="qualified">
  <xsd:complexType name="endpoint_Type">
    <!--
    This element defines URL of A-REX service as seen from outside.
    -->
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:element name="endpoint" type="endpoint_Type"/>
  <xsd:complexType name="gmconfig_Type">
    <!--
    This element defines path to arc0 Grid Manager configuartion file.
    By default it is /etc/arc.conf.
    -->
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:element name="gmconfig" type="gmconfig_Type"/>
```

```
  <xsd:simpleType name="gmrun_Type">
    <!--
    This element defines how grid-manager part of A-Rex is run.
    {*} internal - as a thread inside service container.
    {*} none - no grid-manager is run.
    {*} external - as a separate executable (not supported anymore).
    Default is 'internal'.
    -->
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="internal"/>
      <xsd:enumeration value="external"/>
      <xsd:enumeration value="none"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="gmrun" type="gmrun_Type"/>
  <xsd:complexType name="usermap_Type">
    <xsd:sequence>
      <xsd:element name="defaultLocalName" type="xsd:string"
                   minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="usermap" type="usermap_Type"/>
  <!-- CommonName attribute of bes-factory. -->
  <xsd:element name="commonName" type="xsd:string"/>
  <!-- LongDescription attribute of bes-factory. -->
  <xsd:element name="longDescription" type="xsd:string"/>
  <!-- Name of Local Resource Management System. -->
  <xsd:element name="LRMSName" type="xsd:string"/>
  <!--
  Name of Operating System.
  The values are based on the OSType field of the CIM_OperatingSystem model:
  http://www.dmtf.org/standards/cim/cim_schema_v29
  -->
  <xsd:element name="OperatingSystem" type="xsd:string"/>
</xsd:schema>
```

C A-REX WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://www.nordugrid.org/schemas/a-rex"
 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:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:bes-factory="http://schemas.ggf.org/bes/2006/08/bes-factory"
 xmlns:bes-mgmt="http://schemas.ggf.org/bes/2006/08/bes-management"
 xmlns:deleg="http://www.nordugrid.org/schemas/delegation"
 xmlns:wsrf-rpw="http://docs.oasis-open.org/wsrf/rpw-2"
 xmlns:a-rex="http://www.nordugrid.org/schemas/a-rex">
  <wsdl:import namespase="http://schemas.ggf.org/bes/2006/08/bes-factory"
               location="./bes-factory.wsdl"/>
  <wsdl:import namespase="http://schemas.ggf.org/bes/2006/08/bes-management"
               location="./bes-management.wsdl"/>
  <wsdl:import namespase="http://www.nordugrid.org/schemas/delegation"
               location="../schemas/delegation.wsdl"/>
  <wsdl:import namespase="http://docs.oasis-open.org/wsrf/rpw-2"
               location="http://docs.oasis-open.org/wsrf/rpw-2.wsdl"/>
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.nordugrid.org/schemas/a-rex">
      <xsd:import namespace="http://www.w3.org/2005/08/addressing"
                  schemaLocation="./ws-addr.xsd"/>
      <xsd:simpleType name="ActivitySubStateType">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="Accepting"/>
          <xsd:enumeration value="Accepted"/>
          <xsd:enumeration value="Preparing"/>
          <xsd:enumeration value="Prepared"/>
          <xsd:enumeration value="Submitting"/>
```

```
        <xsd:enumeration value="Executing"/>
        <xsd:enumeration value="Killing"/>
        <xsd:enumeration value="Executed"/>
        <xsd:enumeration value="Finishing"/>
        <xsd:enumeration value="Finished"/>
        <xsd:enumeration value="Failed"/>
        <xsd:enumeration value="Deleted"/>
        <xsd:enumeration value="Pending"/>
        <xsd:enumeration value="Held"/>
      </xsd:restriction>
    </xsd:simpleType>
    <xsd:element name="State" type="a-rex:ActivitySubStateType"/>
    <xsd:complexType name="ResourceInformationDocumentType">
      <xsd:sequence>
         <xsd:element name="BESFactory"
                      type="bes-factory:FactoryResourceAttributesDocumentType"/>
        <xsd:complexType name="Glue2Resource" minOccurs='0'>
          <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax"
                     minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="Activities" minOccurs='0'>
          <xsd:sequence>
            <xsd:complexType name="Activity" minOccurs='0' maxOccurs='unbounded'>
              <xsd:sequence>
                <xsd:element name="ActivityIdentifier"
                             type="wsa:EndpointReferenceType"/>
                <xsd:element ref="bes-factory:ActivityDocument" minOccurs='0'/>
                <xsd:complexType name="Glue2Job" minOccurs='0'>
                  <xsd:sequence>
                    <xsd:any namespace="##other" processContents="lax"
                             minOccurs="0" maxOccurs="unbounded"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ChangeActivityStatusRequestType">
      <xsd:sequence>
        <xsd:element name="ActivityIdentifier" type="wsa:EndpointReferenceType"/>
        <xsd:element name="OldStatus" type="bes-factory:ActivityStatusType"
                     minOccurs="0"/>
        <xsd:element name="NewStatus" type="bes-factory:ActivityStatusType"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="ChangeActivityStatus"
                 type="a-rex:ChangeActivityStatusRequestType"/>
    <xsd:complexType name="ChangeActivityStatusResponseType">
      <xsd:sequence>
        <xsd:element name="NewStatus" type="bes-factory:ActivityStatusType"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="ChangeActivityStatusResponse"
                 type="a-rex:ChangeActivityStatusResponseType"/>
    <xsd:complexType name="MigrateActivityType">
      <xsd:sequence>
<xsd:element name="ActivityIdentifier"
             type="wsa:EndpointReferenceType"
             minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="bes-factory:ActivityDocument" minOccurs="0"/>
<xsd:element name="ForceMigration" type="xsd:boolean" minOccurs="0" maxOccurs="1"/>
        <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
    </xsd:complexType>

    <xsd:complexType name="MigrateActivityResponseType">
      <xsd:sequence>
        <xsd:element name="ActivityIdentifier" type="wsa:EndpointReferenceType"/>
```

```
          <xsd:element ref="bes-factory:ActivityDocument" minOccurs="0"/>
          <xsd:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:anyAttribute namespace="##other" processContents="lax"/>
      </xsd:complexType>

      <xsd:element name="MigrateActivity"
        type="a-rex:MigrateActivityType"/>
      <xsd:element name="MigrateActivityResponse"
        type="a-rex:MigrateActivityResponseType"/>

    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="ChangeActivityStatusRequest">
    <wsdl:part name="ChangeActivityStatusRequest"
               element="a-rex:ChangeActivityStatus"/>
  </wsdl:message>
  <wsdl:message name="ChangeActivityStatusResponse">
    <wsdl:part name="ChangeActivityStatusResponse"
               element="a-rex:ChangeActivityStatusResponse"/>
  </wsdl:message>

  <wsdl:message name="MigrateActivityRequest">
    <wsdl:part name="MigrateActivityRequest"
element="a-rex:MigrateActivity"/>
  </wsdl:message>

  <wsdl:message name="MigrateActivityResponse">
    <wsdl:part name="MigrateActivityResponse"
        element="a-rex:MigrateActivityResponse"/>
  </wsdl:message>

  <wsdl:portType name="a-rex">
    <wsdl:operation name="ChangeActivityStatus">
      <wsdl:documentation>
        This operation allows any simple status change request
        which involves no additional parameters. It should be
        used to modify job/activity execution flow:
          - To put job on hold
          - To rerun job in case of failure
          - To cancel job (same as TerminateActivity of BESFActory)
          - To remove/release job - as long as non-existence is a state
          - Any other status change no supported by BES
      </wsdl:documentation>
      <wsdl:input name="ChangeActivityStatusRequest"
        message="a-rex:ChangeActivityStatusRequest"/>
      <wsdl:output name="ChangeActivityStatusResponse"
        message="a-rex:ChangeActivityStatusResponse"/>
      <wsdl:fault name="NotAuthorizedFault"
        message="bes-factory:NotAuthorizedFault"
        wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
      <wsdl:fault name="InvalidActivityIdentifierFault"
        message="bes-factory:InvalidActivityIdentifierFault"
        wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
      <wsdl:fault name="CantApplyOperationToCurrentStateFault"
         message="bes-factory:CantApplyOperationToCurrentStateFault"
         wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
      <wsdl:fault name="OperationWillBeAppliedEventuallyFault"
         message="bes-factory:OperationWillBeAppliedEventuallyFault"
         wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
    </wsdl:operation>
    <wsdl:operation name="MigrateActivity">
      <wsdl:input
         name="MigrateActivity"
         message="a-rex:MigrateActivityRequest"
      <wsdl:output
         name="MigrateActivityResponse"
         message="a-rex:MigrateActivityResponse"
      <wsdl:fault name="NotAuthorizedFault"
         message="bes-factory:NotAuthorizedFault"
        wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
      <wsdl:fault name="NotAcceptingNewActivitiesFault"
         message="bes-factory:NotAcceptingNewActivitiesFault"
```

```
        wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
      <wsdl:fault name="UnsupportedFeatureFault"
         message="bes-factory:UnsupportedFeatureFault"
         wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
      <wsdl:fault name="InvalidRequestMessageFault"
         message="bes-factory:InvalidRequestMessageFault"
         wsa:Action="http://schemas.ggf.org/bes/2006/08/bes-factory/BESFactoryPortType/Fault"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="a-rex" type="a-rex:a-rex">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="ChangeActivityStatus">
      <soap:operation soapAction="ChangeActivityStatus"/>
      <wsdl:input name="ChangeActivityStatusRequest">
         <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="ChangeActivityStatusResponse">
         <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="NotAuthorizedFault">
        <soap:fault name="NotAuthorizedFault" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="InvalidActivityIdentifierFault">
        <soap:fault name="InvalidActivityIdentifierFault" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="CantApplyOperationToCurrentStateFault">
        <soap:fault name="CantApplyOperationToCurrentStateFault" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="OperationWillBeAppliedEventuallyFault">
        <soap:fault name="OperationWillBeAppliedEventuallyFault" use="literal" />
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="MigrateActivity">
      <soap:operation soapAction="MigrateActivity" />
      <wsdl:input name="MigrateActivity">
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output name="MigrateActivityResponse">
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="NotAuthorizedFault">
        <soap:fault name="NotAuthorizedFault" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="NotAcceptingNewActivitiesFault">
        <soap:fault name="NotAcceptingNewActivitiesFault" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="UnsupportedFeatureFault">
        <soap:fault name="UnsupportedFeatureFault" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="InvalidRequestMessageFault">
        <soap:fault name="InvalidRequestMessageFault" use="literal" />
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="GetResourcePropertyDocument"
               type="wsrf-rpw:GetResourcePropertyDocument">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="GetResourcePropertyDocument">
      <soap:operation soapAction="GetResourcePropertyDocument"/>
      <wsdl:input name="wsrf-rpw:GetResourcePropertyDocumentRequest">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="wsrf-rpw:GetResourcePropertyDocumentResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ResourceUnknownFault">
        <soap:fault name="ResourceUnknownFault" use="literal" />
      </wsdl:fault>
      <wsdl:fault name="ResourceUnavailableFault">
        <soap:fault name="ResourceUnavailabbleFault" use="literal" />
      </wsdl:fault>
```

```
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="GetResourceProperty" type="wsrf-rpw:GetResourceProperty">
  <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="GetResourceProperty">
    <soap:operation soapAction="GetResourceProperty"/>
    <wsdl:input name="wsrf-rpw:GetResourcePropertyRequest">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="wsrf-rpw:GetResourcePropertyResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="ResourceUnknownFault">
      <soap:fault name="ResourceUnknownFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="ResourceUnavailableFault">
      <soap:fault name="ResourceUnavailabbleFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="InvalidResourcePropertyQNameFault">
      <soap:fault name="InvalidResourcePropertyQNameFault" use="literal" />
    </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="QueryResourceProperties" type="wsrf:QueryResourceProperties">
  <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="QueryResourceProperties">
    <soap:operation soapAction="QueryResourceProperties"/>
    <wsdl:input name="wsrf-rpw:QueryResourcePropertiesRequest">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="wsrf-rpw:QueryResourcePropertiesResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="ResourceUnknownFault">
      <soap:fault name="ResourceUnknownFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="ResourceUnavailableFault">
      <soap:fault name="ResourceUnavailabbleFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="InvalidResourcePropertyQNameFault">
      <soap:fault name="InvalidResourcePropertyQNameFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="UnknownQueryExpressionDialectFault">
      <soap:fault name="UnknownQueryExpressionDialectFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="InvalidQueryExpressionFault">
      <soap:fault name="InvalidQueryExpressionFault" use="literal" />
    </wsdl:fault>
    <wsdl:fault name="QueryEvaluationErrorFault">
      <soap:fault name="QueryEvaluationErrorFault" use="literal" />
    </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="a-rex">
  <wsdl:port name="delegation" binding="deleg:DelegationBinding">
  </wsdl:port>
  <wsdl:port name="bes-factory" binding="bes-factory:BESFactoryBinding">
  </wsdl:port>
  <wsdl:port name="bes-mgmt" binding="bes-mgmt:BESManagementBinding">
  </wsdl:port>
  <wsdl:port name="GetResourcePropertyDocument"
             binding="a-rex:GetResourcePropertyDocument">
  </wsdl:port>
  <wsdl:port name="GetResourceProperty" binding="a-rex:GetResourceProperty">
  </wsdl:port>
  <wsdl:port name="QueryResourceProperties"
             binding="a-rex:QueryResourceProperties">
  </wsdl:port>
  <wsdl:port name="a-rex" binding="a-rex:a-rex">
  </wsdl:port>
</wsdl:service>
```

```
</wsdl:definitions>
```

D Delegation WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://www.nordugrid.org/schemas/delegation"
 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:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:deleg="http://www.nordugrid.org/schemas/delegation">
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.nordugrid.org/schemas/delegation">
      <!-- Common types -->
      <xsd:simpleType name="TokenFormatType">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="x509"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="ReferenceType">
        <xsd:sequence>
          <xsd:any namespace="##other" processContents="lax"
                   minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="DelegatedTokenType">
        <xsd:sequence>
          <xsd:element name="Id" type="xsd:string"/>
          <xsd:element name="Value" type="xsd:string"/>
          <xsd:element name="Reference" type="deleg:ReferenceType"
                       minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="Format" type="deleg:TokenFormatType"
                       use="required"/>
      </xsd:complexType>
      <xsd:element name="DelegatedToken" type="deleg:DelegatedTokenType"/>
      <xsd:complexType name="TokenRequestType">
        <xsd:sequence>
          <xsd:element name="Id" type="xsd:string"/>
          <xsd:element name="Value" type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="Format" type="deleg:TokenFormatType"
                       use="required"/>
      </xsd:complexType>
      <xsd:element name="TokenRequest" type="deleg:TokenRequestType"/>
      <!-- Types for messages -->
      <xsd:complexType name="DelegateCredentialsInitRequestType">
        <xsd:sequence>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DelegateCredentialsInit"
                   type="deleg:DelegateCredentialsInitRequestType"/>
      <xsd:complexType name="DelegateCredentialsInitResponseType">
        <xsd:sequence>
          <xsd:element name="TokenRequest" type="deleg:TokenRequestType"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="DelegateCredentialsInitResponse"
                   type="deleg:DelegateCredentialsInitResponseType"/>
      <xsd:complexType name="UpdateCredentialsRequestType">
        <xsd:sequence>
          <xsd:element name="DelegatedToken" type="deleg:DelegatedTokenType"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="UpdateCredentials"
                   type="deleg:UpdateCredentialsRequestType"/>
      <xsd:complexType name="UpdateCredentialsResponseType">
        <xsd:sequence>
```

```
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="UpdateCredentialsResponse"
                 type="deleg:UpdateCredentialsResponseType"/>
    <!-- Faults -->
    <xsd:complexType name="UnsupportedFaultType">
      <xsd:sequence>
        <xsd:element name="Description" type="xsd:string"
                     minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="UnsupportedFault" type="deleg:UnsupportedFaultType"/>
    <xsd:complexType name="ProcessingFaultType">
      <xsd:sequence>
        <xsd:element name="Description" type="xsd:string"
                     minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="ProcessingFault" type="deleg:ProcessingFaultType"/>
    <xsd:complexType name="WrongReferenceFaultType">
      <xsd:sequence>
        <xsd:element name="Description" type="xsd:string"
                     minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="WrongReferenceFault" type="deleg:WrongReferenceFaultType"/>
  </xsd:schema>
</wsdl:types>
<wsdl:message name="DelegateCredentialsInitRequest">
  <wsdl:part name="DelegateCredentialsInitRequest"
             element="deleg:DelegateCredentialsInit"/>
</wsdl:message>
<wsdl:message name="DelegateCredentialsInitResponse">
  <wsdl:part name="DelegateCredentialsInitResponse"
             element="deleg:DelegateCredentialsInitResponse"/>
</wsdl:message>
<wsdl:message name="UpdateCredentialsRequest">
  <wsdl:part name="UpdateCredentialsRequest" element="deleg:UpdateCredentials"/>
</wsdl:message>
<wsdl:message name="UpdateCredentialsResponse">
  <wsdl:part name="UpdateCredentialsResponse"
             element="deleg:UpdateCredentialsResponse"/>
</wsdl:message>
<wsdl:message name="UnsupportedFault">
  <wsdl:part name="Detail" element="deleg:UnsupportedFault"/>
</wsdl:message>
<wsdl:message name="ProcessingFault">
  <wsdl:part name="Detail" element="deleg:ProcessingFault"/>
</wsdl:message>
<wsdl:message name="WrongReferenceFault">
  <wsdl:part name="Detail" element="deleg:WrongReferenceFault"/>
</wsdl:message>
<wsdl:portType name="DelegationPortType">
 <wsdl:operation name="DelegateCredentialsInit">
    <wsdl:documentation>
    </wsdl:documentation>
    <wsdl:input name="DelegateCredentialsInitRequest"
      message="deleg:DelegateCredentialsInitRequest"/>
    <wsdl:output name="DelegateCredentialsInitResponse"
      message="deleg:DelegateCredentialsInitResponse"/>
    <wsdl:fault name="UnsupportedFault"
      message="deleg:UnsupportedFault"/>
    <wsdl:fault name="ProcessingFault"
      message="deleg:ProcessingFault"/>
  </wsdl:operation>
 <wsdl:operation name="UpdateCredentials">
    <wsdl:documentation>
    </wsdl:documentation>
    <wsdl:input name="UpdateCredentialsRequest"
      message="deleg:UpdateCredentialsRequest"/>
    <wsdl:output name="UpdateCredentialsResponse"
      message="deleg:UpdateCredentialsResponse"/>
    <wsdl:fault name="UnsupportedFault"
```

```
        message="deleg:UnsupportedFault"/>
      <wsdl:fault name="ProcessingFault"
        message="deleg:ProcessingFault"/>
      <wsdl:fault name="WrongReferenceFault"
        message="deleg:WrongReferenceFault"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="DelegationBinding" type="deleg:DelegationPortType">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="DelegateCredentialsInit">
      <soap:operation soapAction="DelegateCredentialsInit"/>
      <wsdl:input name="DelegateCredentialsInitRequest">
         <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="DelegateCredentialsInitResponse">
         <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="UpdateCredentials">
      <soap:operation soapAction="UpdateCredentials"/>
      <wsdl:input name="UpdateCredentialsRequest">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="UpdateCredentialsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
</wsdl:definitions>
```

E ARC extensions for JSDL schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.nordugrid.org/ws/schemas/jsdl-arc"
            xmlns:jsdl-arc="http://www.nordugrid.org/ws/schemas/jsdl-arc"
            targetNamespace="http://www.nordugrid.org/ws/schemas/jsdl-arc">
 <xsd:simpleType name="GMState_Type">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="ACCEPTED"/>
   <xsd:enumeration value="PREPARING"/>
   <xsd:enumeration value="SUBMIT"/>
   <xsd:enumeration value="INLRMS"/>
   <xsd:enumeration value="FINISHING"/>
   <xsd:enumeration value="FINISHED"/>
   <xsd:enumeration value="DELETED"/>
   <xsd:enumeration value="CANCELING"/>
  </xsd:restriction>
 </xsd:simpleType>
 <xsd:complexType name="Version_Type">
  <xsd:sequence>
   <xsd:element name="UpperExclusive" type="xsd:string"
                minOccurs="0"/>
   <xsd:element name="LowerExclusive" type="xsd:string"
                minOccurs="0"/>
   <xsd:element name="Exact" type="xsd:string" minOccurs="0"
                maxOccurs="unbounded"/>
   <xsd:element name="Exclusive" type="xsd:boolean"
                minOccurs="0"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:simpleType name="SessionType_Type">
  <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
  <!-- xsd:element ref="SessionType" minOccurs="0"/ -->
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="INTERNAL"/>
   <xsd:enumeration value="LIMITED"/>
   <xsd:enumeration value="READONLY"/>
   <xsd:enumeration value="FULL"/>
  </xsd:restriction>
```

```
</xsd:simpleType>
<xsd:simpleType name="IsExecutable_Type">
 <xsd:documentation> For jsdl:DataStaging_Type (default: false) </xsd:documentation>
 <!-- xsd:element ref="IsExecutable" minOccurs="0"/ -->
 <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>
<xsd:simpleType name="FileParameters_Type">
 <xsd:documentation> For jsdl:DataStaging_Type </xsd:documentation>
 <!-- xsd:element ref="IsExecutable" minOccurs="0"/ -->
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:simpleType name="JoinOutputs_Type">
 <xsd:documentation> For jsdl:JobDescription_Type (default: false) </xsd:documentation>
 <!-- xsd:element ref="JoinOutputs" minOccurs="0"/ -->
 <xsd:restriction base="xsd:boolean"/>
</xsd:simpleType>
<xsd:simpleType name="Reruns_Type">
 <xsd:documentation> For jsdl:JobDescription_Type (default: false) </xsd:documentation>
 <!-- xsd:element ref="Reruns" minOccurs="0"// -->
 <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>
<xsd:complexType name="RunTimeEnvironment_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="RunTimeEnvironment" minOccurs="0"
                  maxOccurs="unbounded"/ -->
 <xsd:sequence>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="Version" type="Version_Type"
               minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="Middleware_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="Middleware" minOccurs="0"
                  maxOccurs="unbounded"/ -->
 <xsd:sequence>
  <xsd:element name="Name" type="xsd:string"/>
  <xsd:element name="Version" type="Version_Type"
               minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="RemoteLogging_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="RemoteLogging" minOccurs="0"
                  maxOccurs="3"/ -->
 <xsd:sequence>
  <xsd:element name="URL" minOccurs="1" maxOccurs="1"
               type="xsd:anyURI"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="CredentialServer_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="CredentialServer" minOccurs="0"/ -->
 <xsd:sequence>
  <xsd:element name="URL" minOccurs="1" maxOccurs="1"
               type="xsd:anyURI"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="LocalLogging_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="LocalLogging" minOccurs="0"
                  maxOccurs="1"/ -->
 <xsd:sequence>
  <xsd:element name="Directory" minOccurs="1" maxOccurs="1"
               type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="AccessControlType_Type">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="GACL"/>
 </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="AccessControl_Type">
```

```
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="AccessControl" minOccurs="0"/ -->
 <xsd:sequence>
  <xsd:element name="OwnerAlwaysAllowed" type="xsd:boolean"
               minOccurs="0"/>
  <xsd:element name="Type" type="AccessControlType_Type"
               minOccurs="0"/>
  <xsd:element name="Content" minOccurs="0" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="NotificationType_Type">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="Email"/>
 </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="Notify_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="Notify" minOccurs="0" maxOccurs="3"/
   -->
 <xsd:sequence>
  <xsd:element name="Type" type="NotificationType_Type"
               minOccurs="0"/>
  <xsd:element name="Endpoint" minOccurs="0" type="xsd:string"/>
  <xsd:element name="State" minOccurs="1" maxOccurs="unbounded"
               type="GMState_Type"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="SessionLifeTime_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="SessionLifeTime" minOccurs="0"
                  maxOccurs="1"/ -->
 <xsd:restriction base="xsd:long"/>
</xsd:simpleType>
<xsd:simpleType name="GridTimeLimit_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="GridTimeLimit" minOccurs="0"
                  maxOccurs="1"/ -->
 <xsd:restriction base="xsd:positiveInteger"/>
</xsd:simpleType>
<xsd:complexType name="CandidateTarget_Type">
 <xsd:documentation> For jsdl:Resources_Type </xsd:documentation>
 <!-- xsd:element ref="jsdl-arc:CandidateTarget" minOccurs="0"
                  maxOccurs="1"/ -->
 <xsd:sequence>
  <xsd:element name="HostName" minOccurs="0" type="xsd:string"/>
  <xsd:element name="QueueName" minOccurs="0" type="xsd:string"/>
 </xsd:sequence>
</xsd:complexType>
<xsd:simpleType name="Time_Type">
 <xsd:documentation> For jsdl:JobDescription_Type </xsd:documentation>
 <!-- xsd:element ref="ProcessingStartTime" minOccurs="0"
                  maxOccurs="1"/ -->
 <xsd:restriction base="xsd:dateTime"/>
</xsd:simpleType>
<!--============================================================-->
<xsd:element name="IsExecutable" type="IsExecutable_Type"/>
<xsd:element name="FileParameters" type="FileParameters_Type"/>
<xsd:element name="RunTimeEnvironment" type="RunTimeEnvironment_Type"/>
<xsd:element name="Middleware" type="Middleware_Type"/>
<xsd:element name="RemoteLogging" type="RemoteLogging_Type"/>
<xsd:element name="LocalLogging" type="LocalLogging_Type"/>
<xsd:element name="AccessControl" type="AccessControl_Type"/>
<xsd:element name="Notify" type="Notify_Type"/>
<xsd:element name="SessionLifeTime" type="SessionLifeTime_Type"/>
<xsd:element name="SessionType" type="SessionType_Type"/>
<xsd:element name="JoinOutputs" type="JoinOutputs_Type"/>
<xsd:element name="Reruns" type="Reruns_Type"/>
<xsd:element name="CredentialServer" type="CredentialServer_Type"/>
<xsd:element name="GridTimeLimit" type="GridTimeLimit_Type"/>
<xsd:element name="CandidateTarget" type="CandidateTarget_Type"/>
<xsd:element name="ProcessingStartTime" type="Time_Type"/>
</xsd:schema>
```

F Error messages of A-REX (outdated)

If job has not finished successfully the A-REX puts one or more lines into *job.ID.failed*. Possible values include those generated by the A-REX itself:

Error string	*Reason/description*
Internal error	Error in internal algorithm
Internal error: can't read local file	Error manipulating files in the control directory
Failed reading local job information	-//-
Failed reading status of the job	-//-
Failed writing job status	-//-
Failed during processing failure	-//-
Serious troubles (problems during processing problems)	-//-
Failed initiating job submission to LRMS	Could not run backend executable to pass job to LRMS
Job submission to LRMS failed	Backend executable supposed to pass job to LRMS returned non-zero exit code
Failed extracting LRMS ID due to some internal error	Output of Backend executable supposed to contain local ID of passed job could not be parsed
Failed in files upload (post-processing)	Failed to upload some or all output files
Failed in files upload due to expired credentials – try to renew	Failed to upload some or all output files most probably due to expired credentials (proxy certificate)
Failed to run uploader (post-processing)	Could not run *uploader* executable
uploader failed (postprocessing)	Generic error related to *uploader* component
Failed in files download (pre-processing)	Failed to upload some or all input files
Failed in files download due to expired credentials – try to renew	Failed to download some or all input files most probably due to expired credentials (proxy certificate)
Failed to run downloader (pre-processing)	Could not run *downloader* executable
downloader failed (preprocessing)	Generic error related to *downloader* component
User requested to cancel the job	A-REX detected external request to cancel this job, most probably issued by user
Could not process RSL	Job description could not be processed due to syntax errors or missing elements
User requested dryrun. Job skiped.	Job description contains request not to process this job
LRMS error: (CODE) DESCRIPTION	LRMS returned error. CODE is replaced with numeric code of LRMS, and DESCRIPTION with textual description
Plugin at state STATE failed: OUTPUT	External plugin specified in A-REX configuration returned non-zero exit code. STATE is replcaced by name of state to which job was going to be passed, OUTPUT by textual output generated by plugin.

Failed running plugin at state STATE	External plugin specified in A-REX configuration could not be executed.

Provided by downloader component (URL is replcaced by source of input file, FILE by name of file):

Error string	*Reason/description*
Internal error in downloader	Generic error
Input file: URL – unknown error	Generic error
Input file: URL – unexpected error	Generic error
Input file: URL – bad source URL	Source URL is either malformed or not supported
Input file: URL – bad destination URL	Shouldn't happen
Input file: URL – failed to resolve source locations	File either not registred or other problems related to Data Indexing service.
Input file: URL – failed to resolve destination locations	Shouldn't happen
Input file: URL – failed to register new destination file	Shouldn't happen
Input file: URL – can't start reading from source	Problems related to accessing instance of file at Data Storing service.
Input file: URL – can't read from source	-//-
Input file: URL – can't start writing to destination	Access problems in a session directory
Input file: URL – can't write to destination	-//-
Input file: URL – data transfer was too slow	Timeouted while trying to download file
Input file: URL – failed while closing connection to source	Shouldn't happen
Input file: URL – failed while closing connection to destination	Shouldn't happen
Input file: URL – failed to register new location	Shouldn't happen
Input file: URL – can't use local cache	Problems with A-REX cache
Input file: URL – system error	Operating System returned error code where unexpected
Input file: URL – delegated credentials expired	Access to source requires credententials and they are either outdated or missing (not delegated).
User file: FILENAME – Bad information about file: checksum can't be parsed.	In job description there is a checksum provided for file uploadable by user interface and this record can't be interpreted.
User file: FILENAME – Bad information about file: size can't be parsed.	In job description there is a size provided for file uploadable by user interface and this record can't be interpreted.

User file: FILENAME – Expected file. Directory found.	Instead of file uploadable by user interface A-REX found directory with same name in a session directory.
User file: FILENAME – Expected ordinary file. Special object found.	Instead of file uploadable by user interface A-REX found special object with same name in a session directory.
User file: FILENAME – Delivered file is bigger than specified.	The size of file uploadable by user interface is bigger
User file: FILENAME – Delivered file is unreadable.	A-REX can't check user uploadable file due to some internal error. Most probably due to improperly configured local permissions.
User file: FILENAME – Could not read file to compute checksum.	A-REX can't read user uploadable file due to some internal error. Most probably due to improperly configured local permissions.
User file: FILENAME – Timeout waiting	A-REX waited for user uploadable file too long.

Provided by uploader component (URL is replaced by destination of output file) :

Error string	*Reason/description*
Internal error in uploader	Generic error
Output file: URL – unknown error	Generic error
Output file: URL – unexpected error	Generic error
User requested to store output locally URL	Destination is URL of type *file*.
Output file: URL – bad source URL	Shouldn't happen
Output file: URL – bad destination URL	Destination URL is either malformed or not supported
Output file: URL – failed to resolve source locations	Shouldn't happen
Output file: URL – failed to resolve destination locations	Problems related to Data Indexing service.
Output file: URL – failed to register new destination file	-//-
Output file: URL – can't start reading from source	User request to store output file, but there is no such file or there are problems accessing session directory
Output file: URL – can't start writing to destination	Problems with Data Storing services
Output file: URL – can't read from source	Problems accessing session directory
Output file: URL – can't write to destination	Problems with Data Storing services
Output file: URL – data transfer was too slow	Timeout during transfer
Output file: URL – failed while closing connection to source	Shouldn't happen
Output file: URL – failed while closing connection to destination	Shouldn't happen

Output file: URL – failed to register new location	Problems related to Data Indexing service.
Output file: URL – can't use local cache	Shouldn't happen
Output file: URL – system error	Operating System returned error code where unexpected
Output file: URL – delegated credentials expired	Access to destination requires credententials and they are either outdated or missing (not delegated).

References

[1] Adrian Taga, Thomas Frgt. *ARC batch system back-end interface guide with support for GLUE 2*. URL `http://svn.nordugrid.org/trac/nordugrid/export/10104/arc1/trunk/doc/backend_interface/Backends-arc1.pdf`. NORDUGRID-TECH-18.

[2] A. Konstantinov. *The NorduGrid Grid Manager And GridFTP Server: Description And Administrator's Manual*. The NorduGrid Collaboration. URL `http://www.nordugrid.org/documents/GM.pdf`. NORDUGRID-TECH-2.

[3] R. Alfieri et al. From gridmap-file to VOMS: managing authorization in a Grid environment. *Future Gener. Comput. Syst.*, 21(4):549–558, 2005. ISSN 0167-739X.

[4] A. Anjomshoaa et al. Job Submission Description Language (JSDL) Specification, Version 1.0 (first errata update). GFD-R.136, July 2008. URL `http://www.gridforum.org/documents/GFD.136.pdf`.

[5] B. Kónya. *The NorduGrid/ARC Information System*. The NorduGrid Collaboration. URL `http://www.nordugrid.org/documents/arc_infosys.pdf`. NORDUGRID-TECH-4.

[6] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997. Available at: http://www.globus.org.

[7] I. Foster et al. OGSA™ Basic Execution Service Version 1.0. GFD-R-P.108, August 2007. URL `http://www.ogf.org/documents/GFD.108.pdf`.

[8] M. Humphrey et al. JSDL HPC Profile Application Extension, Version 1.0. GFD-R.111, August 2007. URL `http://www.gridforum.org/documents/GFD.111.pdf`.

[9] OASIS. OASIS Web Services ResourceProperties specification. , April 2006. URL `http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-os.pdf`.

[10] *The Hosting Environment of the Advanced Resource Connector middleware*. The NorduGrid Collaboration, . NORDUGRID-TECH-19.

[11] *The NorduGrid/ARC1 User Guide - not released yet*. The NorduGrid Collaboration, . NORDUGRID-MANUAL-??

[12] W3C. Web Services Addressing 1.0 - SOAP Binding. , May 2006. URL `http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/`.