The GriPhyN Virtual Data System Properties

Property Documentation

V 1.4.6 automatically generated at

2006-07-28 11:58

Contents

1 Introduction

The wfrc file contains Java-style property settings which control the Euryale planner at run-time. This file will report the majority of the settable properties, and their respective default values. Please refer to the user guide for a more in-depth discussion of the configuration options. Please note that the values rely on proper capitalization, unless explicitely noted otherwise.

Some properties can be made to rely on other properties. However, such a construction requires the dependent properties to be declared and visible. As a notation, the curly braces refer to the value of the named property. For instance, ${user.home} means that the value depends on the value of the user.home property plus any noted additions. You can use this notation to refer to other properties, though the extend of the subsitutions are limited. Usually, you want to refer to a set of the standard system properties. Nesting is not allowed. Substitutions will only be done once. The following *system* properties are available:

property	**meaning**
file.separator	string that separates filename components.
java.home	$JAVA_HOME if it exists.
java.class.path	$CLASSPATH if it exists.
java.io.tmpdir	a location of a temporary directory.
os.name	name of operating system, e.g. Linux
os.version	kernel version, e.g. 2.4.31
os.arch	operating system architecture, e.g. i686
user.dir	working directory at point of invocation.
user.home	user's home directory
user.language	$LANG or set to *en*
user.name	user's account name
user.timezone	value of $TZ
vds.home	value of $VDS_HOME

The wfrc properties only affect the Euryale planner, and certain portions of the GVDS planning subsystem. All properties start with the letter "w", usually either "wf" or "work". The Pegasus planner is **not** affected by properties declared in the wfrc file.

The following example provides a sensible set of properties to be set by the user property file. These properties use mostly non-default settings. It is an example only, and will not work for you:

```
wf.profile.vo.name  ivdgl
wf.profile.vo.group ivdgl1
wf.script.pre       ${vds.home}/libexec/prescript.pl
wf.script.post      ${vds.home}/libexec/postscript.pl
wf.site.selector    ${vds.home}/contrib/Euryale/mock-const
wf.pool.style       xml
wf.pool.file        ${vds.home}/var/sites.xml
wf.tc.style         new
wf.tc.file          ${vds.home}/var/tc.data
wf.rc.style         LRC
wf.rc.lrc           rls://evitable.uchicago.edu
wf.base.dir          ${user.home}/work
wf.tailstatd.show    true
work.db              Pg
work.db.database     ${user.name}
work.db.user         ${user.name}
work.db.password     ${user.name}
```

2 Profiles

Profiles are a uniform abstraction of specific run-time environment options. Profiles are propagated in four priority levels through the GVDS, please refer to the user guide.

Any property with the prefix "wf.profile" will set a GVDS profile. The next component in the property key is the profile namespace. The fourth and final component in the property key is the profile key.

namespace	**planner**	**usage**
condor	all	adding to a Condor submit file.
dagman	Pegasus	sending configuration options to DAGMan
env	all	environment variable settings
hints	Pegasus	deprecated.
globus	all	Add Globus RSL instructions
selector	Pegasus	Pass information to the site selector

2.1 wf.profile.vds.voname

Systems	workflow managers, Euryale
Type	arbitrary identifier
Default	ivdgl
See Also	wf.profile.vds.vogroup, section 2.3 (page 7)

The Virtual Organization (VO) name is the name of the group a scientist belongs to, and under which auspices grid time is accounted. It is strongly suggested to set the VO name in order to augment GVDS provenance and book-keeping information.

Plans: In future releases, merges with vds.profile.

2.2 wf.vo.group

Systems	workflow managers
Type	arbitrary identifier
Default	ivdgl1
New name	wf.profile.vds.vogroup, section 2.3 (page 7)

2.3 wf.profile.vds.vogroup

Systems	workflow managers, Euryale
Type	arbitrary identifier
Default	ivdgl1

The VO group is a sub-partition of a VO. Some VOs are partitioned into groups. If you are not aware of any VO group, please use your VO name appended with the digit 1. The information is required for to plan workflows in the Euryale workflow environment. It becomes part of the path where workflows are stored.

Plans: In future releases, merges with vds.profile.

3 Scripts

The Euryale logic resides mostly in the pre- and post script that are automatically run by DAGMan for each job. Only the computational jobs are visible in the Condor submit files. All auxilliary action, like staging files, setup and clean, are done from the parametrized scripts.

3.1 wf.script.pre

Systems	Euryale compile-time
Type	file location path
Default	${vds.home}/libexec/prescript.pl

The pre script is a heavy lifter, handling replanning, selecting a site through the site selector call-out, rewriting the submit file according to the site, and staging in files, and preparing almost every action of the post script.

3.2 wf.script.post

System	Euryale compile-time
Type	file location path
Default	${vds.home}/libexec/prescript.pl

The post script is run in case of a successful job run. It stages files, cleans up, and registered files.

3.3 wf.exitcode.file

System	Euryale run-time
Type	file location path
Default	none

This property specifies the location of VDS "exitcode" program. You will need to have set up your database correctly in $HOME/.vdsrc or $VDS_HOME/etc/properties! If not defined, uses internal parser to just obtain the exitcode. Howver, in this case, you will miss all provenance tracking. It is strongly suggested to use the GVDS exitcode program.

4 Transfers

4.1 wf.final.output

System	Euryale run-time
Type	site handle or gridftp URL
Default	none
See also	wf.ftp.uberftp, section 4.5 (page 9)

This property names the destination for stagable output files (marked as such in VDL). The value is either a site identifier, which will be looked up in the site catalog (SC) to determine its gridftp server(s), or a gridftp URI.

Use of this option requires the presence of the uberftp program, which is shipped with the VDT. If uberftp is not found, transfer to the final resting place are likely to fail.

4.2 wf.transfer.program

System	Euryale run-time
Type	boolean
Value[0]	false – means transfer
Value[1]	true – means T2
Default	false
See also	wf.transfer.arguments, section 4.3 (page 8)

This option determines which transfer tool is being used. Eventually it will take a file location, but for now, there are only two options. If false, the default, the "transfer" program is used to transfer files. If true, the "T2" program is used to transfer files. Each has advantages and disadvantages.

4.3 wf.transfer.arguments

System	Euryale run-time
Type	string
Default	none
See also	wf.transfer.program, section 4.2 (page 8)

If set, this option provides additional arguments to the transfer program. You cannot have anything that requires shell interpretation in these argument string. Defaults to -q arguments (less verbose).

4.4 wf.check.gridftp

System	Euryale run-time
Type	boolean
Value[0]	false
Value[1]	true
Default	true

By default, the remote gridftp server is checked for reachability each time a prescript or postscript runs. This check creates additional load, but the transfer and T2 scripts have retry mechanisms to deal with hick-ups better than the built-in check of Euryale. However, the traditional style is to have Euryale check.

4.5 wf.ftp.uberftp

System	Euryale run-time
Type	Executable location
Default	none

If an uberftp executable is found, which ships with VDT, it is used to re-create the deep directory levels on the final resting place. This property points to the location of an uberftp installation. Additionally, the runtime system will search the PATH for an uberftp installation. If not found, the filenames at the final resting place will be flattened.

5 Worker Node Behavior

5.1 wf.permit.hardlinks

System	Euryale run-time
Type	boolean
Value[0]	false
Value[1]	true
Default	false

This is an optimization option. When creating links for existing remote data files in the job directory at the remote site, permit the use of hardlinks, if it can be safely detected that files reside on the same filesystem, e.g. for files in the workdir. Warning: You cannot use hard links on AFS.

5.2 wf.use.relative

System	Euryale run-time
Type	boolean
Value[0]	false
Value[1]	true
Default	true

If this option evaluates to true, Euryale will translate LFNs for the commandline and in the profiles section using the basename of the file, relative to the working directory defined in the site catalog. If the option is false, Euryale will always use absolute pathnames. Warning: There is a strict 4k limit on the Condor commandline, which is hit quicker with absolute paths.

5.3 wf.remote.job.queues

System	Euryale run-time
Type	site-handle equals string list
Example	jazz=SC03
Default	none

Some sites require that jobs are submitted into certain queues on the site. This option provides the queue information. The property value is a comma-separated list of site handle equals queue handle.

5.4 wf.gridstart.stat

System	Euryale run-time
Type	boolean
Value[0]	false
Value[1]	true
Default	false

This option configures whether to use kickstart's stat options -s and -S. However, due to a bug in kickstarts < 1.21, it is not advisable to use this feature. It is experimental for now.

If configured to true, for each input file a stat call will be attempted before any jobs are started, and for each output file a stat call after all jobs ran. The results are shipped in the standard kickstart records, and thus provide additional information, if data products were scrambled or if the job misbehaved in any other way.

6 Site Selection And Helper

6.1 wf.site.selector

System	Euryale run-time
Type	file location path
Default	none, mandatory

The site selector is central to the approach of Euryale's just in time planning. It is an external application which is called with a single temporary file that contains information about the job to be run. The site selector does whatever is necessary to determine a good site to run a job at. It returns the solution on its stdout filehandle. Please refer to the user guide (?) for in-depth information how to write your own site selector. A number of site selectors are provided:

mock-const	a simple 2-liner shell script. Always returns the same hard-coded site. This is good for testing.
mock-rnd	a simple linear random site selector. It uses the list of site candidates provided, and randomly picks a candidate.
mock-wlr	a weighted random site selectors with affinity for large pools. Uses an auxilliar file mock-wlr.txt in the same dir.
mock-wlr2	a weighted random site selectors with affinity for responsive pools. Uses an auxilliary file mock-wlr2.txt in the same directory.
select_site.py	Kavitha's own selector, uses Catalin's monitoring. However, it uses the deprecated IPC interface.
mock-upcomsel	Catalin's site selector, uses his policy manager.
sitesel-lfu	An example site selector. This can be used to model your own site-selector.

6.2 wf.kss.version

System	Euryale run-time
Type	Enumeration
Value[0]	old
Value[1]	new
Default	new

This property is deprecated, and will soon become void! Do not use any longer! The *new* interface is already the standard. The old API is no longer available.

6.3 wf.policy.manager

System	Euryale run-time
Type	boolean
Default	true
New name	wf.initially.held, section 6.4 (page 12)

This option name is deprecated.

6.4 wf.initially.held

System	Euryale run-time
Type	boolean
Default	true

Modern site selectors interleave with queue managers, and expect all runnable jobs to be submitted in held state so that they are visible. If your site selector depends on a new mechanism which initially puts jobs on hold, uncomment the following. However, other site selectors work differently, especially our old ones. The DEPRECATED name is "wf.policy.manager", the NEW name is "wf.initially.held".

6.5 wf.queue.manager

System	Euryale run-time
Type	host ':' port
Example	localhost:60010
Default	none

If you use Catalin's new policy manager, you will have to set this option to point to the queue manager. For any other site selector, comment it out.

6.6 wf.replica.pin

System	Euryale run-time
Type	file location path
Default	none

Pinning is the option of prohibiting the file-management subsystem (transfers and replica catalog) to remove a file that was marked to be in use. There is currently no implementation for it, so "null" is as good as it gets.

6.7 wf.popularity.manager

System	Euryale run-time
Type	file location path
Default	none

The popularity manager is invoked from the postscript after a job ran successfully. The interface is simple: The commandline of the application has two arguments, the site handle and a temporary file name. The temporary file contains just a list of LFNs.

Popularity management is part of certain site selectors, and not generally applicable. Thus, this option is usually unset. The special value of "null" avoids forking.

6.8 wf.slow.start

System	Euryale site selector
Type	file location path
Default	none

The slowstart module needs to agree with tailstatd on the database where the window sizes are kept for the workflow. Make this a relative file that is created in the working directory where the workflow runs, where tailstatd is started, and where the site selectors are started.

This option is specific to the "slow start" site selector. It does not hurt others.

6.9 wf.cache.manager

System	Kavitha’s site selector
Type	file location path
Default	none

Used by Kavitha’s site selector, experimental.

6.10 wf.data.scheduler

System	Kavitha’s site selector
Type	file location path
Default	none

Used by Kavitha’s site selector, experimental.

6.11 wf.popularity.table

System	Kavitha’s site selector
Type	file location path
Default	none

Used by Kavitha’s site selector, experimental.

7 Catalogs

There are three catalogs central to the planning process. The site catalog (SC) contains knowledge about the site layout, where directories are, what gatekeeper and gridftp server to use. The transformation catalog (TC) contains knowledge about where applications are installed and what extra information they may require to invoke them. Finally, the replica catalog (RC) keeps track of files that were produced.

Properties are passed as set of property values with a common prefix to their respective handling catalog. The `style` property usually determines the implementation.

prefix	**catalog**
`wf.rc`	replica catalog (RC)
`wf.pool`	site catalog (SC)
`wf.tc`	transformation catalog (TC)

This feature permits to specify more properties than are documented here. This section shows the most basic and required properties for each catalog.

7.1 wf.pool.style

System	Euryale run-time
Type	Enumeration
Value[0]	old
Value[1]	new
Value[2]	xml
Default	none, mandatory
See also	wf.pool.file, section 7.2 (page 15)

Several different formats provide equivalent information about the remote site layout. You have to chose one, XML is recommended.

old	multi-column textual format.
new	multi-line textual format.
xml	XML-formatted text.

7.2 wf.pool.file

System.	Euryale run-time
Type	file location path
Default	none, mandatory
See also	wf.pool.style, section 7.1 (page 14)

The site catalog is kept in a file. The format of the file is determined by vds.pool.style. This option determines the location of the file.

7.3 wf.tc.style

System	Euryale run-time
Type	Enumeration
Value[0]	old
Value[1]	new
Value[2]	vds
Default	none, mandatory
See also	wf.tc.file, section 7.4 (page 15)

The transformation catalog can be kept in various formats, with increasing benefits. The database format relies on the correct setup of the java-based properties relating to the TC schema and driver, as the database is shared between Pegasus and Euryale. The database format is new and experimental code, use "new" for production.

old	old multi-column format, deprecated
new	new multi-column textual format
vds	database driven textual format

The `vds` format parses the GVDS properties to determine the connection information to the TC database.

7.4 wf.tc.file

System	Euryale run-time
Type	file location path
Default	none, mandatory

If the transformation catalog style is "old" or "new", then the transformation catalog is kept in a file. The format of the file is

7.5 wf.tc.keep

System	Euryale run-time
Type	boolean
Default	?

Historically, the TC permitted transformation name separators to be underscore (_) instead of colon (:). However, all TCs should now permit proper transformation names, which corresponds to the "true" value.

This value has been deprecated. Interpretation was removed. Please remove from existing property files.

7.6 wf.rc.style

System	Euryale run-time
Type	Enumeration
Value[0]	LRC
Value[1]	DBI
Value[2]	vds
Value[3]	RLS
Default	none, mandatory

There are two basic replica catalog styles supported by Euryale:

LRC	Use RLS at the LRC level only.
DBI	Use a simple table in a local database.
vds	Use the new VDS RC API (experimental).
RLS	Use full RLS, lookups from an RLI.

For the LRC-based catalog, it makes sense to have a proper setup of Globus and GVDS tools. The environment variables JAVA_HOME, VDS_HOME and GLOBUS_LOCATION should all be set to sensible values.

The DBI-based catalog requires a simple table in a DBI-reachable database of your choice, including SQLite2.

```
create table RC_MAP (
LFN  VARCHAR(255),
PFN  VARCHAR(255),
UNIQUE(LFN,PFN)
);
create index RC_MAX_IDX on RC_MAP(LFN);
```

7.7 wf.rc.lrc

System	Euryale LRC and RLS RC
Type	RLS URI
Default	none, mandatory for LRC
See also	wf.rc.style, section 7.6 (page 16)

The LRC is where knowledge about staged or stagable files are kept.

7.8 wf.rc.rli

System	Euryale RLS RC
Type	RLS URI
Default	${wf.rc.lrc}
See also	wf.rc.lrc, section 7.7 (page 17)

The RLI is an index server to map which LRC has knowledge where files are kept. This one is only available in RLS mode. If unset, it defaults to the value for the LRC.

7.9 wf.rc.java.home

System	Euryale LRC and RLS RC
Type	directory location
Default	none
See also	wf.rc.lrc, section 7.7 (page 17)

This option provides the location of JAVA_HOME, should the environment variable be unset for some reason.

7.10 wf.rc.globus.location

System	Euryale LRC and RLS RC
Type	directory location
Default	none
See also	wf.rc.lrc, section 7.7 (page 17)

This option provides the location of GLOBUS_LOCATION, should the environment variable be unset for some reason.

7.11 wf.rc.vds.home

System	Euryale LRC and RLS RC
Type	directory location
Default	none
See also	wf.rc.lrc, section 7.7 (page 17)

This option provides the location of VDS_HOME, should the environment variable be unset for some reason.

7.12 wf.rc.grc

System	Euryale LRC and RLS RC
Type	file location path
Default	none
See also	wf.rc.lrc, section 7.7 (page 17)

This option points to your globus-rls-cli application. Usually, there is no need to set it: $GLOBUS_LOCATION/bin/globus
rls-cli

7.13 wf.rc.rls.client

System	Euryale LRC and RLS RC
Type	file location path
Default	none
See also	wf.rc.lrc, section 7.7 (page 17)

This option points to your rls-client application. Usually, there is no need to set it: $VDS_HOME/bin/rls-client

7.14 wf.rc.pool

System	Euryale LRC and RLS RC
Type	site handle
Default	none
See also	wf.rc.lrc, section 7.7 (page 17)

Do not set.

7.15 wf.rc.uri

System	Euryale DBI RC
Type	DBI URI
Default	none

Please refer to Perl's DBI manpage and the various DBD documents how to set this driver-specific URI.

7.16 wf.rc.dbuser

System	Euryale DBI RC
Type	string
Default	none

This string provides the name to use with the DBI database engine when connecting.

7.17 wf.rc.dbpass

System	Euryale DBI RC
Type	string
Default	none

This string provides the database account password. The password is usually required for relation database systems on a remote machine.

8 Euryale Behavior

8.1 wf.job.retries

System	Euryale run-time
Type	int
Default	?

This option specifies the number of retries for a job before DAGMan gives up. For naughty grids you may want to increase it. Also note that a bug in 6.5.5 DAGMan will retry the prescript an infinite number of times. A number of 10 is recommended.

8.2 wf.site.temp.unlink

System	Euryale run-time
Type	boolean
Default	true
See also	wf.site.selector, section 6.1 (page 11)

If set to true, the temporary files generated for call-outs are removed after the call-out finished. The only reason to set to false is for debugging purposes.

8.3 wf.site.temp.suffix

System	Euryale run-time
Type	string
Default	.lof
See also	wf.site.selector, section 6.1 (page 11)

This string provides the suffix to be used when generating temporary filenames. Go with the default.

8.4 wf.site.temp.dir

System	Euryale run-time
Type	directory location path
Default	/tmp
See also	wf.site.selector, section 6.1 (page 11)

This option can be used to generate temporary files in a different directory. On modern Linux systems, /dev/shm is a fast directory in RAM. On even more modern Linux systems, /tmp is also sometimes a fast directory in RAM.

8.5 wf.job.keep.output

System	Euryale run-time
Type	boolean
Default	true
See also	wf.job.keep.error, section 8.6 (page 20)

Those files on the submit host which capture the remote stdout are usually removed, if they are empty. If set to "true", even empty files are retained.

8.6 wf.job.keep.error

System	Euryale run-time
Type	boolean
Default	true
See also	wf.job.keep.output, section 8.5 (page 20)

Those files on the submit host which capture the remote stderr are usually removed, if they are empty. If set to "true", even empty files are retained.

8.7 wf.bad.timeout

System	Euryale run-time
Type	positive integer
Default	86400

Bad sites are periodically retried, if they eminate from the global bad file. This is the timeout for re-admissions of bad sites. This is a prototypical feature, do not use.

8.8 wf.dump.variables

System	Euryale run-time
Type	boolean
Default	false

The pre script, for debugging purposes, can dump certain state variables into a file for perusal.

8.9 wf.keep.rewrite

System	Euryale run-time
Type	boolean
Default	false

If this option is active, the raw submit file is kept, even after

8.10 wf.max.idletime

System	Euryale run-time, workflow monitor daemon
Type	non-negative integer
Default	14400

This property configures the maximum time a job can stay pending (or idle) in the remote queue, before Condor is instructed to automatically remove it. This is to avoid queue starvation, but may have undesired consequences on loaded systems.

9 Workflow Monitoring Daemon Controls

The Condor DAGMan can be monitored by a VDS entity called the tailstat daemon (tailstatd). The daemon is started automatically by the vds-run script. It will update the WF tables with the current state of the workflow, how it exited, and for each job it will update the job state tables.

9.1 wf.tailstatd.show

System	workflow monitor daemon
Since	1.4.2
Type	boolean
Default	false

If the value evaluates to a true value (true, yes, on, 1), the workflow monitoring daemon will run the contributed software show-run and show-job after DAGMan finished.

9.2 wf.tailstatd.fuse

System	workflow monitor daemon
Since	1.4.2
Type	seconds
Default	300

Sometimes, the plotting programs can run away for some reason. In order to limit the damage of run-away programs, the workflow monitor can impose a limit on the run-time of the plotting programs. If the limit is exceeded, the plotting program is killed (alas, it may not be perfect yet). The default of a 5 minutes wait interval should be sufficient for all but the largest cases or low-resource machines.

10 Workflow Management

Some items relating more to workflow management by the vds-plan and vds-run tools are kept in the wfrc file. They will move eventually.

The workflow management software uses advanced property parsing techniques. It permits to overwrite arbitrary property settings on the command-line, using Java-style -D options. However, these options are only recognized by the workflow management software.

10.1 wf.properties

System	workflow management, Euryale planner
Type	file location path
Default	${user.home}/.wfrc

The properties permits the workflow management software to overwrite the default location of the property files.

10.2 wf.base.dir

System	workflow managemant, Euryale
Type	directory location path
Default	none, mandatory

The vds-plan and vds-run tools work with the workflow database. However, in order to store planned, past and executing workflow, they create subdirectories underneath some base directory. You specify here where they are to create the tree of workflow directories.

10.3 wf.tailstatd

System	workflow monitor
Type	file location path
Default	${vds.home}/bin/tailstatd

The workflow monitoring daemon is called `tailstatd` and usually resides in the `bin` directory. This property permits to overwrite the default location of the workflow monitoring daemon.

10.4 work.db

System	workflow manager
Type	Perl DBD identifier
Value[0]	Pg
Value[1]	SQLite2
Value[2]	mysql
Default	none, mandatory

The workflow information is kept inside the same database as the VDC and PTC. In order to access the tables from perl, access information must be provided. This option declares the DBD module that is used to access the WF database. This is research.

10.5 work.db.database

System	workflow manager
Type	database (schema) identifier
Default	none, mandatory

Any rDBMS permits several users to have several "databases".

10.6 work.db.user

System	workflow manager
Type	account name
Default	none

The account name of the WF database.

10.7 work.db.password

System	workflow manager
Type	account name
Default	none

The account password for the WF database.

11 Dagman Control

The Euryale submission software to DAGMan wraps the Condor DAG submit command. The wrapping is necessary to install sensible throttles for the Euryale workflow management.

11.1 vds.dagman.maxpre

System	Euryale workflow submit
Type	non-negative integer
Default	20
See Also	vds.dagman.maxpost, section 11.2 (page 24)

This property limits the number of concurrently active pre scripts in DAGMan. The value of 20 has shown to be sensible. Pre scripts usually generate a heavier load than post scripts.

11.2 vds.dagman.maxpost

System	Euryale workflow submit
Type	non-negative integer
Default	20
See Also	vds.dagman.maxpre, section 11.1 (page 24)

This property limits the number of concurrently active post scripts in DAGMan. The value of 20 has shown to be sensible. Post scripts usually generate JVM load when parsing exit codes, but are otherwise light weight.

11.3 vds.dagman.maxjobs

System	Euryale workflow submit
Type	positive integer
Default	unlimited (0)

DAGMan can throttle the maximum number of jobs it makes visible to the Condor sub-system. Usually, the throttles in Condor-G are sufficient.

11.4 vds.dagman.notify

System	Euryale workflow submit
Type	Enumeration
Value[0]	NEVER
Value[1]	ERROR
Value[2]	COMPLETE
Default	NEVER

Our default is not to send any email messages upon completion of DAGMan. Condor's default of *complete* is thus overwritten.

11.5 vds.dagman.verbose

System	Euryale workflow submit
Type	boolean
Default	false

If set to true, has DAGMan report more about the job submission.

Index