Inca::Reporter API Reference

Inca::Reporter ->**new**(
  [**body** => $str] [, **completed** => $bool] [, **description** => $str] [, **fail_message** => $str]
  [, **url** => $str] [, **version** => $str]
)

addArg(
  $name, $description [, $default [, $pattern]]
)
Add a reporter-specific command-line argument to those recognized

addDependency($package [, $package …])
Register reporter package dependencies

argValue($name [, $position])
Retrieve a single command-line argument value

argValues($name)
Retrieve an array of command-line argument values

compiledProgramOutput(
  code => $str [, **compiler** => $str]
  [, **language** => $str] [, **out_switch** => $str]
  [, **switches** => $str] [, **timeout** => $int]
)
Compile and run a program; return stdout

failPrintAndExit($message)
Exit after printing reporter failure XML

getBody(), **getCompleted**(),
getDescription(), **getFailMessage**(), **getUrl**(),
getVersion()
setBody($body), **setCompleted**($completed),
setDescription($description),
setFailMessage($message), **setUrl**($url),
setVersion($version)
Accessor methods

log($type, $message [, $message …])
Add messages to the reporter log

loggedCommand($command [, $timeout])
Log and run a command; return stdout

print([$verbose])
Print the reporter XML

processArgv(@ARGV)
Parse command-line arguments; handle -help and -version

report([$verbose])
Construct and return reporter XML

reportBody()
Abstract method for derived classes; constructs and returns the body XML

setResult($completed [, $message])
Combines setCompleted & setFailMessage

tempFile($path [, $path …])
Register files to be deleted on exit

xmlElement(
  $name, $escape, $content [, $content …]
)
Return a formatted XML tag and contents

Inca::Reporter::SimpleUnit API Reference

Inca::Reporter::SimpleUnit ->**new**([**unit_name** => $str])
See also Inca::Reporter

getUnitName()
setUnitName($name)
Accessor methods

reportBody()
Construct and return XML for the body of the report

unitFailure($message)
Indicate failure of the unit test

unitSuccess($name)
Indicate success of the unit test

Inca::Reporter::GlobusUnit API Reference

Inca::Reporter::GlobusUnit ->**new**()
See also Inca::Reporter, Inca::SimpleUnit

submitCSource(
code => $str [, **arguments** => $str] [, **check** => $int] [, **cleanup** => $bool] [, **count** => $int]
[, **debug** => $bool] [, **duroc** => $bool] [, **env** => $str] [, **host** => $str] [, **mpi** => $bool]
[, **service** => $str] [, **timeout** => $int]
)
Compile and submit a C program; return a two-element array of [stdout, stderr]

submitJob(
executable => $str [, **arguments** => $str] [, **check** => $int] [, **cleanup** => $bool]
[, **count** => $int] [, **debug** => $bool] [, **duroc** => $bool] [, **env** => $str] [, **host** => $str]
[, **mpi** => $bool] [, **remote** => $bool] [, **service** => $str] [, **timeout** => $int]
)
Submit a job; return a two-element array of [stdout, stderr]

Inca::Reporter::Performance API Reference

Inca::Reporter::Performance ->**new**([**test_name** => $str])
See also Inca::Reporter

addBenchmark($name, $benchmark)
Add a benchmark to the list of those generated by the reporter

getTestName()
setTestName($name)
Accessor methods

reportBody()
Construct and return XML for the body of the report

Inca::Reporter::Version API Reference

Inca::Reporter::Version ->**new**([**package_name** => $str] [, **package_version** => $str])
See also Inca::Reporter

getPackageName(), **getPackageVersion**(), **getSubpackageNames**(),
getSubpackageVersion($name)
setPackageName($name), **setPackageVersion**($version),
setSubpackageVersion($name, $version)
Accessor methods

reportBody()
Construct and return XML for the body of the report

setVersionByCompiledProgramOutput(
code => $str [, **compiler** => $str] [, **language** => $str] [, **out_switch** => $str]
[, **pattern** => $str] [, **switches** => $str] [, **timeout** => $int]
)
Compile and run a program; obtain version from a pattern in its stdout

setVersionByExecutable($command [, $pattern] [, $timeout])
Run a program; obtain version from a pattern in its stdout

setVersionByFileContents($path [, $pattern])
Obtain version by looking for a pattern in a file

setVersionByGptQuery($prefix [, $prefix])
Obtain subpackage versions by looking for gpt packages that begin with certain prefixes

setVersionByRpmQuery($pattern)
Obtain subpackage versions by looking for rpm packages matching a given pattern

Inca Reporter Tutorial

Write some reporters:

1) A Version reporter that determines the installed version of Perl ("perl –v" reports this; see cluster.compiler.gcc.version)

2) A SimpleUnit reporter that determines whether the command "wget –O /dev/null http://www.cnn.com/index.html" succeeds (see data.transfer.gridftp.unit.copy). Allow the user to specify a different web page via the -page command-line option.

3) Modify a copy of the SimpleUnit reporter above to produce a Performance reporter that records the bandwidth, in kB/second, reported by wget (see grid.benchmark.performance.ping).

4) A Reporter that records the pid, user, cpu %, and command of the non-root process using the highest cpu % on the system ("ps waux" reports this; see cluster.filesystem.scratch.level)

5) (Extra credit) A Version reporter that determines the subpackage versions of all installed CPAN modules ("perldoc –t perllocal" reports this; see cluster.rpms)

6) (Extra credit) A reporter that determines the installed version of bash ("bash --version" reports this), written in a language other than Perl

To run your Perl reporters, make sure your PERL5LIB environment variable includes the path to the Perl Inca Reporter libraries.

Try running with -help, -version, and -log to see how the output changes.

Create a repository catalog for your reporters by running incpack in the directory where you created your reporters. If you created #6 above, you'll need first to create a *.attrib file for the reporter.

Answer Key—Reporters to Write

1) A Version reporter that determines the installed version of Perl

```
use Inca::Reporter::Version;
my $reporter = new Inca::Reporter::Version(
  version => 1.0,
  description => 'Reports the version of perl',
  url => 'http://cpan.org',
  package_name => 'perl'
);
$reporter->processArgv(@ARGV);
$reporter->setVersionByExecutable
  ('perl -v', 'perl, v([\d\w\.\-]+)');
$reporter->print();
```

2) A SimpleUnit reporter that determines whether a web page can be downloaded via wget

```
use Inca::Reporter::SimpleUnit;
my $reporter = new Inca::Reporter::SimpleUnit(
  version => 1.0,
  description => 'Reports whether wget can download a web page',
  url => 'http://inca.ucsd.edu',
  unit_name => 'wget'
);
$reporter->addArg('page', 'web page url to download',
                  'http://cnn.com/index.html');
$reporter->processArgv(@ARGV);
my $page = $reporter->argValue('page');
my $output =
  $reporter->loggedCommand("wget -O /dev/null $page");
if(!defined($output)) {
  $reporter->unitFailure("wget command failed: $!");
} elsif($?) {
  $reporter->unitFailure("wget command failed: $output");
} else {
  $reporter->unitSuccess();
}
$reporter->print();
```

3) A Performance reporter that records the bandwidth reported by wget.

```
use Inca::Reporter::Performance;
use Inca::Reporter::Performance::Benchmark;
my $reporter = new Inca::Reporter::Performance(
  version => 1.0,
  description => 'Reports the bandwidth reported by wget',
  url => 'http://inca.ucsd.edu',
  test_name => 'wget'
);
$reporter->addArg('page', 'web page url to download',
                  'http://cnn.com/index.html');
$reporter->processArgv(@ARGV);
my $page = $reporter->argValue('page');
my $output =
  $reporter->loggedCommand("wget -O /dev/null $page");
if(!defined($output)) {
  $reporter->setResult(0, "wget command failed: $!");
} elsif($? || $output !~ /([\d.]+) KB\/s/) {
  $reporter->setResult(0, "wget command failed: $output");
} else {
  my $benchmark = new Inca::Reporter::Performance::Benchmark();
  $benchmark->setStatistic('bandwidth', $1, 'kilobytes/second');
  $reporter->addBenchmark('download', $benchmark);
  $reporter->setResult(1);
}
$reporter->print();
```

4) A Reporter that determines the pid, user, cpu %, and command of the non-root process using the highest cpu % on the system

```
use Inca::Reporter;
my $reporter = new Inca::Reporter(
  version => 1.0,
  description => 'Reports top non-root CPU % process',
  url => 'http://inca.sdsc.edu'
);
$reporter->processArgv(@ARGV);
my $output = $reporter->loggedCommand('ps waux');
$reporter->failPrintAndExit("ps failed: $!")
  if !defined($output);
my $highestCpu = 0;
my $highestLine;
foreach my $line(split(/\n/, $output)) {
  next if $line =~ /%|^root/; # Skip header and root procs
  my @columns = split(/\s+/, $line);
  my $cpu = $columns[2];
  next if $cpu < $highestCpu;
  $highestCpu = $cpu;
```

```
  $highestLine = $line;
}
$reporter->failPrintAndExit('No non-root process found')
  if !defined($highestLine);
my @columns = split(/\s+/, $highestLine);
$reporter->setBody(
  $reporter->xmlElement('topcpu', 0,
    $reporter->xmlElement('ID', 1, 'topcpu'),
    $reporter->xmlElement('user', 1, $columns[0]),
    $reporter->xmlElement('pid', 1, $columns[1]),
    $reporter->xmlElement('cpu', 1, $columns[2]),
    $reporter->xmlElement('command', 1, $columns[$#columns]),
  )
);
$reporter->setResult(1);
$reporter->print();
```

5) A Version reporter that determines the subpackage versions of all installed CPAN modules

```
use Inca::Reporter::Version;
my $reporter = new Inca::Reporter::Version(
  version => 1.0,
  description => 'Reports versions of installed Perl modules',
  url => 'http://www.cpan.org'
);
$reporter->processArgv(@ARGV);
my $output = $reporter->loggedCommand('perldoc -t perllocal');
$reporter->failPrintAndExit("perldoc failed: $!")
  if !defined($output);
my $currentModule = '';
foreach my $line(split(/\n/, $output)) {
  if($line =~ s/^.*"Module"\s*//) {
    $currentModule = $line;
  } elsif($line =~ /VERSION: ([^"]+)/) {
    $reporter->setSubpackageVersion($currentModule, $1);
  }
}
$reporter->print();
```

6) A reporter that determines the version of bash installed on the system, written in any language other than Perl

```
#! /bin/sh
reporterName=cluster.admin.bash.version
reporterVersion=1.0

# parse argv
help=no; log=0; verbose=1; version=no
for a in $@; do
```

```
  case $a in
    *=*) argVal=`echo ${a} | sed 's/.*=//'`;;
    *) argVal=yes
  esac
  arg=`echo ${a} | sed s/.// | sed 's/=.*//'`
  eval "${arg}=${argVal}"
done

# special processing for -version and -help
if test ${version} = 'yes'; then
  /bin/echo "${reporterName} ${reporterVersion}"
  exit 0
elif test ${help} = 'yes'; then
  if test ${verbose} eq 0; then
    # TODO
    exit 0
  fi
  helpXml='' # TODO
fi

# determine host name and gmt for XML
hostname=`hostname`
datestr=''
for w in `date -u | sed 's/:/ /g'`; do
  case $w in
    Jan)w=1;; Feb)w=2;; Mar)w=3;; Apr)w=4;; May)w=5;; Jun)w=6;;
    Jul)w=7;; Aug)w=8;; Sep)w=9;; Oct)w=10;; Nov)w=11;; Dec)w=12;;
  esac
  datestr="${datestr}${w} "
done
gmt=`/bin/echo ${datestr} |
  awk ' {printf "%04d-%02d-%02dT%02d:%02d:%02dZ", $8, $2, $3, $4, $5, $6}'`

if test -e /bin/bash; then
  completed=true
  version=`bash --version | grep version | sed 's/.*version //' |
    sed 's/ .*//'`
  bodyXml="<package><ID>bash</ID><version>${version}</version></package>"
  messageXml=''
else
  completed=false
  bodyXml='<body/>'
  message='bash not installed'
  messageXml="<fail_message>${message}</fail_message>"
fi

if test ${verbose} -gt 0; then
  /bin/echo "xml version='1.0'?>
<rep:report xmlns:rep='http://inca.sdsc.edu/dataModel/report_2.1'>
  <gmt>${gmt}/gmt>
  <hostname>${hostname}</hostname>
  <name>${reporterName}</name>
  <version>${reporterVersion}</version>
  <args>
    <arg>
      <name>help</name>
      <value>${help}</value>
    </arg>
    <arg>
      <name>log</name>
```

```
      <value>${log}</value>
    </arg>
    <arg>
      <name>verbose</name>
      <value>${verbose}</value>
    </arg>
    <arg>
      <name>version</name>
      <value>${version}</value>
    </arg>
  </args>
  ${bodyXml}
  <exitStatus>
    <completed>${completed}</completed>
    ${messageXml}
  </exitStatus>
  ${helpXml}
</rep:report>
" | grep -v '^ *$'
elif test "${completed}" = "true"; then
  /bin/echo 'completed'
else
  /bin/echo "failed: ${message}"
fi
```