
Administering Platform Application Center

Platform Application Center
Version 8.0.2
November 2011



Copyright

© 1994-2011 Platform Computing Inc.

Although the information in this document has been carefully reviewed, Platform Computing Corporation ("Platform") does not warrant it to be free of errors or omissions. Platform reserves the right to make corrections, updates, revisions or changes to the information in this document.

UNLESS OTHERWISE EXPRESSLY STATED BY PLATFORM, THE PROGRAM DESCRIBED IN THIS DOCUMENT IS PROVIDED "AS IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL PLATFORM COMPUTING BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING WITHOUT LIMITATION ANY LOST PROFITS, DATA, OR SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

We'd like to hear from you

You can help us make this document better by telling us what you think of the content, organization, and usefulness of the information. If you find an error, or just want to make a suggestion for improving this document, please address your comments to doc@platform.com.

Your comments should pertain only to Platform documentation. For product support, contact support@platform.com.

Document redistribution and translation

This document is protected by copyright and you may not redistribute or translate it into another language, in part or in whole.

Internal redistribution

You may only redistribute this document internally within your organization (for example, on an intranet) provided that you continue to check the Platform Web site for updates and update your version of the documentation. You may not make it available to your organization over the Internet.

Trademarks

LSF is a registered trademark of Platform Computing Corporation in the United States and in other jurisdictions.

ACCELERATING INTELLIGENCE, PLATFORM COMPUTING, PLATFORM SYMPHONY, PLATFORM JOB SCHEDULER, PLATFORM ISF, PLATFORM ENTERPRISE GRID ORCHESTRATOR, PLATFORM EGO, and the PLATFORM and PLATFORM LSF logos are trademarks of Platform Computing Corporation in the United States and in other jurisdictions.

UNIX is a registered trademark of The Open Group in the United States and in other jurisdictions.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

ANSYS, ANSYS Workbench, AUTODYN, CFX, FLUENT and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries in the United States or other countries. [ICEM CFD is a trademark used by ANSYS, Inc. under license.]* All other brand, product, service and feature names or trademarks are the property of their respective owners.

NASTRAN is a registered trademark of the National Aeronautics Space Administration.

MSC Nastran is an enhanced proprietary version developed and maintained by MSC Software Corporation.

ABAQUS is a registered trademark of ABAQUS, Inc.

Other products or services mentioned in this document are identified by the trademarks or service marks of their respective owners.

Third-party license agreements

<http://www.platform.com/Company/third.part.license.htm>

Contents

1	Platform Application Center Overview	5
	Platform Application Center	6
	Manage hosts	8
2	Configure and Manage Platform Application Center	11
	Platform Application Center Licenses	12
	Enable viewing license usage through Platform Application Center	14
	Platform Analytics reports in Platform Application Center	15
	Job directories, repositories, and shared directories	17
	Prepare and publish an application	21
	Create an Oracle database schema	22
	Setting access controls	24
	Configure Single URL for Failover	32
	Enabling Remote Consoles	36
	Job data purging	39
	Adding custom job actions	41
	Platform Application Center log files and log levels	43
3	Platform Application Center Reporting	45
	Overview of reporting	46
	Introduction to reporting	47
	Standard reports	48
	Custom reports	50
	System description	53
	Reports administration	56
	Test the reporting feature	65
4	Platform Application Center Web Services	67
	Introduction to Platform Application Center Web Services	68
	pacclient.py	70
	RESTful web service reference	75
	Python API reference	91
5	Create Custom Pages with the SDK	99
	Overview	100
	Custom page examples	101
	Integrate a custom page into Platform Application Center	104
	Page customization framework XML reference	105

6	Platform Application Center Reference	123
	perfadmin	124
	perfremoverc.sh	126
	perfsetrc.sh	127
	pmcadmin	128
	pmcremoverc.sh	130
	pmcsetrc.sh	131

Platform Application Center Overview

Platform Application Center

Platform Application Center allows users and administrators to monitor hosts and to submit and monitor jobs.

Open Platform Application Center for the first time

Launch Platform Application Center to monitor hosts and jobs and run jobs.

Platform Application Center allows you to monitor hosts and jobs, as well as submit jobs.

All administrators and non-administrator users logging on to Platform Application Center must provide an OS user name and password.

1. Open Platform Application Center in a supported Web browser.
`http://pac_host:8080/platform`
2. Log on with your username and password.
3. If the log on failed or the URL could not be found, restart Platform Application Center.
 - If EGO is disabled, run `pmcadmi n stop` then `pmcadmi n start` from the command line.
 - If EGO is enabled, run `egosh service stop WEBGUI` then `egosh service start WEBGUI` from the command line.

In most cases, the services required to run Platform Application Center start automatically; however, if Platform Application Center goes down, you may need to restart Platform Application Center services and daemons manually.

Enable cookies

Platform Application Center requires that Web browsers accept first-party and third-party cookies. In some cases, your browser default settings may block these cookies. In this case, you need to manually change this setting.

Note:

By default, IE7 blocks cookies.

For IE7:

1. In your browser, go to Tools > Internet Options > Privacy > Advanced.
2. Select Override automatic cookie handling.
3. Select Accept for both first-party and third-party cookies.
4. Click OK to save your settings.

Language support

Platform Application Center displays the language preference set in your web browser. If you set a new language preference or change the order of your preferred languages in your browser options, you must log out and log back in to Platform Application Center to make the change take effect. See your browser help for more information about setting language preferences.

Start Platform Application Center services and log on as administrator

With UNIXroot privileges, you can use the `pmcadmi n` command to administer Platform Application Center.

Platform Application Center allows you to monitor, administer, and configure your cluster.

1. Find out which host is running Platform Application Center.

- If EGO is enabled, run:

```
egosh service list -s WEBGUI
```

- If EGO is disabled, run:

```
pmcadmi n list
```

2. Browse to the host:

```
http://SWEBGUI_HOST: 8080
```

3. Log in.

To see administrator features in Platform Application Center, you must log in as an administrator.

Manage hosts

Hosts in the cluster can function as the master host, master candidates, management hosts, compute hosts, or the web server host.

Master host

A cluster requires a master host. This is the first host installed. The master host controls the rest of the hosts in the grid.

Master candidates

There is only one master host at a time. However, if the master host ever fails, another host automatically takes over the master host role, allowing work to continue. This process is called failover. When the master host recovers, the role switches back again.

Hosts that can act as the master are called master candidates. This includes the original master host and all hosts that can take over the role in a failover scenario. All master candidates must be management hosts.

Master host failover

During master host failover, the system is unavailable for a few minutes while hosts are waiting to be contacted by the new master.

The master candidate list defines which hosts are master candidates. By default, the list includes just one host, the master host, and there is no failover. If you configure additional candidates to enable failover, the master host is first in the list. If the master host becomes unavailable, the next host becomes the master. If that host is also unavailable, the next host is considered to become the master, and so on down the list. A short list with two or three hosts is sufficient for practical purposes.

For failover to work properly, the master candidates must share a file system and the shared directory must always be available.

Important:

The shared directory should not reside on a master host or any of the master candidates. If the shared directory resides on the master host and the master host fails, the next candidate cannot access the necessary files.

Compute host

Compute hosts are distributed to cluster consumers to execute workload units. By default, compute hosts belong to the `ComputeHosts` resource group.

The `ComputeHosts` group excludes hosts with the `mg` tag, which is assigned to management hosts when you run `egoconfig mghost`. If you create your own resource groups to replace `ComputeHosts`, make sure they also exclude hosts with the `mg` tag.

By default, the number of slots on a compute host is equal to the number of CPUs.

Web server host or Platform Application Center host

The web server is the host that runs the Platform Application Center. Only one host at a time acts as the Platform Application Center web server host. If EGO controls the Platform Application Center, the web server host can be any management host in the cluster. Which host is the web server host is decided at cluster startup. The web server host fails over if the original host goes down. If EGO does not control the Platform Application Center, you must configure the Platform Application Center host manually. If you specify the Platform Application Center host, there is no failover of the Platform Application Center.

Configure and Manage Platform Application Center

Platform Application Center Licenses

No License

When no license is installed, you can still use Platform Application Center, but functionality is limited and not all features are available.

License installed

When the `pcc_app_center_enterprise` license is installed, all functionality and features are available.

View available licenses

1. Choose Settings > PAC License.

Information about installed Platform Application Center license features is displayed:

- Total licenses
- Used licenses
- Rejected licenses

Install the license file

1. Log on to the Platform Application Center host as root.
2. Set your Platform Application Center environment:

For example:

- For `csch` or `tcsh`:

```
% source /opt/pac/cshrc.platform
```
- For `sh`, `ksh`, or `bash`:

```
$ ./opt/pac/profile.platform
```

3. In `$GUI_CONFIGDIR/pmc.conf`, specify the full path to your license file with the parameter `PAC_LICENSE_FILE`.

For example:

```
PAC_LICENSE_FILE=/tmp/license.dat
```

4. Restart the web server.

```
pmcadmin stop
perfadmin stop all
pmcadmin start
perfadmin start all
```

5. Log on to Platform Application Center and select Settings > PAC License.

You should be able to see your new license information.

Configure how Platform Application Center checks out licenses from a license server

When you use a license server such as FLEXlm, you may want to limit the number of licenses that one installation of Platform Application Center checks out.

By default, Platform Application Center checks out all available licenses when the first user logs in and does not release them until Platform Application Center is shut down. If you have multiple installations of Platform Application Center running at the same time, you can limit the number of licenses used by each installation by changing the parameter `MaxRequestedLicenses` in `pmc.conf`. In this way, you can share the available Platform Application Center licenses across your multiple installations.

1. Log on to the Platform Application Center host as root.
2. Set your Platform Application Center environment:

For example:

- For `csh` or `tcsh`:

```
% source /opt/pac/cshrc.platform
```
- For `sh`, `ksh`, or `bash`:

```
$ . /opt/pac/profile.platform
```

3. In `$GUI_CONFIGDIR/pmc.conf`, specify the maximum number of licenses this installation of Platform Application Center can check out with the parameter `MaxRequestedLicenses`.

This will limit the number of concurrent users to the number you specify.

For example, to limit the number of licenses that are checked out to 10, set:

```
MaxRequestedLicenses=10
```

4. Restart the web server.

```
pmcadmin stop
perfadmin stop all
pmcadmin start
perfadmin start all
```

5. Log on to Platform Application Center and select Settings > PAC License.

You should be able to see in the Used column the number you set for `MaxRequestedLicenses`.

Enable viewing license usage through Platform Application Center

Platform Application Center can display license usage when Platform License Scheduler is installed and licensed.

Platform Application Center displays Resources > Licenses. You can view license usage by feature, by job, by project, and by cluster.

Requirements:

- Platform License Scheduler 8.0 is installed in your LSF cluster.
- You have an enterprise license for Platform Application Center.
- Platform Application Center 8.0 is installed and can access \$LSF_ENVDIR.

After installation, you need to enable viewing licenses through Platform Application Center.

1. In `$SGUI_CONFIGDIR/pmc.conf`, set `ENABLE_LS_GUI=Y`.
2. Restart the web server.

```
pmcadmin stop  
perfadmin stop all  
pmcadmin start  
perfadmin start all
```

3. Log on to Platform Application Center and select Resources > Licenses.

You should be able to see license usage information.

Platform Analytics reports in Platform Application Center

About Platform Analytics in Platform Application Center

Platform Application Center embeds Platform Analytics. You need to install the Platform Application Center Analytics add-on package. The package comes with installation instructions. You can download the add-on package from the Platform FTP site, in the same location as Platform Analytics.

Access to reports:

- When the add-on has been installed, the Reports in Application Center are replaced with the reports from Platform Analytics. You can access them through Reports > By Workbook.

Users and access control:

- In order to have access to the reports in Platform Application Center, users must be a Viewer, Interactor, or administrator in Tableau. Users can use the same UNIX account name as they use to log on to Tableau.
- Through the Platform Application Center Settings tab, the Cluster administrator can configure access per workbook. Users will only be able to see workbooks for which they have permission in Platform Application Center or in Tableau.

User roles:

Role	Permissions
Report administrator	<p>Platform Application Center automatically loads Tableau Licensed users with the new role Report administrator.</p> <p>A user with the Report administrator role can:</p> <ul style="list-style-type: none"> • Display the report list and view a report • View past reports • Manage report scheduling • Subscribe to a report, or unsubscribe from a report to receive emails when reports are updated • Add extra email addresses for sending reports
Normal user	<p>A user with the Normal user role can:</p> <ul style="list-style-type: none"> • View past reports • Subscribe to a report, or unsubscribe from a report to receive emails when reports are updated
Cluster administrator	<p>A user with the Cluster administrator role can:</p> <ul style="list-style-type: none"> • Configure permissions on who can view a workbook through Platform Application Center's access control • Add emails to user properties for application users through Settings > Users and User Groups List.

Enable viewing Platform Analytics reports

Requirements:

- Platform Analytics 8.0 is installed in your LSF cluster.
- You have an enterprise license for Platform Application Center.
- Platform Application Center 8.0 is installed and can access \$LSF_ENVDIR.
- The Vertica client driver is installed on your Tableau server. You can download the driver package from the platform FTP site, in the same location as Platform Analytics.
- You have installed the Platform Application Center Analytics add-on package. The package comes with installation instructions. You can download the add-on package from the Platform FTP site, in the same location as Platform Analytics.

After installation, you need to enable viewing Platform Analytics reports through Platform Application Center.

1. In `$GUI_CONFDIR/pmc.conf`, set `ENABLE_PA_REPORT=Y`.
2. Restart the web server.

```
pmcadmin stop  
perfadmin stop all  
pmcadmin start  
perfadmin start all
```

3. Log on to Platform Application Center and select Reports > By Workbook.

You should be able to see the Platform Analytics Reports.

Job directories, repositories, and shared directories

You must be an LSF administrator. You must set up read, write, and execute permissions on the directories you specify for the users you specify.

In the following tasks, *PAC_TOP* is the top-level installation directory for Platform Application Center.

Repositories and job directories

Repository

The job repository (*\$repository*) is the location where a user's job files are stored.

By default, the job repository for a user is defined at the cluster level, and the job files for all applications are located together under one directory. However, you can manually configure any application and define a different repository for a user, and the application-level configuration overrides the cluster-level configuration. In this way, a user can have multiple repositories for multiple applications.

The root account must have read and write permission on every job repository.

Job Directory

For each job, job data goes to the *\$repository/\$jobname_timestamp* directory, which is called the job directory. The job directory is created automatically when the user submits the job.

If the job is submitted without a job name, job data goes to the *\$repository/\$applicationname_timestamp* directory.

Configure Repository.xml

This file defines the default repository location. If the application definition file (*appl icat ion_name.xml*) does not have the `<j obdat a>` element present, Platform Application Center uses *Repository.xml* to determine the repository location.

The repository for each user is *repositorylocation/username*, unless the repository location is defined as *\$HOME*.

1. Edit *PAC_TOP/gui/conf/Repository.xml* with a text editor.

This is the same file you use to configure shared directories.

2. For each user, the repository is *repositorylocation/username*. Modify the default repository location or add different locations for different users.

The *repositorylocation/username* subdirectory is not created if the special variable *\$HOME* is used as the repository location.

Multiple repository locations can be defined for different users. One user can have only one repository location, if multiple repository locations are defined for a user, the first repository location will be used.

Each new `<Repository>` element defines a new repository location.

A `<Repository>` element must contain a `<Path>` sub-element.

The repository location is defined by the <Path> sub-element.

- Restart Platform Application Center.
 - If EGO is disabled, run `pmcadmi n stop` then `pmcadmi n start` from the command line.
 - If EGO is enabled, run `egosh service stop WEBGUI` then `egosh service start WEBGUI` from the command line.

Configure application-level repositories

Configuring repositories for applications is optional. If the repository is not configured at the application level, the cluster-level default is used.

- To configure the user repositories for an application, log on as administrator and edit the application definition file with a text editor.

For example:

`PAC_TOP/gui/conf/application/draft/applicationname/applicationname.xml`

- Add the <j obdat a> element is nested under the <reposit ory> element, as shown:

```
<agent>
  <reposit ory>
    <j obdat a>
      . . .
    </j obdat a>
  </reposit ory>
</agent>
```

- The <j obdat a> element specifies the repository location path, and each user's repository is a subdirectory of it. Specify user names as a comma-separated list.

A user can have only one repository per application. If a user's name appears more than once in the same <j obdat a> element, the last location specified is used.

Syntax:

```
<l ocation path="reposit ory_l ocation" user="user_name . . ." />
```

Special Keywords:

- `$HOME`: If "\$HOME" is the repository location path, it means the user's repository is his own home directory, we do not create a subdirectory under \$HOME.
- `al l`: If "all" is the user, it indicates all users not explicitly defined elsewhere within the <j obdat a> element.

- Restart Platform Application Center.

Example:

```
<j obdat a>
<l ocation path="/dat a" user="al l" />
<l ocation path="/abc" user="user1" />
<l ocation path="/xyz" user="user2, user3" />
</j obdat a>
```

For this application only:

- The repository for user1 is /abc/user1. This means user1's job directories for this application are located under /abc/user1.
- The repositories for user2 and user3 are /xyz/user2 and /xyz/user3.
- The repositories for all other users are subdirectories of /dat a.

Shared Directories

An administrator can set up shared directories to be used by a subset of users. Shared directories for a user are also shown on the data management page.

Application users must have read permission on the shared directories.

The root account does not need any permissions on shared directories.

Configure shared directories

Configuring shared directories is optional.

1. Edit `PAC_TOP/gui/conf/Repository.xml` with a text editor.
2. Add shared directories and assign users to access to these directories.

Add as many new `<ShareDirectory>` elements as needed. Each new `<ShareDirectory>` element defines a new shared directory.

Note:

Syntax of `<ShareDirectory>` element

- A `<ShareDirectory>` element must contain a `<Path>` sub-element.
 - Variable `$USER` can be used in `<Path>` under the `<ShareDirectory>` element.
-

For example:

```
<ParamConfs>
  <Configurati on>
    <ShareDi rectory>
      <Path>/shared/common/</Path>
    </ShareDi rectory>
    <ShareDi rectory>
      <Path>/shared/testi ng/$USER</Path>
    </ShareDi rectory>
  </Configurati on></ParamConfs>
```

In this example, the users have access to the following directories:

- user1: `/shared/common/`
- user2: `/shared/common/`
- user3: `/shared/common/` and `/shared/testi ng/user3/`
- user4: `/shared/testi ng/user4/`

Remote Job Directory

For better performance, you may choose to modify an application and enable the remote job directory, which is the local job directory on the execution host. Once the job creates a job directory locally on the execution host, you can operate on the directory through Platform Application Center as if it was a shared data directory.

By default, the applications do not support the remote job directory. If you want to enable the feature, the application script must generate a `bsub` submission script.

For example, you can edit the ABAQUS application form and add this code at the end of the script. The bsub submission script creates the remote job directory before starting the job. You can modify other applications based on this example.

```
#####
# Begin: create bsub submission script
#####
#
BSUB_SCRIPT=$OUTPUT_FILE_LOCATION/bsub. $JOB_NAME
exec 3>&1          # Link file descriptor #3 with stdout.
exec > $BSUB_SCRIPT # stdout replaced with file "bsub. $JOBNAME".
#
echo "#!/bin/sh"
echo "#BSUB $RUNHOST_OPT"
echo "#BSUB $JOB_NAME_OPT"
echo "#BSUB $DECK_OPT"
echo "#BSUB $QUEUE_OPT"
echo "#BSUB $NCPUS_OPT"
echo "#BSUB $OUTPUT_OPT"
echo "#BSUB $MEMARC_OPT"
echo "#BSUB $ABQ_LI C"
echo "#BSUB $EXTRA_PARAMS"
#
# Create the remote job directory on the computes nodes for the job.
# The logic below removes duplicates when using hosts with multiple CPUs.
# "/tmp/$EXECUTIONUSER/" is used as remote job directory in the example code below
#
echo "          echo `hostname` >> "$OUTPUT_FILE_LOCATION"/nodelist"
echo "          echo `hostname`: /tmp/$EXECUTIONUSER/$JOB_NAME >>"
"$OUTPUT_FILE_LOCATION"/.rspooler"
echo "          mkdir -p /tmp/$EXECUTIONUSER/$JOB_NAME"
#
# Copy job files to the local scratch directory.
#
echo "cd /tmp/$EXECUTIONUSER/$JOB_NAME"
echo "cp $OUTPUT_FILE_LOCATION/* /tmp/$EXECUTIONUSER/$JOB_NAME"
#
# Create the EXE_CMD for the application used in the bsub submission script
EXE_CMD="{ABAQUS_CMD} {ABAQUS_CPU_OPT} {ABAQUS_OPTIONS} {OTHER_OPTS} int"
echo "$EXE_CMD"
#
exec 1>&3 3>&- # Restore stdout and close file descriptor #3.
#####
# End: create bsub submission script
#####
JOB_RESULT=`/bin/sh -c "bsub < $BSUB_SCRIPT 2>&1" `
export JOB_RESULT OUTPUT_FILE_LOCATION
${GUI_CONFIG_DIR}/application/job-result.sh
```

Prepare and publish an application

By default, users can only submit jobs using a generic application form. The built-in applications must be edited before they can work properly in your system.

1. Log in as administrator.
2. Choose Resources > Submission Templates > Application Templates.
3. Select an application that has not been published before. You may publish and unpublish as often as you like, but you are only prompted for configuration information once.
4. Choose Publish. Fill in the required information and click confirm as prompted.

Only published applications are available to users.

Create an Oracle database schema

Follow these instructions to:

- Create database schemas in Oracle to support Platform Application Center data

1. Untar the schema package.

```
tar -xvf pcc-appcenter-8.0.2-dbschema.tar
```

2. Go to the Oracle directory.

```
cd DBschema/Oracle
```

3. In the command console, run the script to create the EGO database schema.

```
sqlplus user_name/password@connect_string @create_egobasic_rawdata_schema.sql  
data_tablespace index_tablespace
```

where

- *user_name* is the user name on the database.
- *password* is the password for this user name on the database.
- *connect_string* is the named SQLNet connection for this database.
- *data_tablespace* is the name of the tablespace where you intend to store the table schema.
- *index_tablespace* is the name of the tablespace where you intend to store the index.

4. Run the script to create the LSF database schema.

```
sqlplus user_name/password@connect_string @create_lsfbasic_rawdata_schema.sql  
data_tablespace index_tablespace
```

where

- *user_name* is the user name on the database.
- *password* is the password for this user name on the database.
- *connect_string* is the named SQLNet connection for this database.
- *data_tablespace* is the name of the tablespace where you intend to store the table schema.
- *index_tablespace* is the name of the tablespace where you intend to store the index.

5. Run the scripts to create the access control list tables and initialize them.

```
sqlplus user_name/password@connect_string @create_schema.sql data_tablespace  
index_tablespace
```

```
sqlplus user_name/password@connect_string @init.sql data_tablespace index_tablespace
```

Configure the database connection

Follow these instructions to write the database connection string in the Platform Application Center configuration file `SPERF_TOP/conf/dat asource. xml` with encrypted passwords.

You have a user name, password, and URL to access the database.

1. If you connected to the UNIX host via telnet and are running xserver on a local host, set your display environment.

Test your display by running `xclock` or another X-Windows application.

If the application displays, your display environment is already set correctly; otherwise, you need to set your display environment.

- For `csh` or `tcsh`:

```
setenv DISPLAY hostname:0.0
```

- For `sh`, `ksh`, or `bash`:

```
DISPLAY=hostname:0.0
```

```
export DISPLAY
```

where *hostname* is your local host.

2. Launch the database configuration tool.

Run **`$PERF_TOP/1.2/bin/dbconfig.sh`**.

3. In the User ID and Password fields, specify the user account name and password with which to connect to the database.
4. In the JDBC driver field, select the driver for your database.
5. In the JDBC URL field, enter the URL for your database.

This should be similar to the format given in Example URL format.

6. In the Maximum connections field, specify the maximum allowed number of concurrent connections to the database server.
7. Click Test to test your database connection.
8. Click OK to save your settings.

Setting access controls

Platform Application Center has a flexible access control mechanism. You can control access to all pages of the web interface, and you can also control whether users can take actions on objects in pages.

About users and user groups

You do not create users or user groups in Platform Application Center.

Platform Application Center automatically loads users and user groups defined in LSF. This is controlled with the parameter `ENABLE_USERGROUP` in `/opt/pac/gui/conf/pmc.conf`.

You can also configure Platform Application Center to add other users and groups from existing authentication databases like NIS or LDAP. To limit the number of user groups that is loaded and improve performance, you need to specify which user groups you want to load in the configuration file `/opt/pac/gui/conf/osusergroup.conf` and restart the web server.

Users and groups are preloaded when you start Platform Application Center, and updated every 3600 seconds. You can configure this interval in the configuration file `/opt/pac/gui/conf/pmc.conf` with the parameter `ACL_SYNC_INTERVAL`.

About user roles

To give permissions to users and user groups, you define user roles. A user role is a common group of permissions.

You can assign more than one role to a user or user group. When multiple roles are assigned, the user or user group receives all the permissions defined for all the roles.

Built-in user roles

Platform Application Center has several predefined user roles with default permission settings.

You cannot remove permissions from built-in roles, assign or unassign users or user groups, or delete built-in roles.

You can, however, add permissions to the built-in roles.

User role	User/User group members	Summary of permissions
Normal user	<p>Any operating system user.</p> <p>All users are automatically assigned this role.</p>	<p>Jobs:</p> <ul style="list-style-type: none"> View all jobs submitted by all users. Control only jobs that the application user submitted. View and use job submission forms created by other users. <p>If Process Manager is installed:</p> <ul style="list-style-type: none"> View Job Flow definitions <p>If Application Licenses is installed:</p> <ul style="list-style-type: none"> View all licenses <p>If Reports are installed:</p> <ul style="list-style-type: none"> View and run all reports <p>If the Platform Analytics add-on is installed, a user with the Normal user role can:</p> <ul style="list-style-type: none"> View past reports Subscribe to a report, or unsubscribe from a report to receive emails when reports are updated
Cluster administrator	<p>All LSF administrators defined in the LSF <code>lsf.cluster.cluster_name</code> file.</p> <p>Users and user groups are automatically assigned based on the configuration of your <code>lsf.cluster.cluster_name</code> file.</p>	<p>Full permission on all Platform Application Center pages and objects (view, control, configure).</p> <p>If the Platform Analytics add-on is installed, a user with the Cluster administrator role can:</p> <ul style="list-style-type: none"> Configure permissions on who can view a workbook through Platform Application Center's access control Add emails to user properties for application users through Settings > Users and User Groups List.
Administrator of user group	<p>Displayed as the name of the user group.</p> <p>Includes all user group administrators defined in the LSF <code>lsb.users</code> file.</p> <p>Users are automatically assigned based on the configuration of your <code>lsb.users</code> file.</p>	<ul style="list-style-type: none"> Same permissions as Normal user View, control, configure all jobs submitted by users who are members of the user group.
Flow administrator	<p>All Process Manager administrators defined in the Process Manager configuration file <code>js.conf</code>.</p> <p>Users and user groups are automatically assigned based on the configuration of your <code>js.conf</code> file.</p>	<ul style="list-style-type: none"> Same permissions as Normal user View, control, configure all job definitions and job flows.

User role	User/User group members	Summary of permissions
Flow control administrator	<p>All Process Manager control administrators defined in the the Process Manager configuration file <code>js.conf</code>.</p> <p>Users and user groups are automatically assigned based on the configuration of your <code>js.conf</code> file.</p>	<ul style="list-style-type: none"> • Same permissions as Normal user • View, control, configure all job flows.
Report administrator	<p>Used when the Platform Analytics add-on is installed.</p> <p>Platform Application Center automatically loads Tableau Licensed users as Report administrator.</p>	<p>A user with the Report administrator role can:</p> <ul style="list-style-type: none"> • Display the report list and view a report • View past reports • Manage report scheduling • Subscribe to a report, or unsubscribe from a report to receive emails when reports are updated • Add extra email addresses for sending reports

Custom user roles

You can create your own user roles and assign users and user groups, and permissions to objects in Platform Application Center as desired.

By default, a new user role has the same permissions as the Normal user role.

About assigning permissions

You can assign permissions to most objects in Platform Application Center.

The following are the possible permissions that you can assign, and the objects that you can assign these permissions to. You can assign a combination of all these permissions to an object.

The tables below provide details on each permission category and actions a user can take based on the permission category.

Application template permissions

Object	View	Control	Configure	None
Application templates navigation tree	<ul style="list-style-type: none"> • View application templates in navigation tree 	N/A	N/A	Not visible in navigation tree.

Object	View	Control	Configure	None
All application templates All built-in job submission forms	<ul style="list-style-type: none"> View all application templates that exist No actions on submission forms View template details through web services 	In submission form: <ul style="list-style-type: none"> Submit Save As Revert Can submit a form through web services	<ul style="list-style-type: none"> View all application templates on the list Take all actions on application templates: <ul style="list-style-type: none"> Publish Unpublish Save Modify 	All templates are not visible to the user.
Specific application templates Specific job submission forms	Same view as All application templates and submission forms with the exception that user can only view specific application templates	Same control actions as All application templates with the exception that can only perform actions on specific application templates	Same actions as All application templates with the exception that actions can only be taken on specific application templates	Only specific template visible to the user.

Custom pages permissions

Applies to	View	Control	Configure	None
Individual custom pages (pages created outside of web interface)	Custom page is visible to user.	User can interact with custom page and click buttons.	N/A (custom pages configured outside of web interface)	Custom page is not visible to user.

Hosts permissions

Applies to	View	Control	Configure	None
All hosts	<ul style="list-style-type: none"> View Hosts in the navigation tree View Hosts list table; no actions available on hosts Can sort and filter hosts 	<ul style="list-style-type: none"> Open hosts Close hosts 	N/A	Hosts not visible to the user.

Jobs, Job Data, Remote Console permissions

Permissions are set for Jobs. These permissions also affect Job Data, and Remote Consoles.

Important:

Directories a user can see in **Job Data > Shared Directories**, and actions a user can take in those directories is not defined through Platform Application Center. Those permissions are defined through the user's file

permissions in the operating system on the host on which those files reside.

Applies to	View	Control	Configure	None
All jobs, job data, and remote consoles	<ul style="list-style-type: none"> View Jobs in the navigation tree View Job list table, job details, remote consoles from job list; no actions Sort and filter jobs View Job Data in the navigation tree View job directory table and file properties; no actions Sort and filter job data View Remote Consoles in navigation tree View Remote Consoles list table; no actions 	<p>All job control actions on all jobs in the system:</p> <ul style="list-style-type: none"> New Suspend Resume Kill Requeue View output <p>Control jobs through web services</p> <p>All data control actions:</p> <ul style="list-style-type: none"> View Download Copy to Move to Delete More actions <p>All remote console actions</p> <ul style="list-style-type: none"> Open Close 	N/A	Jobs, job data, and remote consoles not visible to user.
Jobs of individual application templates	Same views as All Jobs with the exception that only jobs for specified application templates are visible	Same control actions as All Jobs with the exception that can only control jobs that were submitted with the specific submission forms	N/A	Jobs submitted with the specific submission forms are not visible to user.
Jobs of users in a specific user group	Same views as All Jobs with the exception that only jobs submitted by users in the specified user group are visible to the user	Same control actions as All Jobs with the exception that can only control jobs, data, and consoles, for jobs that were submitted by users in the specific user group	N/A	Jobs submitted with the specific submission forms are not visible to user.

Applies to	View	Control	Configure	None
User's own jobs	Same view as All jobs with the exception that user can only view jobs that he submitted.	Same control actions as All jobs with the exception that can only control jobs, data, consoles for jobs that he submitted.	N/A	Jobs that the user submitted will not be visible to the user.

Job Flows permissions

Job flows require a license. You will not be able to see Job flows unless you have installed the Platform Process Manager license. Any permissions you specify will not take effect until there is a license installed.

Applies to	View	Control	Configure	None
All job flows	<ul style="list-style-type: none"> View Job Flows in the navigation tree View Job Flows page; no actions 	All job flow actions on all job flows in the system: <ul style="list-style-type: none"> Kill Suspend Resume Rerun Rerun with variable Set flow variables 	N/A. Job flows are configured outside Platform Application Center.	Job flows are not visible to the user.
User's own job flows	Same view as All job flows with the exception that user can only view job flows that he submitted.	Same control actions as All job flows with the exception that user can only control flows that he submitted.	N/A. Job flows are configured outside Platform Application Center.	Job flows are not visible to the user.

Job Flow Definitions permissions

Job flow definitions require a license. You will not be able to see Job flow definitions unless you have installed the Platform Process Manager license. Any permissions you specify will not take effect until there is a license installed.

Applies to	View	Control	Configure	None
All job flow definitions	<ul style="list-style-type: none"> View Job Flows in the navigation tree View Job Flows page; no actions on job flow definitions 	All job flow definitions actions on all job flows in the system: <ul style="list-style-type: none"> Trigger Trigger with variable Hold Release Publish Unpublish Remove 	N/A. Job flow definitions are configured outside Platform Application Center.	Job flows and job flow definitions are not visible to the user.
User's own job flow definitions	Same view as All job flow definitions with the exception that user can only view job flows that he submitted.	Same control actions as All job flow definitions with the exception that user can only control flows that he submitted.	N/A. Job flow definitions are configured outside Platform Application Center.	Job flows and job flow definitions not visible to the user.

License Management permissions

License Management requires that Platform License Scheduler be installed.

Platform License Scheduler controls the software license sharing in your organization. Platform License Scheduler works with FlexNet™ products to control and monitor license usage.

Applies to	View	Control	Configure	None
Platform License Scheduler	<ul style="list-style-type: none"> View License Management on the navigation tree View License Usage pages Cannot kill jobs in the By Jobs page 	Kill jobs in the By Jobs page	N/A	License Management is not visible to the user.

Reports permissions

Reports require a Platform Application Center license. You will not be able to see Reports unless you have installed the license. Any permissions you specify will not take effect until there is a license installed.

Applies to	View	Control	Configure	None
Reports	<ul style="list-style-type: none"> View Reports on the navigation tree View report details Cannot generate or copy reports 	Generate reports	<ul style="list-style-type: none"> Copy standard reports View Custom Reports tab and reports list Edit custom reports 	Reports are not visible to the user.

System settings permissions

The System settings permission affects Logos and Colors, Page Settings, the Workload Dashboard under Resources, and Security. Permissions are assigned for all these areas with one control.

Applies to	View	Control	Configure	None
Logos and Colors	<ul style="list-style-type: none"> View Settings, Logos and Colors in navigation tree 	N/A	<ul style="list-style-type: none"> Reset changed values to saved settings Apply Revert 	Settings is not displayed in the navigation tree
Page Settings	<ul style="list-style-type: none"> View Settings, Page Settings in navigation tree 	N/A	Change all values in Page Settings.	Settings is not displayed in the navigation tree
Workload Dashboard	View Resources > Dashboard in navigation tree	N/A	N/A	Settings is not displayed in the navigation tree
Security (user roles and permissions)	<ul style="list-style-type: none"> View Settings, Users in navigation tree No actions 		<ul style="list-style-type: none"> Assign roles to users and user groups Add, remove, edit roles Define permissions for user roles. 	Settings is not displayed in the navigation tree

Configure Single URL for Failover

Platform Application Center Failover

Platform Application Center failover is configured by the `FAILOVER_HOST` parameter in `pacinstall.sh` during installation. Failover configuration cannot be changed after installation. For installation details, see *Installing Platform Application Center*.

Default Platform Application Center URL

By default, the URL you use to access Platform Application Center uses the Platform Application Center host name in the URL. The format is:

```
http://pac_host:8080/
```

If the Platform Application Center host fails over, it runs on a different host, and your usual URL will not work. You need to try again, and use the new host name in the URL.

Note:

To find out the name of the new Platform Application Center host after failover, run the following command:

```
egosh service list -s WEBGUI
```

Accessing Platform Application Center with a single URL

Important:

cluster failover and Platform Application Center host failover must be configured during the installation to use this feature. Configure failover during Platform Application Center installation by specifying host and failover hosts in `FAILOVER_HOST` parameter in the file `pacinstall.sh`. For details, see *Installing Platform Application Center*.

If you configured Platform Application Center failover during installation, you can change your system to allow a single URL for accessing Platform Application Center. If the Platform Application Center host fails over, the URL does not change.

To enable this feature, you must change the DNS manually, either by changing the corporate DNS or by modifying the DNS on the host where you open the browser to access Platform Application Center.

Enable a single URL by modifying the corporate DNS

Note:

If you modify the corporate DNS, the Platform Application Center URL is:

```
http://webgui.service.ego:8080/
```

1. Start the cluster (this starts EGO and the WEBGUI service).
2. Configure the corporate DNS (Corp DNS):

Add the IP addresses of all Platform Application Center hosts to Corp DNS. For each Platform Application Center host, add a new `egonameserver` entry and specify the host IP address. For example:

```
Service NS egonameserver
egonameserver A 172.17.1.70
egonameserver A 172.17.1.71
```

3. Configure the file: `SEGO_TOP/eservice/esd/conf/esdefault.xml`.

Edit the following parameters:

ESD_EGO_DOMAIN

Change `ego` to `service`, for example:

```
<ESD_EGO_DOMAIN>service</ESD_EGO_DOMAIN>
```

ESD_CORP_DOMAIN

Remove the comment markings for this line, and change `@EGO_SD_CORPDOMAIN@` to `ego`

ESD_CORP_KEY

Replace `@EGO_SD_CORPKEYNAME@` with the TSIG key for Corp DNS. If no key is available, do not change this line.

ESD_EGO_KEY

Replace `ESD_EGO_KEY name = "ego. "` with `"service. ego"`. If you do not need this key, comment out this line.

For `ESD_EGO_KEY`, run the `dnssec-keygen` command to generate a key. For example:

```
dnssec-keygen -a HMAC-MD5 -b 128 -n HOST -k service.ego
```

The key you generate is stored under: `ego/1.2/linux2.4-glibc2.3-x86/bin`.

Here is an example of the file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ESDDefaultPluginConfiguration>
  <!-- EGO DNS server name -->
  <!-- ESD_EGO_NAMESERVER>@EGO_SD_NAMESERVER@</ESD_EGO_NAMESERVER -->
  <ESD_EGO_NAMESERVER>egonameserver</ESD_EGO_NAMESERVER>
  <!-- EGO DNS domain name -->
  <!-- ESD_EGO_DOMAIN>@EGO_SD_EGODOMAIN@</ESD_EGO_DOMAIN -->
  <ESD_EGO_DOMAIN>service</ESD_EGO_DOMAIN>
  <!-- Corporation DNS domain name -->
  <ESD_CORP_DOMAIN>ego</ESD_CORP_DOMAIN>
  <!-- EGO DNS sub-domain TSIG key created by dnssec-keygen -->
  <!-- ESD_EGO_KEY name="@EGO_SD_EGOKEYNAME@">@EGO_SD_EGOKEY@</ESD_EGO_KEY -->
  <ESD_EGO_KEY name="service. ego.">rUIWkhrNFCsXkOwZBu/xVA==</ESD_EGO_KEY>
  <!-- CORP DNS domain TSIG key created by dnssec-keygen -->
  <ESD_CORP_KEY name="ego.">e1rBv20yZFh0sUMyL76wqQ==</ESD_CORP_KEY>
</ESDDefaultPluginConfiguration>
```

4. Configure the file: `EGO_TOP/eservice/esd/conf/named/conf/named.conf`.

Edit the following parameters:

key ego.

change `ego.` to `service. ego.`

zone "ego." IN

change ego. to service. ego.

file "db.ego" (under zone "ego." IN)

change db. ego to db. service. ego.

key

change ego. to service. ego.

Here is an example of the file:

```

...
key service. ego. {
    algorithm HMAC-MD5. SIG-ALG. REG. INT;
    secret "rUlWkhrNFCsXkOwZBu/xVA==";
};
...
zone "service. ego." IN {
    type master;
    file "db. service. ego";
    allow-update { key service. ego. ; };
};
...

```

5. Rename the template file: EGO_TOP/eservice/esd/conf/named/namedb/TMPL. db. EGODOMAIN. CORPDOMAIN to db. service. ego.
6. Configure the file: db. service. ego.

Edit the following parameters:

@EGO_SD_EGODOMAIN@.@EGO_SD_CORPDOMAIN@.

Change to service. ego.

@EGO_SD_NAMESERVER@.@EGO_SD_CORPDOMAIN@.

Change to egonameserver. ego.

root.@EGO_SD_EGODOMAIN@.@EGO_SD_CORPDOMAIN@.

Set to root. service. ego.

NS @EGO_SD_NAMESERVER@.@EGO_SD_CORPDOMAIN@.

Change to egonameserver. ego.

NS @EGO_SD_NAMESERVER@.@EGO_SD_EGODOMAIN@.@EGO_SD_CORPDOMAIN@.

Change to egonameserver. service. ego.

Here is an example of the file:

```

$ORIGIN .
$TTL 0 ; 0 seconds
service. ego          IN SOA  egonameserver. ego. root. service. ego. (
                                84           ; serial
                                10800        ; refresh (3 hours)
                                900          ; retry (15 minutes)
                                604800       ; expire (1 week)
                                0            ; minimum (0 seconds)
                                )
NS      egonameserver. ego.
NS      egonameserver. service. ego.

```

7. Start the EGO Service Director.

Run this command:

```
egosh service start ServiceDirector
```

Enable a single URL by modifying your browser host

Note:

If you modify the browser host, the Platform Application Center URL is:

```
http://webgui.ego:8080/
```

1. Start the cluster.

This starts EGO and the WEBGUI service.

2. Start the EGO Service Director.

Run this command:

```
egosh service start ServiceDirector
```

3. Configure your browser host.

- On Linux, Service Director can run on any management host, so you must configure your host to use any management host as DNS.

Add the IP addresses of the Platform Application Center hosts to the `/etc/resolv.conf` file. For each host, add a new `nameserver` entry and specify the host IP address. For example:

```
nameserver 172.17.1.70
```

```
nameserver 172.17.1.71
```

- On Windows, name server entries can be changed interactively, as follows:
 1. From the Windows Start menu, click Control Panel, and select Network Connections.
 2. Right-click Local Area Connection, and then select Properties.
 3. Select Internet Protocol (TCP/IP) from the list of protocols, and then click Properties.
 4. On the General page, click Advanced.
 5. Click the DNS tab.
 6. Add DNS server addresses according to the Windows instructions.

You must repeat this step on any host that you will use to access Platform Application Center.

Enabling Remote Consoles

The remote console feature is disabled by default.

Enable remote consoles in the cluster

Download JDK 5.0 or later to use the required version of `keytool` for generating the `tightvnc` key and `jarsigner` for signing the key.

The remote console feature in Platform Application Center depends on a third-party tool: `tightvnc`. To enable this feature, follow these steps.

1. Download the package:

http://www.tightvnc.com/download/1.3.10/tightvnc-1.3.10_javabin.tar.gz

2. Unzip it.

```
tar -zxvf tightvnc-1.3.10_javabin.tar.gz
```

3. Generate the key.

This step creates the keystore (unless you specify an existing one) and generates the private key with an expiry period of 3650 days:

```
keytool -genkey -alias keystore_alias -keystore keystore_name -keyalg rsa -storetype PKCS12 -validity 3650
```

For example:

```
keytool -genkey -alias vnc -keystore platform.jks -keyalg rsa -storetype PKCS12 -validity 3650
```

When prompted, enter passwords for the keystore and the private key you are generating.

Important:

Use the `keytool` tool that comes with JDK 5.0, because the `jarsigner` tool that comes with Linux may not be up to date.

4. Sign the jar file.

This step extracts the certificate from the keystore and attaches it to the generated signature of the signed JAR file:

```
jarsigner -verbose -keystore keystore_name -storetype PKCS12 -signedjar sVncViewer.jar VncViewer.jar keystore_alias
```

For example:

```
jarsigner -verbose -keystore platform.jks -storetype PKCS12 -signedjar sVncViewer.jar VncViewer.jar vnc
```

Enter your passwords when prompted.

Important:

Use the `jarsigner` tool that comes with JDK 5.0, because the `jarsigner` tool that comes with Linux may not be up to date.

5. Copy the signed jar file to Platform Application Center:

```
cp classes/sVncViewer.jar PAC_TOP/gui/3.0/tomcat/webapps/platform/pac/vnc/lib
```

6. Rename the signed jar file:

```
mv sVncViewer.jar VncViewer.jar
```

7. Restart Platform Application Center.

- If EGO is disabled, run `pmcadmin stop` then `pmcadmin start` from the command line.
- If EGO is enabled, run `egosh service stop WEBGUI` then `egosh service start WEBGUI` from the command line.

Enable remote consoles in an application

The FLUENT application has remote consoles enabled by default.

You can enable remote consoles for other applications by editing the submission script. Make sure you have enabled the feature in the cluster also.

Example

This example is for the FLUENT application and shows the code that checks the VNC support flag and sets the VNC environment. Use it as an example to modify your own application. For the lines starting with "JOB_RESULT=", use the command and parameters of your own application. Put the following code before the line

```
export JOB_RESULT OUTPUT_FILE_LOCATION PAC_TOP/application/job-result.sh
```

where *PAC_TOP* is the top-level Platform Application Center installation directory (for example, `/opt/pac`).

Note:

Note that `VNCSession` must be consistent with the parameter `VNCSession` configured in `PAC_TOP/gui/conf/pmc.conf`.

```
GRAPHIC_OPT="-g"
if [ "$CONSOLE_SUPPORT"="Yes" -a "$VNCSession"="User" ]; then          VNC_SID=`$
{GUI_CONFDIR}/application/vnc/startvnc.sh "${OUTPUT_FILE_LOCATION}" "${EXECUTIONUSER}" $
{VNC_WIDTH} ${VNC_HEIGHT} `
DISPLAY=${HOSTNAME}:${VNC_SID}.0"
GRAPHIC_OPT=""
export DISPLAY VNC_SID
fi

if [ "$CONSOLE_SUPPORT"="Yes" -a "$VNCSession"="Job" ]; then
# copy VNC starter to job data directory
mkdir -p "${OUTPUT_FILE_LOCATION}/.spooler_action"
cp -fp ${GUI_CONFDIR}/application/vnc/startvnc_prejob.sh ${GUI_CONFDIR}/
application/vnc/stopvnc_prejob.sh "${OUTPUT_FILE_LOCATION}/.spooler_action"
cp -fp ${GUI_CONFDIR}/application/vnc/stopvnc_prejob.sh "${
OUTPUT_FILE_LOCATION}/.spooler_action"
cp -fp ${GUI_CONFDIR}/application/vnc/startVncSession_prejob.sh "$
{OUTPUT_FILE_LOCATION}/.spooler_action"
cp -fp ${GUI_CONFDIR}/application/vnc/xstartup.template "$
{OUTPUT_FILE_LOCATION}/.spooler_action/xstartup.template"
export VNC_SID="-1"
GRAPHIC_OPT=""
JOB_RESULT=`/bin/sh -c "bsub -B -N ${JOB_NAME_OPT} ${CWD_OPT} ${SUB_QUEUE_OPT} $
{RUNHOST_OPT} ${NCPU_OPT} ${LSF_RESREQ} ${OUTPUT_FILE_LOCATION_OPT} $
{EXTRA_PARAMS} .spooler_action/startvnc_prejob.sh ${FLUENT_CMD} ${VERSION} $
{RELEASE_OPT} ${GRAPHIC_OPT} ${FLUENT_JOURNAL_OPT} ${FLUENT_NCPU_OPT} ${LSF_OPT} $
{SSH_OPT} ${HOSTTYPE_OPT} ${OTHER_OPTS} 2>&1 "`
else
JOB_RESULT=`/bin/sh -c "bsub -B -N ${JOB_NAME_OPT} ${CWD_OPT} ${SUB_QUEUE_OPT} $
{RUNHOST_OPT} ${NCPU_OPT} ${LSF_RESREQ} ${OUTPUT_FILE_LOCATION_OPT} ${EXTRA_PARAMS} $
{FLUENT_CMD} ${VERSION} ${RELEASE_OPT} ${GRAPHIC_OPT} ${FLUENT_JOURNAL_OPT} $
{FLUENT_NCPU_OPT}
${LSF_OPT} ${SSH_OPT} ${HOSTTYPE_OPT} ${OTHER_OPTS} 2>&1 "`
```

```
fi  
export JOB_RESULT OUTPUT_FILE_LOCATION  
S{GUI_CONFDIR}/application/job-result.sh
```

Job data purging

Summary

Job data includes files in the job directory and the related information in the internal database. To avoid data overflowing the file system and database, job data must be cleaned up after a job is completed (Done or Exited). Platform Application Center supports automatic job data purging.

Platform Application Center runs the purger task periodically, according to a configurable parameter: `AutoPurgeTime`. The purger task checks all job data to see whether it is expired. If it is expired, the purger removes the job data.

```
job data expiry time = job completion time + time to live (ttl)
```

Time to live (`ttl`) is configurable per application, the default is 90 days.

Only an administrator is allowed to modify the two parameters for job data purging. Choose Job Data > By Job to view the scheduled autopurger time displayed in the job directory list.

Configure AutoPurgerTime

1. Log on to the Platform Application Center server host as root.
2. Edit the file:

```
PAC_TOP/gui/conf/pmc.conf
```

3. Set the `AutoPurgeTime` parameter (specify the number of hours):

```
AutoPurgeTime=3
```

4. Restart Platform Application Center.
 - If EGO is disabled, run `pmcadm n stop` then `pmcadm n start` from the command line.
 - If EGO is enabled, run `egosh service stop WEBGUI` then `egosh service start WEBGUI` from the command line.

Configure ttl (time to live) for an application

1. Log on to the Platform Application Center server host as root.
2. Find the application definition file.

If the application is published:

```
PAC_TOP/gui/conf/application/published/application_name/
application_name.xml
```

If the application is unpublished:

```
PAC_TOP/gui/conf/application/draft/application_name/application_name.xml
```

3. Edit the file, find and modify the `ttl` attribute in all XML elements `<application>`.

For example:

```
<repository ttl="90">
```

In this example, the application job data will be kept for 90 days after the job completes.

4. Restart Platform Application Center.
 - If EGO is disabled, run `pmcadm n stop` then `pmcadm n start` from the command line.

Configure and Manage Platform Application Center

- If EGO is enabled, run `egosh service stop WEBGUI` then `egosh service start WEBGUI` from the command line.

Adding custom job actions

Custom job actions

If you want to make a new job action available to users in Platform Application Center, you may choose an existing command or create a custom job action script. The script or command will be executed whenever a user chooses the new job action.

The job action script should follow standard conventions. For example:

- Return code of 0 for success, and non-zero for error
- Use `stderr` output for error messages, and `stdout` output for other messages (errors and messages from the job action script display on the page in Platform Application Center as usual)

When Platform Application Center invokes the job action script or command, it passes the job IDs as arguments. For example, if the job ID is 1234 and the job action is `bswitch priority`:

```
<Command type="CLI">bswitch priority</Command>
```

then Platform Application Center invokes the following command:

```
bswitch priority 1234
```

To reduce the number of buttons on the page in Platform Application Center, you should group related job actions. You will see a cascading button for each group of job actions.

You can control the availability of each job action based on role, application, job state, or number of jobs selected.

Configure a custom job action

To add a new job action, edit the file:

```
PAC_TOP/gui/conf/jobaction.xml
```

You must be an administrator to edit this file. The following tags and attributes are supported:

Job Action Group

- `id`

The group ID must be unique. All job actions in a group are accessible through one cascading button.

- If the group ID is null, the cascading button will not be shown.
- If multiple groups have same ID, only the first one in the file is available in Platform Application Center.

- `name`

This text is displayed on the button in Platform Application Center.

Job Action

- `id`

The action ID must be unique.

- If the action ID is null, the action will not be shown.
- If multiple actions have the same ID, only the first one in the file is available in Platform Application Center.

Command	<p>the path of the job action script or command</p> <ul style="list-style-type: none"> • type (optional) By default, it is CLI. At this time, CLI is the only command type supported. • target (optional) Define the target windows when the command is running.
Role (optional)	<p>By default, it is:</p> <p>1: 0</p> <ul style="list-style-type: none"> • 1 makes the job action available to the administrator. • 0 makes the job action available to users. If the action is not available to you, you cannot see it in Platform Application Center.
EnableStatus (optional)	<p>By default, it is ALL.</p> <p>Specify a list of job states separated by semi-colon (;). The job action is only available to jobs in these states. The keyword ALL means the job action is available to all jobs.</p>
MultipleJobs (optional)	<p>By default, it is true.</p> <p>true means the job action is available when multiple jobs are selected in job list page.</p>
Application (optional)	<p>By default, it is ALL.</p> <p>Specify a list of applications separated by comma (.). The job action is only available to these applications. The keyword ALL means the job action is available to all applications.</p>
EnableIcon	The path to the icon file for the enabled state. The icon appears on the job detail page.
DisableIcon	The path to the icon file for the disabled state. The icon appears on the job detail page.

Example

This file contains a custom job action to raise the priority of a job. An administrator can select pending ABAQUS or CFX jobs in Platform Application Center and then choose Other>Raise Priority.

```
<?xml version="1.0" encoding="UTF-8"?>
<JobActionConf>
<JobActionGroup id="userAction" name="Other">
<JobAction id="raisepriority">
  <Command type="CLI">bswitch priority</Command>
  <DisplayName>Raise Priority</DisplayName>
  <Role>1:0</Role>
  <EnabledStatus>PENDING</EnabledStatus>
  <MultipleJobs>true</MultipleJobs>
  <Application>ABAQUS, CFX</Application>
  <EnableIcon>switchqueue/image/enable.icon</EnableIcon>
  <DisableIcon>switchqueue/image/disable.icon</DisableIcon>
</JobAction>
</JobActionGroup>
</JobActionConf>
```

Platform Application Center log files and log levels

Use the Platform Application Center log file to troubleshoot errors that occur in the graphical user interface. Errors from add-ons such as Platform Process Manager, Platform Analytics, and Platform License Scheduler are also written to the log file.

Log file location

- `/pac/gui/logs/catalina.out`

Logging configuration file

Properties file location

The location of the logging configuration properties file is:

- `$GUI_CONFDIR/log4j.properties`

File format customization

The format of the log-file entries can be changed. For more details, see the `log4cxx` documentation under "Configuration" at <http://logging.apache.org/log4cxx/>.

Log levels

The default log level is INFO.

Level	Description
DEBUG	<p>Logs all debug-level messages.</p> <p>At this log level, all DEBUG, INFO, WARN, ERROR, and FATAL messages are logged.</p>
INFO	<p>Logs all informational messages about events that occurred.</p> <p>For example, Platform Application Center successfully started.</p> <p>At this log level, all INFO, WARN, ERROR, and FATAL messages are logged.</p>
WARN	<p>Logs only those messages that are potentially harmful. The system can tolerate this error. Features and performance are not affected.</p> <p>At this log level, all WARN, ERROR, and FATAL messages are logged.</p>
ERROR	<p>Logs only those messages that indicate error conditions. Includes system errors and user errors. These error conditions might allow the application to continue running. Features and performance may be partially affected.</p> <p>For example, cannot create a process to execute a job, failed to create directories because of disk quota, etc.</p> <p>At this log level, all ERROR and FATAL messages are logged.</p>

Level	Description
FATAL	Logs only those messages in which the system is unusable. The whole system cannot work normally. For example, the database connection failed. At this log level, all FATAL messages are logged.

Platform Application Center Reporting

Platform Application Center reporting allows you to look at the overall statistics of your entire cluster. You can analyze the history of hosts, resources, and workload in your cluster to get an overall picture of your cluster's performance.

Overview of reporting

An efficient cluster maximizes the usage of resources while minimizing the average wait time of a workload. To ensure your cluster is running efficiently at all times, you can analyze the activity within your cluster to find areas for improvement.

The reporting feature collects data from the cluster and maintains this data in a relational database system. Cluster data is extracted from the database and displayed in reports either graphically or in tables. You can use these reports to analyze and improve the performance of your cluster, to perform capacity planning, and for troubleshooting.

The reporting feature depends on the Platform Enterprise Reporting Framework (PERF) architecture. This architecture defines the communication between your cluster, relational database, and data sources.

Platform Application Center collects various types of data, which can be reported using the standard, out-of-the box reports. In addition, Platform Application Center can be configured to collect customer-specific data, which can be reported using custom reports.

Introduction to reporting

An efficient cluster maximizes the usage of resources while minimizing the average wait time of workload. To ensure that your cluster is running efficiently at all times, you need to analyze the activity within your cluster to see if there are any areas for improvement.

The reporting feature uses the data loader controller service, the job data transformer service, and the data purger service to collect data from the cluster, and to maintain this data in a relational database system. The reporting feature collects the cluster data from a relational database system and displays it in reports either graphically or in tables. You can use these reports to analyze and improve the performance of your cluster, and to troubleshoot configuration problems.

You can access the reporting feature from the Platform Application Center.

Standard and custom reports

Platform has provided a set of standard reports to allow you to immediately analyze your cluster without having to create any new reports. These standard reports provide the most common and useful data to analyze your cluster.

You can also create custom reports to perform advanced queries and reports beyond the data produced in the standard reports.

The database

The reporting feature includes the MySQL 5.x database, a small-footprint, open source database. If you want to use the reporting feature to produce regular reports for a production cluster, you can also use a supported commercial database. The reporting feature supports Oracle 9i and Oracle 10g databases.

Important:

The Apache Derby database is no longer supported in Platform Application Center.

Standard reports

For your convenience, Platform has provided several standard reports for you to use. These reports allow you to keep track of some useful statistics in your cluster.

Standard reports overview

Standard reports are based on raw data stored in the relational database, and do not perform any data aggregation or calculations.

The following is a list of the standard reports that are included with the reporting feature. For further details on a report, open its full description.

Table 1: Standard reports

Name	Description	Category
Active Job States Statistics by Queue	Number of active jobs in each active job state in a selected queue.	LSF
Cluster Availability - LSF	LSF host availability in an LSF cluster.	LSF
Cluster Job Hourly Throughput	Number of submitted, exited, and done jobs in a cluster.	LSF
Cluster Job Slot Utilization	Job slot utilization levels in your cluster.	LSF
Host Resource Usage	Resource usage trends for selected hosts.	LSF
Job Slot Usage by Application Tag	Job slots used by applications as indicated by the application tag.	LSF
Jobs Forwarded to Other Clusters	The number of jobs forwarded from your cluster to other clusters. You can only produce this report if you use LSF MultiCluster.	LSF MultiCluster
Jobs Received from Other Clusters	The number of jobs forwarded to your cluster from other clusters. You can only produce this report if you use LSF MultiCluster.	LSF MultiCluster
License Usage	The license usage under License Scheduler. You can only produce this report if you use LSF License Scheduler.	LSF License Scheduler
Performance Metrics	Internal performance metrics trend for a cluster. You can only produce this report if you enabled performance metric collection in your cluster (badmin perfmon start)	LSF
Service Level Agreement (SLA)	Job statistics by job state over time, compared with SLA goals.	LSF

View the full description of a report

1. In the Console, navigate to Reports, then Standard Reports.
2. Click the name of your report to open it.
3. Click Report properties.

Produce a standard report

The reports stored in the system do not include actual data. Instead, the reports define what data to extract from the system, and how to display it graphically.

Reports need to be produced before you can see the data. When you produce a report, you query the database and extract specific data. The amount of system overhead depends on how much data is in the report.

Standard reports have configurable parameters so you can modify the report and get exactly the data that you want.

1. In the Console, navigate to Reports, then Standard Reports.
2. Click the name of your report to open it.
3. Set the report parameters as desired. Default settings are shown, but you can modify them to suit your needs.
4. Click Produce Report.

After a short time, the resulting data is displayed graphically.

When you close the report window, you lose the contents of the report unless you export it first.

Export report data

Data expires from the database periodically, so producing a report at a later date may return different data, or return no output at all. After you produce a report, you can keep your results by exporting the report data as comma-separated values in a CSV file. In this way you can preserve your data outside the system and integrate it with external programs, such as a spreadsheet. You can also keep your graphical results by using your browser to save the report results as an image.

Once you produce a report, exporting is the best way to save the data for future use. You cannot produce the same report at a later date if the data has expired from the database.

1. In the Console, produce and view your report.
2. Click Export Report Data.
3. In the browser dialog, specify the output path and name the exported file.

In the Save as type field, specify "CSV".

Custom reports

You can create and use custom reports if the standard reports are insufficient for your needs.

What are custom reports?

While standard reports are provided for your use by Platform, custom reports are reports you create as needed to satisfy specific reporting needs at your site.

Custom reports let you define combinations of data that are not available in the standard reports. Custom report output is always displayed in tabular format.

What can I do with custom reports?

Create reports

The easiest way to create a custom report is to copy an existing report, then customize the SQL query string as desired. To customize the SQL query string, you may need to refer to the data schema, which describes the organization of information in the relational database. The data schema for each standard report is available in the Console by opening the report and clicking Help.

Even if you cannot edit SQL, saving a report as a custom report lets you re-use the report data without having to re-input the parameters in the standard report.

- If the time period is fixed, you get the same data every time you produce the report, but the report will be empty when the data expires from the database.

- If the time period is relative, you can get data for a different time period each time you produce the report.

You can also define custom reports from a blank template and input the SQL query string directly.

When you create custom reports, you can enter a category and use it to group the reports any way you want.

Delete reports

Unlike standard reports, custom reports can be deleted. You might prefer to rename old reports (by modifying them) instead of deleting them.

Use reports

You produce custom reports and export the data in the same way as standard reports.

Data expires from the database periodically, so producing a report at a later date may return different data, or return no output at all. After you produce a report, you can keep your results by exporting the report data as comma-separated values in a CSV file. In this way you can preserve your data outside the system and integrate it with external programs, such as a spreadsheet. You can also keep your graphical results by using your browser to save the report results as an image.

If you ever want to modify parameters of a custom report, you must edit the SQL query string directly.

Create a custom report from an existing report

This method is convenient because you can extend an existing report. Examine your current standard and custom reports and select one with similar data sources or output to the new report that you want to create.

1. In the Console, select the report that you want to copy, with all the parameters configured as you wish to copy them.
2. Click Copy to New Custom Report.
3. Edit the report properties and query string as desired.
 - a) In the Report properties section, you should give the new report a unique name. You can also modify the report summary, description, and category.
 - b) In the Report query section, you can modify the SQL query directly.

To edit the SQL query, you will need to know about the data schema of the database. For further information on the data schema, refer to Platform Application Center Reports Data Schema in the Platform Application Center Knowledge Center.

- c) To validate your SQL query string and ensure that your report delivers the appropriate results, click Produce Report.

This will actually produce the report, so you might want to limit your testing to a small set of data.

You can continue to edit your SQL query string and test the results of your report until you are ready to save it.

4. To finish, click Create.

To access your new custom report, navigate to Reports then Custom Reports.

Create a new custom report

You must be able to construct valid query strings with Structured Query Language (SQL).

1. In the Console, navigate to Reports then Custom Reports.
2. Select Global Actions > Create Custom Report.
3. Define the report properties and query string as desired.
 - a) In the Report properties section, specify the report name, summary, description, and category.
 - b) In the Report query section, input your SQL query string.

For further information on the data schema, refer to Platform Application Center Reports Data Schema in the Platform Application Center Knowledge Center.

- c) To validate your SQL query string and ensure that your report delivers the appropriate results, click Produce Report.

This will actually produce the report, so you might want to limit your testing to a small set of data.

You can continue to edit your SQL query string and test the results of your report until you are ready to save it.

4. To finish, click Create.

To access your new custom report, navigate to Reports then Custom Reports.

Modify a custom report

1. In the Console, navigate to Reports then Custom Reports.
2. Click the name of your report.
3. Modify the report properties and query string as desired.
 - a) Edit the report properties and SQL query string.

For further information on the data schema, refer to Platform Application Center Reports Data Schema in the Platform Application Center Knowledge Center.

- b) To validate your SQL query string and ensure that your report delivers the appropriate results, click Produce Report.

This will actually produce the report, so you might want to limit your testing to a small set of data.

You can continue to edit your SQL query string and test the results of your report until you are ready to save it.

4. To confirm your changes, click Save.

Produce a custom report

1. In the Console, navigate to Reports then Custom Reports.
2. Click the name of your report to open it.
3. Click Produce Report.

After a short time, the resulting data is displayed in tabular format.

When you close the report window, you will lose the contents of the report unless you export it first.

Export report data

Once you produce a report, exporting is the best way to save the data for future use. You cannot produce the same report at a later date if the data has expired from the database.

1. In the Console, produce and view your report.
2. Click Export Report Data.
3. In the browser dialog, specify the output path and name the exported file.

In the Save as type field, specify "CSV".

Delete a custom report

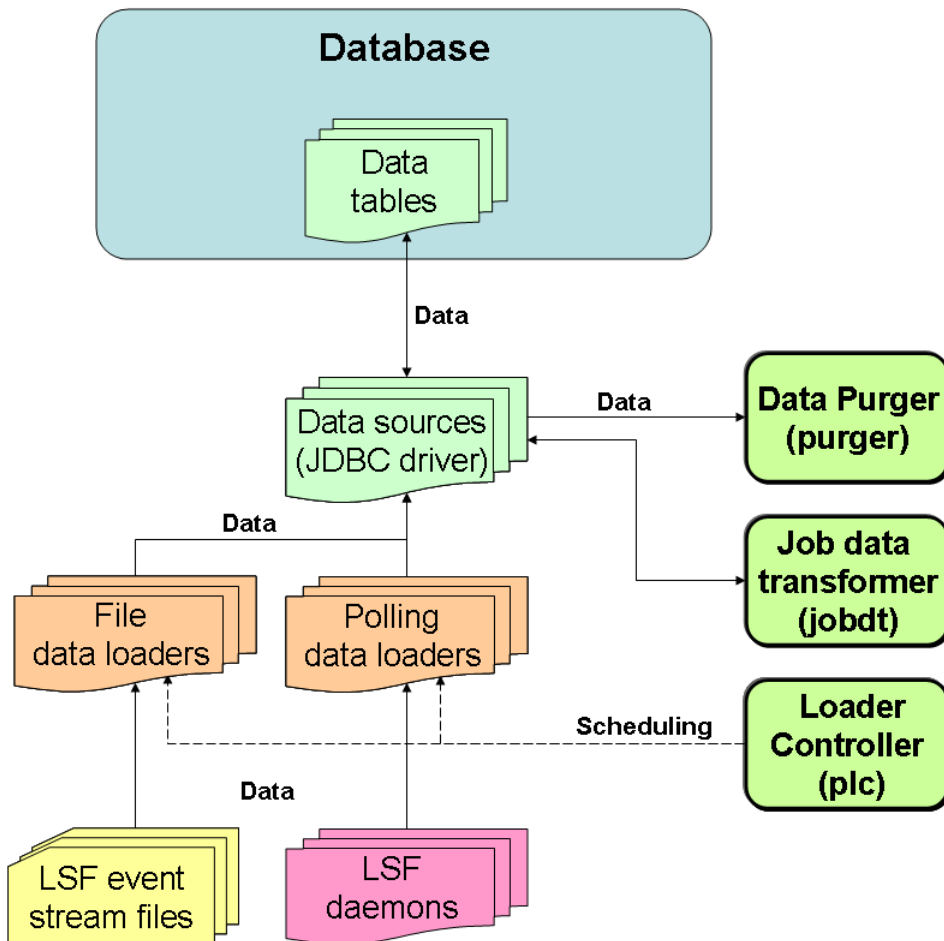
1. In the Console, navigate to Reports then Custom Reports.
2. Locate your report in the list.
3. Select Actions > Delete Report.

System description

The reporting feature is built on top of the Platform Enterprise Reporting Framework (PERF) architecture. This architecture defines the communication between your EGO cluster, relational database, and data sources via the PERF Loader Controller (PLC). The loader controller is the module that controls multiple loaders for data collection.

PERF architecture

The following diagram illustrates the PERF architecture as it relates to your cluster, reporting services, relational database, and data loaders.



Data loaders

The reporting feature collects cluster operation data using data loaders to load data into tables in a relational database. The data loaders connect to the database using a JDBC driver. The data loaders handle daylight savings automatically by using GMT time when collecting data.

Default data loaders

The following are lists of the data loaders and default behavior:

Table 2: LSF data loaders

Data loader name	Data type	Data gathering interval	Data loads to	Loader type
License Scheduler (bl dloader)	license usage	5 minutes	BLD_LICENSE_USAGE	polling
Host metrics (hostmetricsloader)	host-related metrics	5 minutes	RESOURCE_METRICS RESOURCES_RESOURCE_METRICS	polling
Host properties (hostpropertiesloader)	resource properties	1 hour	LSF_RESOURCE_PROPERTIES	polling
Bhosts (lsfbhostloader)	host utilization and state-related	5 minutes	LSF_BHOSTS	polling
LSF events (lsfeventloader)	events with a job ID, performance events, resource events	10 seconds	LSB_EVENTS LSB_EVENTS_EXECHOSTLIST LSF_PERFORMANCE_METRIC	
Resource properties (lsfresproploader)	shared resource properties	1 hour	LSF_RESOURCE_PROPERTIES	polling
SLA (lsfslaloader)	SLA performance	5 minutes	LSF_SLA	polling
Shared resource usage (sharedresusageloader)	shared resource usage	5 minutes	SHARED_RESOURCE_USAGE SHARED_RESOURCE_USAGE_HOSTLIST	polling

Table 3: EGO data loaders

Data loader name	Data type	Data gathering interval	Data loads to	Loader type
Consumer resource (egoconsumerresloader)	resource allocation	5 minutes	CONSUMER_DEMAND CONSUMER_RESOURCE_ALLOCATION CONSUMER_RESOURCELIST	polling

Data loader name	Data type	Data gathering interval	Data loads to	Loader type
Dynamic metric (egodynamimetricloader)	host-related dynamic metric	5 minutes	RESOURCE_METRICS RESOURCES_RESOURCE_METRICS	polling
EGO allocation events (egoeventsloder)	resource allocation	5 minutes	ALLOCATION_EVENT	file
Static attribute (egostaticloader)	host-related static attribute	1 hour	ATTRIBUTES_RESOURCE_METRICS RESOURCE_ATTRIBUTES	polling

System services

The reporting feature uses system services. If your cluster has PERF controlled by EGO, these services are run as EGO services. Each service uses one slot on a management host.

Loader controller

The loader controller service (plc) controls the data loaders that collect data from the system and writes the data into the database.

Data purger

The data purger service (purger) maintains the size of the database by purging old records from the database. By default, the data purger purges all data that is older than 14 days, and purges data every day at 12:30am.

Job data transformer

The job data transformer service (jobdt) converts raw job data in the relational database into a format usable by the reporting feature. By default, the job data transformer converts job data every hour at thirty minutes past the hour (that is, at 12:30am, 1:30am, and so on throughout the day).

Reports administration

Reports directories

The reporting feature resides in various `perf` subdirectories within the Platform Application Center directory structure. This document uses `PAC_TOP` to refer to the top-level Platform Application Center installation directory, and `LSF_TOP` to refer to the top-level LSF installation directory. The reporting feature directories include the following:

Directory name	Directory description	Default file path
<code>\$PERF_TOP</code>	Reports framework directory	<code>PAC_TOP/perf</code>
<code>\$PERF_CONFDIR</code>	Configuration files	<code>PAC_TOP/perf/conf/</code>
<code>\$PERF_LOGDIR</code>	Log files	<code>PAC_TOP/perf/log</code>
<code>\$PERF_WORKDIR</code>	Working directory	<code>PAC_TOP/perf/work</code>
<code>\$PERF_DATADIR</code>	Data directory	<code>PAC_TOP/perf/data</code>

Reporting services

The reporting feature uses the following services.

- Job data transformer (`j obdt`)
- Loader controller (`pl c`)
- Data purger (`purger`)

If your cluster has PERF controlled by EGO, these services are run as EGO services.

You need to stop and restart a service after editing its configuration files. If you are disabling the reporting feature, you need to disable automatic startup of these services.

Log files for these services are available in the `PERF_LOGDIR` directory. There are seven logging levels that determine the detail of messages recorded in the log files. In decreasing level of detail, these are ALL (all messages), TRACE, DEBUG, INFO, WARN, ERROR, FATAL, and OFF (no messages). By default, all service log files log messages of INFO level or higher (that is, all INFO, WARN, ERROR, and FATAL messages). You can change the logging level of the `pl c` service using the loader controller client tool or the logging level of the other services.

Job data transformer

The job data is logged in the relational database in a raw format. At regular intervals, the job data transformer converts this data to a format usable by the reporting feature. By default, the data transformer converts the job data every hour at thirty minutes past the hour (that is, at 12:30am, 1:30am, and so on throughout the day).

To reschedule the transformation of data from the relational database to the reporting feature, you can change the data transformer schedule.

If your cluster has PERF controlled by EGO, you can edit the `j obdt.xml` configuration file, but you need to restart the `j obdt` service and EGO on the LSF master host after editing the file. The `j obdt.xml` file is located in the EGO service directory: `LSF_CONFDIR/ego/cluster_name/eservice/esc/conf/services`.

Loader controller

The loader controller manages the data loaders. By default, the loader controller manages the following data loaders:

- `bl dl oader` (License Scheduler data loader)
- `desktopj obdat al oader` (Desktop job data loader)
- `desktopcl i entdat al oader` (Desktop client data loader)
- `deskt opeventl oader` (Desktop active event data loader)
- `egoconsumerresl oader` (consumer resource data loader)
- `egodynami cresl oader` (dynamic metric data loader)
- `egoeventsl oader` (EGO allocation events data loader)
- `egostat i cresl oader` (static attribute data loader)
- `l sfbhostsl oader` (bhosts data loader)
- `l sfeventsl oader` (LSF events data loader)
- `l sfsl al oader` (SLA data loader)
- `l sfredprop l oader` (LSF resource properties data loader)
- `sharedresusagel oader` (share resource usage data loader)

You can view the status of the loader controller service using the loader controller client tool.

Log files for the loader controller and data loaders are available in the `PERF_LOGDIR` directory. There are logging levels that determine the detail of messages recorded in the log files. In decreasing level of detail, these are ALL (all messages), TRACE, DEBUG, INFO, WARN, ERROR, FATAL, and OFF (no messages). By default, all service log files log messages of INFO level or higher (that is, all INFO, WARN, ERROR, and FATAL messages). You can change the logging level of the `pl c` service using the loader controller client tool or the logging level of the data loaders using the client tool.

To balance data accuracy with computing power, you can change how often the data loaders collect data by changing the frequency of data collection per loader. To reduce the amount of unwanted data logged in the database, you can also disable individual data loaders from collecting data.

If you edit any `pl c` configuration files, you need to restart the `pl c` service.

If your cluster has PERF controlled by EGO, you can edit the `pl c_servi ce. xml` service configuration file, but you must restart the `pl c` service and EGO on the LSF master host after editing the file. The `pl c_servi ce. xml` file is located in the EGO service directory: `LSF_CONFDIR/ego/cl uster_name/eservi ce/esc/conf/servi ces`.

Data purger

The relational database needs to be kept to a reasonable size to maintain optimal efficiency. The data purger manages the database size by purging old data at regular intervals. By default, the data purger purges records older than 14 days at 12:30am every day.

To reschedule the purging of old data, you can change the purger schedule. To reduce or increase the number of records in the database, you can change the duration of time that records are stored in the database. If there are specific tables that are containing too much or too little data, you can also change the duration of time that records are stored in each individual table within the database.

If you edit any purger configuration files, you need to restart the purger service.

If your cluster has PERF controlled by EGO, you can edit the `purger_servi ce. xml` service configuration file, but you must restart the purger service and EGO on the LSF master host after editing the file. The `purger_servi ce. xml` file is located in the EGO service directory: `LSF_CONFDIR/ego/cl uster_name/eservi ce/esc/conf/servi ces`.

Event data files

The events logger stores event data in event data files. The EGO allocation event data file (for EGO-enabled clusters only) is named `ego.stream` by default and has a default maximum size of 10MB. The LSF event data file is named `lsb.stream` by default and has a default maximum size of 100MB. When a data file exceeds this size, the events logger archives the file and creates a new data file.

The events logger only maintains one archive file and overwrites the old archive with the new archive. The default archive file name is `ego.stream.0` for EGO and `lsb.stream.0` for LSF. The two LSF files are located in `LSF_TOP/work/cluster_name/logdir/stream` by default, and the two EGO files are located in `LSF_TOP/work/cluster_name/ego/data` by default. The event data loaders read both the data files and the archive files.

If your system logs a large number of events, you should increase the maximum file size to see more archived event data. If your disk space is insufficient for storing the four files, you should decrease the maximum file size, or change the file path to a location with sufficient storage space. Change the disk usage of your LSF event data files or the file path. Change the disk usage or file path of your EGO allocation event data files.

You can manage your event data files by editing the system configuration files. Edit `ego.conf` for the EGO allocation event data file configuration and `lsb.params` for the LSF event data file configuration.

Determine if your cluster is EGO-enabled and has PERF controlled by EGO

You need to determine whether your cluster is EGO-enabled and has PERF controlled by EGO in order to determine which command you use to manage the reporting services.

1. In the command console, run `egosh service list` to see the list of EGO services.
 - If you see a list of services showing that the reporting services are `STARTED`, your cluster is EGO-enabled and has PERF controlled by EGO. The reporting services are run as EGO services, and you use `egosh service` to manage the reporting services.
 - If you see a list of services showing that the reporting services are not `STARTED`, your cluster is EGO-enabled but does not have PERF controlled by EGO. You use `perfadmin` to manage the reporting services.
 - If you get an error running `egosh service list`, your cluster is not EGO-enabled and therefore does not have PERF controlled by EGO. You use `perfadmin` to manage the reporting services.

Stop or restart reporting services (PERF controlled by EGO)

Your cluster must have PERF controlled by EGO.

Stop or restart the `jobdt`, `plc`, and `purger` services. If your cluster has PERF controlled by EGO, the reporting services are run as EGO services, and you use the `egosh service` command to stop or restart these services.

1. In the command console, stop the service by running `egosh service stop`.


```
egosh service stop service_name
```
2. If you want to restart the service, run `egosh service start`.

```
egosh service start service_name
```

Stop or restart reporting services (PERF not controlled by EGO)

Your cluster must have PERF not controlled by EGO.

Stop or restart the `jobdt`, `plc`, and `purger` services. If your cluster does not have PERF controlled by EGO, you use the `perfadmin` command to stop or restart these services.

1. In the command console, stop the service by running `perfadmin stop`.

```
perfadmin stop service_name
```

2. If you want to restart the service, run `perfadmin start`.

```
perfadmin start service_name
```

View the status of the loader controller

Use the loader controller client tool to view the status of the loader controller.

1. Launch the loader controller client tool with the `-s` option.

- In UNIX, run `$PERF_TOP/version/bin/plcclient.sh -s`.
- In Windows, run `%PERF_TOP%\version\bin\plcclient -s`.

Dynamically change the log level of your loader controller log file

Use the loader controller client tool to dynamically change the log level of your `plc` log file if it does not cover enough detail, or covers too much, to suit your needs.

If you restart the `plc` service, the log level of your `plc` log file will be set back to the default level. To retain your new log level, change the level of your `plc` log file.

1. Launch the loader controller client tool with the `-l` option.

- In UNIX, run `$PERF_TOP/bin/plcclient.sh -l log_level`.
- In Windows, run `$PERF_TOP\bin\plcclient -l log_level`.

In decreasing level of detail, the log levels are ALL (for all messages), TRACE, DEBUG, INFO, WARN, ERROR, FATAL, and OFF (for no messages).

Dynamically change the log level of your data loader log files

Use the loader controller client tool to dynamically change the log level of your individual data loader log files if they do not cover enough detail, or cover too much, to suit your needs.

If you restart the `plc` service, the log level of your data loader log files will be set back to the default level. To retain your new log level, change the level of your data loader log files.

1. If you are using the default configuration file, launch the loader controller client tool with the `-n` and `-l` options.

Run `$PERF_TOP/version/bin/plcclient.sh -n data_loader_name -l log_level`.

In decreasing level of detail, the log levels are ALL (for all messages), TRACE, DEBUG, INFO, WARN, ERROR, FATAL, and OFF (for no messages).

Change the log level of your log files

Change the log level of your log files if they do not cover enough detail, or cover too much, to suit your needs.

1. Edit the `log4j.properties` file, located in the reports configuration directory (`PERF_CONFIGDIR`).
2. Navigate to the section representing the service you want to change, or to the default loader configuration if you want to change the log level of the data loaders, and look for the `log4j.logger.com.platform.perf` variable.

For example, to change the log level of the data purger log files, navigate to the following section, which is set to the default INFO level:

```
# Data purger ("purger") configuration log4j.logger.com.platform.perf.purger=INFO,
com.platform.perf.purger
```

3. Change the `log4j.logger.com.platform.perf` variable to the new logging level.

In decreasing level of detail, the valid values are ALL (for all messages), TRACE, DEBUG, INFO, WARN, ERROR, FATAL, and OFF (for no messages). The services or data loaders only log messages of the same or lower level of detail as specified by the `log4j.logger.com.platform.perf` variable. Therefore, if you change the log level to ERROR, the service or data loaders will only log ERROR and FATAL messages.

For example, to change the data purger log files to the ERROR log level:

```
# Data purger ("purger") configuration log4j.logger.com.platform.perf.purger=ERROR,
com.platform.perf.purger
```

4. Restart the service that you changed (or the `plc` service if you changed the data loader log level).

Change the disk usage of LSF event data files

If your system logs a large number of events and you have sufficient disk space, increase the disk space allocated to the LSF event data files.

1. Edit `lsb.params` and specify or change the `MAX_EVENT_STREAM_SIZE` parameter.

```
MAX_EVENT_STREAM_SIZE = integer
```

If unspecified, this is 1024 by default. Change this to the new desired file size in MB.

The recommended size is 2000 MB.

2. In the command console, reconfigure the LSF master host to activate this change.

```
badadmin reconfig
```

Change the location of the LSF event data files

If your system logs a large number of events and you do not have enough disk space, move the LSF event data files to another location.

1. Edit `lsb.params` and specify or change the `EVENT_STREAM_FILE` parameter.

```
EVENT_STREAM_FILE = file_path
```

If unspecified, this is `LSF_TOP/work/cluster_name/logdir/stream/lsb.stream` by default.

2. In the command console, reconfigure the LSF master host to activate this change.

```
badadmin reconfig
```

3. Restart the `plc` service on the LSF master host to activate this change.

Change the disk usage of EGO allocation event data files

Your cluster must be EGO-enabled.

If your system logs a large number of events, increase the disk space allocated to the EGO allocation event data files. If your disk space is insufficient, decrease the space allocated to the EGO allocation event data files or move these files to another location.

1. Edit `ego.conf`.
 - a) To change the size of each EGO allocation event data file, specify or change the `EGO_DATA_MAXSIZE` parameter.


```
EGO_DATA_MAXSIZE = integer
```

If unspecified, this is 10 by default. Change this to the new desired file size in MB.
 - b) To move the files to another location, specify or change the `EGO_DATA_FILE` parameter.


```
EGO_DATA_FILE = file_path
```

If unspecified, this is `LSF_TOP/work/cluster_name/ego/data/ego.stream` by default.
2. In the command console, restart EGO on the LSF master host to activate this change.

```
egosh ego restart master_host_name
```

Change the data purger schedule

Your cluster must be EGO-enabled.

To reschedule the deletion of old data, change the time in which the data purger deletes the old data.

1. Edit `purger_service.xml` in the EGO service directory.


```
LSF_CONFDIR/ego/cluster_name/eservice/esc/conf/services
```
2. Navigate to `<ego: Command>` with the `-t` parameter in the purger script.

```
<ego: Command> ... purger.sh -t ...
```

By default, the data purger is scheduled to delete old data at 12:30am every day.

3. Change the `-t` parameter in the data purger script to the new time (`-t new_time`).

You can change the data purger schedule to a specific daily time, or at regular time intervals, in minutes, from when the purger service first starts up.

For example, to change the schedule of the data purger:

- To delete old data at 11:15pm every day:

```
<ego: Command> ... purger... -t 23:15
```

- To delete old data every 12 hours from when the purger service first starts up:

```
<ego: Command> ... purger... -t *[12]
```

4. In the command console, restart EGO on the master host to activate these changes.

```
egosh ego restart master_host_name
```

5. Restart the purger service.

Change the data transformer schedule

Your cluster must be EGO-enabled.

To have reschedule the transformation of data from the relational database to the reporting feature, change the time in which the data transformer converts job data.

1. Edit `j obdt . xml` in the EGO service directory.

```
LSF_CONFDIR/ego/cluster_name/eservice/esc/conf/services
```

2. Navigate to `<ego: Command>` with the `-t` parameter in the purger script.

```
<ego: Command> ... j obdt . sh -t ...
```

By default, the data transformer converts the job data every hour at thirty minutes past the hour (that is, at 12:30am, 1:30am, and so on throughout the day).

3. Change the `-t` parameter in the data transformer script to the new time (`-t new_time`).

You can change the data transformer schedule to a specific daily time, a specific hourly time, or at regular time intervals, in minutes or hours, from when the `j obdt` service first starts up.

For example, to change the schedule of the data transformer:

- To convert job data at 10:20pm every day:

```
<ego: Command> ... j obdt . . . -t 22: 20
```

- To convert job data at the 25th minute of every hour:

```
<ego: Command> ... j obdt . . . -t *: * 25
```

- To convert job data every fifteen minutes from when the `j obdt` service first starts up:

```
<ego: Command> ... j obdt . . . -t *: *[ 15]
```

- To convert job data every two hours from when the `j obdt` service first starts up:

```
<ego: Command> ... j obdt . . . -t *[ 2]
```

4. In the command console, restart EGO on the master host to activate these changes.

```
egosh ego restart master_host_name
```

5. Restart the `j obdt` service.

Change the default record expiry time

To reduce or increase the number of records stored in the database, change the duration of time that a record is stored in the database before it is purged. This applies to all tables in the database unless you also specify the record expiry time in a particular table.

1. Edit the purger configuration files for your data loaders.

- For EGO data loaders, edit `purger_ego_rawdata . xml`.
- For LSF data loaders, edit `purger_lsf_bas ic_rawdata . xml`.

The purger configuration files are located in the `purger` subdirectory of the reports configuration directory:

\$PERF_CONFDIR/purger

2. In the `<TableList>` tag, edit the `Duration` attribute to your desired time in days, up to a maximum of 31 days.

For example, to have the records purged after 7 days:

```
<TableList Duration="7">
```

By default, the records are purged after 14 days.

3. Restart the purger service.

Change the record expiry time per table

To reduce or increase the number of records stored in the database for a particular table, change the duration of time that a record is stored in the database per table before it is purged. The duration only applies to this particular table.

1. Edit the purger configuration files for your data loaders.
 - For EGO data loaders, edit `purger_ego_rawdata.xml`.
 - For LSF data loaders, edit `purger_lsf_basic_rawdata.xml`.

The purger configuration files are located in the `purger` subdirectory of the reports configuration directory:

\$PERF_CONFDIR/purger

2. Navigate to the specific `<Table>` tag with the `TableName` attribute matching the table that you want to change.

For example:

```
<Table TableName="RESOURCE_METRICS" TimestampColumn="TIME_STAMP" ... />
```

3. Add or edit the `Duration` attribute with your desired time in days, up to a maximum of 31 days.

For example, to have the records in this table purged after 10 days:

```
<Table TableName="RESOURCE_METRICS" TimestampColumn="TIME_STAMP" Duration="10" ... />
```

4. Restart the purger service.

Change the frequency of data collection

To change how often the data loaders collect data, change the frequency of data collection per loader.

1. Edit the `plc` configuration files for your data loaders.
 - For EGO data loaders, edit `plc_ego.xml`.
 - For LSF data loaders, edit `plc_lsf.xml`.

The `plc` configuration files are located in the `plc` subdirectory of the reports configuration directory:

\$PERF_CONFDIR/plc

2. Navigate to the specific `<DataLoader>` tag with the `Name` attribute matching the data loader that you want to change.

For example:

```
<DataLoader Name="egodynamicsloader" Interval="300" ... />
```

3. Add or edit the `Interval` attribute with your desired time in seconds.

For example, to have this plug-in collect data every 200 seconds:

```
<DataLoader Name="egodynamicaloader" Interval="200" ... />
```

4. Restart the pl c service.

Disable data collection for individual data loaders

To reduce unwanted data from being logged in the database, disable data collection for individual data loaders.

1. Edit the pl c configuration files for your data loaders.

- For EGO data loaders, edit pl c_ego.xml .
- For LSF data loaders, edit pl c_lsf.xml .

The pl c configuration files are located in the pl c subdirectory of the reports configuration directory:

```
$PERF_CONFDIR/pl c
```

2. Navigate to the specific <DataLoader> tag with the Name attribute matching the data loader that you want to disable.

For example:

```
<DataLoader Name="egodynamicaloader" ... Enable="true" ... />
```

3. Edit the Enable attribute to "false".

For example, to disable data collection for this plug-in:

```
<DataLoader Name="egodynamicaloader" ... Enable="false" ... />
```

4. Restart the pl c service.

Test the reporting feature

Verify that components of the reporting feature are functioning properly.

1. Check that the reporting services are running.
 - If your cluster has PERF controlled by EGO, run **egosh service list**.
 - If your cluster has PERF not controlled by EGO, run **perfadmin list**.
2. Check that there are no error messages in the reporting logs.
 - a) View the loader controller log file.
 - UNIX: `$PERF_LOGDIR/pl c. log. host_name`
 - Windows: `%PERF_LOGDIR%/pl c. log. host_name. txt`
 - b) Verify that there are no ERROR messages and that, in the DataLoader Statistics section, there are data loader statistics messages for the data loaders in the last hour.

You need to find statistics messages for the following data loaders:

- bl d l o a d e r
 - d e s k t o p j o b d a t a l o a d e r
 - d e s k t o p c l i e n t d a t a l o a d e r
 - d e s k t o p e v e n t l o a d e r
 - l s f b h o s t s l o a d e r
 - l s f e v e n t s l o a d e r
 - l s f s l a l o a d e r
 - l s f r e s p r o p l o a d e r
 - s h a r e d r e s u s a g e l o a d e r
 - EGO data loaders (for EGO-enabled clusters only):
 - e g o c o n s u m e r r e s l o a d e r
 - e g o d y n a m i c r e s l o a d e r
 - e g o e v e n t s l o a d e r
 - e g o s t a t i c r e s l o a d e r
- c) View the data purger and data loader log files and verify that there are no ERROR messages in these files.

You need to view the following log files (*PERF_LOGDIR* is *LSF_LOGDIR/perf*):

- *PERF_LOGDIR*/data loader/bl d l o a d e r. *host_name*. log
- *PERF_LOGDIR*/data loader/desktopjobdata loader. *host_name*. log
- *PERF_LOGDIR*/data loader/desktopclientdata loader. *host_name*. log
- *PERF_LOGDIR*/data loader/desktopevent loader. *host_name*. log
- *PERF_LOGDIR*/job d t. *host_name*. log
- *PERF_LOGDIR*/data loader/l s f b h o s t s l o a d e r. *host_name*. log
- *PERF_LOGDIR*/data loader/l s f e v e n t s l o a d e r. *host_name*. log
- *PERF_LOGDIR*/data loader/l s f s l a l o a d e r. *host_name*. log
- *PERF_LOGDIR*/purger. *host_name*. log
- *PERF_LOGDIR*/data loader/l s f r e s p r o p l o a d e r. *host_name*. log
- *PERF_LOGDIR*/data loader/sharedresusagel o a d e r. *host_name*. log
- EGO data loader log files (EGO-enabled clusters only):

- *PERF_LOGDIR*/data/oader/egoconsumerres/oader. *host_name*.log
 - *PERF_LOGDIR*/data/oader/egodynamicires/oader. *host_name*.log
 - *PERF_LOGDIR*/data/oader/egoeventsl/oader. *host_name*.log
 - *PERF_LOGDIR*/data/oader/egostatires/oader. *host_name*.log
3. Check the report output.
 - a) Produce a standard report.
 - b) Verify that the standard report produces a chart or table with data for your cluster.

If you were not able to verify that these components are functioning properly, identify the cause of these problems and correct them.

Platform Application Center Web Services

Introduction to Platform Application Center Web Services

Platform Application Center provides standard RESTful web services to allow application submission, data management, job information query, job actions, and more. The web service can be accessed from any host in network. The host that accesses the web service is called the web service client host. The client host must run on Linux, but it does not need to have Platform Application Center or LSF installed as long as standard HTTP protocol is supported.

The Platform Application Center web service API can be integrated with many languages and methods: Python, Perl, Java, and more.

License Requirements

Platform Application Center web services are licensed and controlled by the license feature `pcc_app_center_enterprise` in the `PAC_TOP/gui/conf/license.dat` file. The license type can be DEMO, OEM or floating license.

User Requirements

Log into Platform Application Center at least once before using the web service. This automatically configures user access permissions for your account.

Protocol and Port

Web services share the same protocol and port with Platform Application Center.

The default protocol is HTTP, port: 8080. The default web service URL is `http://host_name:8080`.

If you enable HTTPS, the default port is 8443. The web service URL is `https://host_name:8443`.

To change protocol and port:

1. Edit `$GUI_CONFDIR/wsm_web_gui.conf`:
 - `CATALINA_START_PORT=8080`
 - `CATALINA_HTTPS_START_PORT=8443`
2. Run `pacadmin` to switch between HTTP and HTTPS.

Client Host Requirements

The client host has the following requirements:

- Python 2.6.x installed (<http://www.python.org/download/releases/>)
- `httplib2 V0.6` (<http://code.google.com/p/httplib2/downloads/list>)

pacclient.py

Get `pac_api.py` and `pacclient.py` from the Platform Application Center server: `PAC_TOP/gui/3.0/bin/`

1. To log in, run `pacclient.py login`. The connection information and security information is saved in `$HOME/.pacpass`.

2. To perform a task such as listing jobs or submitting a job, run the corresponding command, such as `paccli ent.py job` or `paccli ent.py submit`.
3. To log out, run `paccli ent.py logout`.

pacclient.py

The `pacclient.py` command is used to access the web service for Platform Application Center.

Synopsis

```
pacclient.py subcommand [options]
```

```
pacclient.py [subcommand] -h | help
```

```
pacclient.py -V
```

Description

Use `pacclient.py` to submit commands to Platform Application Center through the web service.

-h | help

Prints command or subcommand usage to `stdout` and exits.

-V

Prints product version to `stdout` and exits.

Subcommand synopsis

pacclient.py ping

```
pacclient.py ping [--url|-l URL]
```

pacclient.py logon

```
pacclient.py logon
```

```
 [--url|-l URL]
```

```
 [--user|-u user_name]
```

```
 [--pass|-p password]
```

pacclient.py logout

```
pacclient.py logout
```

pacclient.py app

```
pacclient.py app
```

```
 [--param|-p app_name[:template_name]]
```

pacclient.py submit

```
pacclient.py submit
```

```
 --app|-a app_name[:template_name]
```

```
 [--conf|-c file_path]
```

```
 [--param|-p field_ID=value;[field_ID=value;]]
```

pacclient.py job

```
pacclient.py job
[--name|-n job_name]
[--status|-s Pending|Running|Done|Exited|Suspended]
[job_ID]
```

pacclient.py jobaction

```
pacclient.py jobaction
--action|-a kill|suspend|requeue|resume
job_ID
```

pacclient.py jobdata

```
pacclient.py jobdata job_ID
```

pacclient.py download

```
pacclient.py download
[--dir|-d directory]
[--file|-f file_name]
job_ID
```

pacclient.py ping

Detects whether or not the web service is running at the specified URL.

For example:

```
pacclient.py ping -l http://pac_host:8080/
```

If `-l` is omitted, you will be prompted interactively.

[--url|-l *URL*]

Specify the URL of the Platform Application Center web service in the format:

http://host_name:port_number

pacclient.py logon

Logs you in to Platform Application Center.

For example:

```
pacclient.py logon -l http://pac_host:8080/ -u user2 -p mypassword
```

Use double quotes around any value that contains spaces.

If any parameter is omitted, you will be prompted interactively.

Once you are logged in, you may run other `pacclient.py` commands, until you log out. You will be logged out automatically if you do not run any `pacclient.py` commands for 60 minutes consecutively.

[- - url | - l *URL*]

Specify the URL of the Platform Application Center web service in the format:

http://pac_host:port_number

[- - user | - u *user_name*]

Specify your user name.

[- - pass | - p *password*]

Specify your password.

pacclient.py logout

Logs you out of Platform Application Center.

For example:

```
pacclient.py logout
```

Once you are logged out, you must log in to run more `pacclient.py` commands.

You will be logged out automatically if you do not run any `pacclient.py` commands for 60 minutes consecutively.

pacclient.py app

List applications or parameters of an application.

For example, to list all the parameters and default values for an application:

```
pacclient.py app -p FLUENT
```

By default, lists all applications and application status.

[- - param | - p *app_name[:template_name]*]

List all parameters and default values for an application. To see parameter values saved in your customized application, specify both the published form name and the custom template name.

pacclient.py submit

Submits a job to Platform Application Center.

For example:

```
pacclient.py submit -a "FLUENT" -c C:\mydir\fluentconf.txt -p  
CONSOLE_SUPPORT=yes; CAS_INPUT_FILE=C:\mydir\fluent\fluent-test.cas.gz link
```

Use double quotes around any value that contains spaces.

Parameters defined in the command line (-p) override all others; parameters defined in a file (-c) override parameters defined in an application form (-a).

--app|-a "*app_name[:template_name]*"

Required. Specify the application form to use for default job submission parameters.

Specify the application name and optionally the name of your customized template.

[- - conf | - c *file_path*]

Define the job submission parameters from a file.

Specify the path to a file on the local host that defines job parameters and input files. The file format is a list of field ID and value pairs:

- *field_ID*

Specify the field ID from the application form template.

You must be an administrator to access editing mode and view the field IDs.

- *value*

Specify the value for the field input.

If the field requires a file for input, you must also choose to upload the file, copy the file to the job directory, or link the file to the job directory. Specify the value in the format:

file_path, upload|copy|link

For example:

```
[Parameter]
JOB_NAME=FF_20100329
VERSION=6.3.26
CONSOLE_SUPPORT=No
[Input file]
FLUENT_JOURNAL=C:\demo\fluent\fluent-test.job, upload
CAS_INPUT_FILE=C:\demo\fluent\fluent-test.cas.gz, upload
```

[- - param | - p *field_ID=value*; [*field_ID=value*;]...]

Define the job submission parameters in the command line.

Specify field ID and value pairs as described in the - - conf | - c option.

pacclient.py job

Show information for one or more jobs. You must specify one and only one method to define which jobs to show.

For example:

```
pacclient.py job -s Running
```

```
pacclient.py job 54321
```

Output includes the external status for a job in the column BSTATUS. You use bpost -d to provide external status information for a job. You can also view this status in the Jobs page, with the Ext Status column. You will need to select the column for display in Options > Preferences.

[- - name | - n *job_name*]

Show information for all jobs matching the specified job name.

You can use the wildcard character "*" in the job name.

[- - status | - s Pending|Running|Done|Exited|Suspended]

Show information for all jobs having the specified status.

[*job_ID*]

Show information for a single job, specify the job ID.

pacclient.py jobaction

Perform a job action on a job.

For example:

```
pacclient.py jobaction -a resume 54321  
--action|-a kill|suspend|requeue|resume
```

Specify the job action. You can kill, suspend, requeue, or resume a job.

job_ID

Specify the job ID.

pacclient.py jobdata

List all the files for a job.

For example:

```
pacclient.py jobdata 1234
```

job_ID

Specify the job ID.

pacclient.py download

Download job data for a job.

For example:

```
pacclient.py download 54321
```

By default, copy all job data files to the current working directory.

[--dir|-d *directory*]

Copy files to the specified directory.

[--file|-f *file_name*]

Download the specified file only.

job_ID

Specify the job ID.

RESTful web service reference

User name, password, and security token

Platform Application Center supports console and web service clients through the same services. The authentication mechanism for web services is same as the Platform Application Center console. The user name and password are from the current Platform Application Center server configuration.

Before calling any service, a security token must be obtained through the logon service. The logon service requires a user name and password, and returns an encrypted security token string. The request must set a cookie in the form

```
platform_token=<token>
```

along with the HTTP request. For example:

```
platform_token=lsfadmin"2010-04-03T09:31:02Z"UihSStsCCyv9yXJ+1hyiaJXrHr6wtc4ZT
+V0lVWKEe99L3Ht4RJBXr7hfzd0nNaJWmWP5l2kyuPZZAcFDIalVwAlS0mhqKLcWLCafadHpgB1LYf3/
g1V8pDUIngGbCe"054LJVytYAa7QagStQiqdw==
```

After all web service requests are done, client must call the logout service to release the license. Otherwise, the license will be occupied for a period of time (by default 60 minutes), then Platform Application Center will log the user out and release the license automatically.

URL

RESTful services accept a URL as a service connection point through standard HTTP or HTTPS protocol. The URL has the format

base_URL/function_URL

where

```
base_URL= http://pac_host:port/
```

or

```
base_URL= https://pac_host:port/
```

```
function_URL=webservi ce/functi on
```

For example:

```
http://host1.my.company.com:8080/webservi ce/logon
```

The default port value is 8080, which is configured in the Platform Application Center configure file `$GUI_CONFDIR/wsm_webgui.conf`.

```
<Parameter>
    <!-- Restful service listening port -->
    <Name>REST_PORT</Name>
    <Value>8080</Value>
</Parameter>
```

Ping (using GET)

Purpose

The ping service detects whether or not a web service is running at the specified URL.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http:// <hostname>:<port>/ webservice/ping	
Content-Type	text/plain	
Accept	text/plain	
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	Web Services are ready on URL: http://xxx:ppp/	
Failure Message	NULL	

Ping (using POST)

Purpose

The ping service detects whether or not a web service is running at the specified URL.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	POST	
URL	http:// <hostname>:<port>/ webservice/pingPost	
Content-Type	text/plain	
Accept	text/plain	
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	Web Services are ready on URL: http://xxx:ppp/	
Failure Message	NULL	

Logon (using GET)

Purpose

The logon service generates the security token required to access the other web services.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http://<hostname>:<port>/webservice/logon	
Content-Type	application/xml	
Accept	text/xml	
Body	<User><name>%s</name> <pass>%s</pass> </User>	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<User><token>%s</token></User>	%s=return token
Failure Message	<User><errMsg>%s</errMsg></User>	%s=return error message

Logon (using POST)

Purpose

The logon service generates the security token required to access the other web services.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	POST	
URL	http://<hostname>:<port>/webservice/logonPost	
Content-Type	application/xml	
Accept	text/xml	
Body	<User><name>%s</name> <pass>%s</pass> </User>	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<User><token>%s</token></User>	%s=return token
Failure Message	<User><errMsg>%s<.errMsg></User>	%s=return error message

Logout

Purpose

The logout service invalidates the security token, and the user can no longer access the other web services.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http://<hostname>:<port>/webservice/logout	
Content-Type	text/plain	
Accept	text/plain	
Cookie	platform_token=<token> <token>: returned from logon	
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	

HTTP Response	Field format/value	Notes
Success Message	"ok"	
Failure Message	Other than "ok"	

Get Application List

Purpose

The get application list service lists all applications and application status.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http:// <hostname>:<port>/ webservice/appStatus	
Content-Type	text/plain	
Accept	text/plain	
Cookie	platform_token=<token> <token>: returned from logon	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<AppInfos><AppInfo><a ppName>%s</ appName><status>%s</ status></AppInfo> <AppInfo><appName>% s</appName><status>% s</status></AppInfo> <AppInfo>...</AppInfo>	
Failure Message	<AppInfos><errMsg>% s</errMsg></AppInfo>	Error details are in the log file.

Get Application Parameters (using GET)

Purpose

The get application parameters service lists all parameters of an application.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http:// <hostname>:<port>/ webservice/appParams	
Content-Type	text/plain	
Accept	text/plain	
Cookie	platform_token=<token>	<token>: returned from logon
Body	Application name	Mandatory parameter

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<AppParams><AppPara m><id>%s</id><label> %s</label<mandatory> %s</ mandatory><defaultValu e>%s</defaultValue>></ AppParam> <AppInfo>...</AppInfo> <AppParams>	
Failure Message	<AppParam><errMsg> %s</errMsg></ AppParam>	Error details are in the log file.

Get Application Parameters (using POST)

Purpose

The get application parameters service lists all parameters of an application.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	POST	
URL	http://<hostname>:<port>/webservice/ appParamsPost	

HTTP Request Field	Field format/value	Notes
Content-Type	text/plain	
Accept	text/plain	
Cookie	platform_token=<token>	<token>: returned from logon
Body	Application name	Mandatory parameter

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<AppParams><AppParam><id>%s</id><label>%s</label><mandatory>%s</mandatory><defaultValue>%s</defaultValue></AppParam><AppParam>...</AppParam><AppParams>	
Failure Message	<AppParam><errMsg>%s</errMsg></AppParam>	Error details are in the log file.

Submit Job

Purpose

The submit job service submits a job to Platform Application Center.

- Precondition: the administrator has prepared and published the application.
- For any parameter, the value specified by the web service request overrides the default value specified by the application form.
- The job is submitted as the logged in user.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	POST	
URL	http://<hostname>:<port>/webservice/submitapp	
Content-Type (header)	multipart/mixed;boundary=YYYYYY	
Accept	text/xml	
Cookie	platform_token=<token>	<token>:returned from logon

HTTP Request Field	Field format/value	Notes
Body	--boundary <Application Name Area (table 1)> --boundary <Parameters Area (table 2)> --boundary <File Attachment (table 3)>	boundary can be any random unique string. For details, see additional tables.

Application Name Area

Field Name	Field format/value	Notes
Content-Disposition	Form-data;name="AppName"	
Content-ID	AppName	
Content	<App_Type[:App_Instance]>	Specify App_Type only if there is no App_Instance

```

Example
-----
Content-Disposition: form-data; name=AppName
Content-ID: <AppName>
"\r\n"
FLUENT: FLUENT_WEB
-----
    
```

Parameters Area

Field Name	Field format/value	Notes
Content-Disposition	Form-data;name="data"	
Content-Type	multipart/mixed;boundary=XXXX	
Content-ID	<Data>	

Field Name	Field format/value	Notes
Content	<pre>--boundary Content-Disposition: form-data; name="c1" Content-Type: application/xml; charset=US-ASCII Content-Transfer- Encoding: 8bit "\r\n" <AppParam> <id>%s</ id> <value>%s</ value><type>%s</type> </AppParam> --boundary Content-Disposition: form-data; name="c1" Content-Type: application/xml; charset=US-ASCII Content-Transfer- Encoding: 8bit "\r\n" <AppParam> <id>%s</ id> <value>%s</ value><type>%s</type> </AppParam> --boundary ... --boundary--</pre>	<p><id>%s</id> defines the parameter name on the application form</p> <p><value>%s</value> defines the value of the parameter. The value format for a file:</p> <p><Filename>,upload copy link</p> <p>Upload: upload file from local. File must be attached in attached area of this request.</p> <p>Copy: copy file to job directory</p> <p>Link: link file to job directory.</p> <p><type>%s</type> defines the parameter type. The possible values are: "file" or empty. "file" indicates the current parameter is a file .</p>

Example

```
-----
Content-Disposition: form-data; name='data'
Content-Type: multipart/mixed; boundary=_Part_1_701508.1145579811786
Content-ID: <data>
"\r\n"
--_Part_1_701508.1145579811786
Content-Disposition: form-data; name='c1'
Content-Type: application/xml; charset=US-ASCII
Content-Transfer-Encoding: 8bit
"\r\n"
<AppParam> <id>JOB_NAME</id> <value>From_webS</value><type></type> </AppParam>
--_Part_1_701508.1145579811786
Content-Disposition: form-data; name='c3'
Content-Type: application/xml; charset=US-ASCII
Content-Transfer-Encoding: 8bit
"\r\n"
<AppParam> <id>FLUENT_JOURNAL</id> <value> fluent-test.jou,upload </value> <type>
file</type> </AppParam>
--_Part_1_701508.1145579811786--
-----
```

File Attachment

Field Name	Field format/value	Notes
Content-Disposition	Form-data;name="f1"	
Content-Type	application/octet-stream	
Content-ID	<Filename>	
Content-Transfer-Encoding	UTF-8	
Content	<File content in bytes>	

Example

```
-----
Content-Disposition: form-data; name='f1'
Content-Type: application/octet-stream
Content-Transfer-Encoding: UTF-8
Content-ID: <fluent-test.job>
"\r\n"
PGh0bWw+Ci AgPGhl YWQ+Ci AgPC9oZWFKPgogI Dxi b2R5PgogI CAgPHA+VGhpcyBpcyB0aGUg
-----
```

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<Job><name>%s</name><id>%s</id> <status>%s</status><cmd>%s</cmd></Job>	
Failure Message	<Job><errMsg>%s</errMsg></Job>	%s=return error message

Get Job

Purpose

The get job service shows job information for a job with the specified job ID.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http://<hostname>:<port>/webservice/jobs/<id>	<id>: integer

HTTP Request Field	Field format/value	Notes
Content-Type	application/xml	
Accept	text/xml	
Cookie	platform_token=<token>	<token>: returned from logon
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<Job><name>%s</name><id>%s</id><status>%s</status><cmd>%s</cmd></Job>	
Failure Message	<Job><errMsg>%s</errMsg></Job>	%s=return error message

Get Jobs for Status

Purpose

The get jobs for status service shows job information for jobs having the specified job status.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http://<hostname>:<port>/webservice/jobsforstatus/<status>	<status>: one of: Running, Pending, Exited, Done, Suspended
Content-Type	application/xml	
Accept	text/xml	
Cookie	platform_token=<token>	<token>: returned from logon
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<Jobs><Job><name>%s</name><id>%s</id><status>%s</status><cmd>%s</cmd></Job> <Job> ...</Job> ... </Jobs>	
Failure Message	<Jobs><Job><errMsg>%s</errMsg></Job></Jobs>	%s=return error message

Get Jobs for Name

Purpose

The get jobs for name service shows job information for jobs matching the specified job name pattern.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http://<hostname>:<port>/webservice/jobsforname/<jobName>	<jobName>: alphanumeric string. Support "*" to match multiple jobs.
Content-Type	application/xml	
Accept	text/xml	
Cookie	platform_token=<token>	<token>: returned from logon
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	

HTTP Response	Field format/value	Notes
Success Message	<pre><Jobs><Job><name>%s</name><id>%s</id><status>%s</status><cmd>%s</cmd></Job> <Job> ...</Job> ... </Jobs></pre>	
Failure Message	<pre><Jobs><Job><errMsg>%s<.errMsg></Job></Jobs></pre>	%s=return error message

Job Operation

Purpose

The job operation service controls jobs. The following job control actions are supported: kill, suspend, resume, and requeue.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	<pre>http:// <hostname>:<port>/ webservice/ jobOperation/<action>/ <ID></pre>	<action>: one of the following: <ul style="list-style-type: none"> kill suspend requeue resume
Content-Type	application/xml	
Accept	text/xml	
Cookie	platform_token=<token>	<token>: returned from logon
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	<pre><Job><actionMsg>%s</actionMsg></Job></pre>	%s=returned action output message

HTTP Response	Field format/value	Notes
Failure Message	<Job><errMsg>%s</errMsg></Job>	%s=return error message

Get Job File List

Purpose

The get job file list service lists all the files for a job.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http://<hostname>:<port>/webservice/jobfiles/<ID>	<ID>: a positive integer
Content-Type	text/plain	
Accept	text/plain	
Cookie	platform_token=<token>	<token>: returned from logon
Body	Empty	

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	/shareDisk/jobRepo/ lsfadmin/myJob_22111/ a.tar.z; /shareDisk/jobRepo/ lsfadmin/myJob_22111/ output.txt	Files are separated by “;”. File names started with “.” are filtered out.
Failure Message	Empty	Error details are in the log file.

Download a File (using GET)

Purpose

The download a file service downloads job data for a job.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	GET	
URL	http:// <hostname>:<port>/ webservice/file/<ID>	<ID>: a positive integer
Content-Type	text/plain	
Accept	text/plain	
Cookie	platform_token=<token>	<token>: returned from logon
Body	/shareDisk/jobRepo/ lsfadmin/myJob_22111/ a.tar.z OR a.tar.z	The file path. A relative path is interpreted relative to the job directory.

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	File content in bytes.	
Failure Message	Empty	Error details are in the log file.

Download a File (using POST)

Purpose

The download a file service downloads job data for a job.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	POST	
URL	http://<hostname>:<port>/webservice/filePost/<ID>	<ID>: a positive integer
Content-Type	text/plain	
Accept	text/plain	
Cookie	platform_token=<token>	<token>: returned from logon

HTTP Request Field	Field format/value	Notes
Body	/shareDisk/jobRepo/lfsadmin/myJob_22111/a.tar.z OR a.tar.z	The file path. A relative path is interpreted relative to the job directory.

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	File content in bytes.	
Failure Message	Empty	Error details are in the log file.

Python API reference

By default, Platform Application Center provides the client API in Python. It provides an example of how to access the Platform Application Center web services.

The location is:

```
PAC_TOP/gui /3.0/bin/pac_api.py
```

To use the API, the Python program must import the modules:

- sys
- os
- getopt
- urllib2
- From xml.dom, import minidom
- From ConfigParser, import ConfigParser
- getpass
- From pac_api, import *

ping (url)

Description

The ping API implements the ping web service. It detects whether or not a web service is running at the specified URL.

Input

url

A web service URL

Format: `http://host_name:port_number/` or `https://host_name:port_number/`

Output

On success

"Web services are ready..."

On failure

"Web services are not ready..."

Logon (url, username, password)

Description

The logon API implements the logon web service. It generates the security token required to access the other web services.

The URL and security token are saved to: `$HOME/.pacpass`.

Input

url

Non-empty string

Format: `http://host_name:port_number/` or `https://host_name:port_number/`

username

Non-empty string

password

Non-empty string

Output

On success

- Get security token
- Save URL and token in `.pacpass`

On failure

Print error message

Logout ()

Description

The `logout` API implements the logout web service. It invalidates the security token, and the user can no longer access the other web services.

`logout` releases the license attached to the token at the following location: `$HOME/.pacpass`.

`logout` also removes the `.pacpass` file.

Input

`$HOME/.pacpass` or `$HOMEPATH\.pacpass`

Output

On success

- Release the license from the Platform Application Center host
- Remove `.pacpass` from the client

On failure

Print error message

`status,message = getAllAppStatus()`

Description

The `getAllAppStatus` API implements the get application list web service. It lists the applications and application status.

Input

Empty

Output

On success

(status="ok")

```
message = <AppInfos><AppInfo><appName>%s</appName><status>%s</status></AppInfo>
```

```
<AppInfo><appName>%s</appName><status>%s</status></AppInfo><AppInfo>...</AppInfo></AppInfos>
```

On failure

(status="error")

```
message = <AppInfo><errMsg>Error message </errMsg></AppInfo>
```

status,message = getAppParameter(appName)

Description

The `getAppParameter` API implements the get application parameters web service. It lists the parameters of an application.

Input

Job ID

Output

On success

(status="ok")

```
message = <AppParams><AppParam><id>%s</id><label>%s</label<mandatory>%s</mandatory>
```

```
<defaultValue>%s</defaultValue></AppParam>
```

```
<AppInfo>...</AppInfo>
```

```
<AppParams>
```

On failure

(status="error")

```
message = <AppInfo><errMsg>Error message </errMsg></AppInfo>
```

status,message = submitJob(jobDict)

Description

The submitJob API implements the submit job web service. It submits a job to Platform Application Center.

The API specifies job parameters and uploads the required data files, and submits the job to LSF.

Input

jobDict

Job parameters dictionary. It must include all the following:

- JobDict[APP_NAME]= <App_Type>:<App_Instance>
- JobDict['PARAMS']={ <ID>:<VALUE>, <ID>:<VALUE> ... }
- JobDict['INPUT_FILES']={ <ID>:<FILEPATH, LOADING_TYPE>, <ID>:<FILEPATH, LOADING_TYPE> ... }

Where:

<ID> is the parameter ID from application form, could be from CONSOLE.

<LOADING_TYPE> is one of the following: “upload”, “copy”, “link”

The <FILEPATH> must be a local path for “upload”.

The <FILEPATH> must be a Platform Application Center server path for “copy” and “link”.

Output

On success

(status="ok") job ID

On failure

(status="error") error messages returned from the web service

status,message = getJobInfo (jobId)

Description

The getJobInfo API implements the get job web service. It shows job information for a job with the specified job ID.

Output includes the external status for a job. You use `bpost -d` to provide external status information for a job. You can also view this status in the Jobs page, with the Ext Status column. You will need to select the column for display in Options > Preferences.

Input

jobId

Non-empty string

Matches the LSF job ID

Output

On success

```
(status="ok") "<Job> <id>%s</id><name>%s</name><status>%s</status><cmd>%s</cmd><extStatus>%s</extStatus></Job>"
```

On failure

(status="error") error messages returned from the web service

status,message = getJobForStatus(jobStatus)

Description

The `getJobForStatus` API implements the `get jobs for status` web service. It shows job information for jobs having the specified job status.

Output includes the external status for a job. You use `bpost -d` to provide external status information for a job. You can also view this status in the Jobs page, with the Ext Status column. You will need to select the column for display in Options > Preferences.

Input

jobStatus

One of the following:

- Running
- Pending
- Suspended
- Exited
- Done

Output

On success

```
(status="ok")
```

```
"<Jobs> <Job> <id>%s</id> <name>%s</name><status>%s</status><cmd>%s</cmd></Job><extStatus>%s</extStatus><Job> <id>%s</id> <name>%s</name><status>%s</status><cmd>%s</cmd><extStatus>%s</extStatus></Job>...</Jobs>"
```

On failure

(status="error") error messages returned from the web service

status,message = getJobForName(jobName)

Description

The `getJobForName` API implements the `get jobs for name` web service. It shows job information for jobs matching the specified job name pattern.

Output includes the external status for a job. You use `bpost -d` to provide external status information for a job. You can also view this status in the Jobs page, with the Ext Status column. You will need to select the column for display in Options > Preferences.

Input

jobName

Job name string pattern

"*" is supported in the job name string to match multiple jobs

Output

On success

(status="ok")

```
"<Jobs><Job> <id>%s</id> <name>%s</name><status>%s</status><cmd>%s</cmd><extStatus>%s</extStatus></Job><Job> <id>%s</id> <name>%s</name><status>%s</status><cmd>%s</cmd><extStatus>%s</extStatus></Job>...</Jobs>"
```

On failure

(status="error") error messages returned from the web service

For example: "Error to get Job info for job name: fluent*"

status,message = doJobAction(jobAction, jobId)

Description

The `doJobAction` API implements the job operation web service. It controls jobs. The following job control actions are supported: kill, suspend, resume, and requeue.

Input

jobAction

One of the following:

- Kill
- Suspend
- Resume
- Requeue

jobId

A unique number for the job ID

Output

On success

(status="ok") action response message from LSF

On failure

(status="error") error messages returned from the web service

files = jobDataList(jobId)

Description

The `jobDataList` API implements the get job file list web service. It lists all the files for a job.

Input

Job ID

Output

A string in the format:

host_name:path;host_name2:path2; ... host_nameN:pathN

downloadJobFiles(jobId, dstPath, dFile)

Description

The `downloadJobFiles` API implements the download a file web service. It downloads job data for a job.

This API downloads all files under the job directory.

Input

jobId

Non-empty string

Matches the LSF job ID

dstPath

The destination directory path

If it is an empty string, use the current directory

dFile

The name of the file

If it is an empty string, download all files

Output

On success

Download the job files and copy them to `<dstPath>`

On failure

Do not copy any files, check error message in Platform Application Center log file

Create Custom Pages with the SDK

Overview

Use the Platform Application Center page customization framework to easily extend the default functionality of Platform Application Center.

The framework provides APIs to create new web interfaces based on your specific requirements. Each new page and its related workflows are defined through an XML element `<pac: service>` in the service repository file: `services.xml`.

To create a new page, edit `services.xml` and add a new XML element: `<pac: service>` to create a new Platform Application Center service, which usually corresponds to a new page in Platform Application Center.

Then edit `PAC_TOP/gui/conf/navigation/pmc_tree.xml` to integrate the new page into Platform Application Center.

You can add or remove pages in `services.xml` and your changes are automatically recognized by Platform Application Center when you click Refresh. You do not need to restart Platform Application Center.

New page structure

A new page supported through this framework is like a normal HTML page with Platform Application Center tags (of the form `<pac: xxxx>`). It can use HTML tags, `<pac: xxxx>` tags, JavaScript, CSS, and shell scripts.

`<pac: xxxx>` tags define user input, trigger events, and connect with scripts and APIs.

Location of services.xml

The location of `services.xml` is `PAC_TOP/gui/conf/service/services.xml`, where `PAC_TOP` is the path to the top-level Platform Application Center installation directory (for example, `/opt/pac`).

A simple service example

The following XML element defines a simple service:

1. Show a page with a simple form and a Submit button
2. Echo a welcome message after the user clicks the Submit button

```
<pac: service id="hello_pac">
  <pac: option id="USER_NAME" label="@{userName}:" type="text" />
  <pac: action id="submit" label="@{submit}">
    PAC_TOP/gui/conf/service/buildin/hello.sh hello_pac
  <pac: result type="text/html" />
</pac: action>
</pac: service>
```

The `<pac: option>` and `<pac: action>` tags are the basic tags which make up a single `<pac: service>` section, and each functional `<pac: service>` section must have at least one option or action.

The options defined in XML become variables in the environment, and are used for the execution of the script, together with a rich set of additional information about the context of your execution.

Custom page examples

Several examples in the default `services.xml` file illustrate how to use the Platform Application Center page customization framework to create new custom pages in Platform Application Center. Use the examples to build your own custom pages in Platform Application Center.

Select Settings > Custom Page Examples to try out the example pages.

View the `services.xml` file source from the example page from this URL:

```
http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=viewsource
```

Multiple step job submission form (serviceId=submitjob)

This example demonstrates basic job submission and LSF job submission output display. After submitting a job, click Data List to see a list of data files (calls `serviceId=data`).

In the example source, this example has `serviceId=submitjob`.

Access Step 1, the first page of the example, at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=submitjob`

Job submission form with redirect to another form (serviceId=serviceflow)

In this example, clicking Submit navigates to a separate form (`serviceId=formelements`), which then submits a job to the single page job submission form (`serviceId=flow_it10`).

In the example source, this example has `serviceId=serviceflow`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=serviceflow`

Single page job submission form with local file upload (serviceId=formelements)

This example demonstrates a page for submitting a job to an application and how to upload a local file. It illustrates code for all supported field types you can implement, and shows you how to display the location of the file upload.

In the example source, this example has `serviceId=formelements`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=formelements`

Submit a job and go to the job details page (serviceId=JOB_CUSTOM_Sample_Service)

This example demonstrates how to submit a job through a job submission form, and display the job details page after submission. It also demonstrates how to create drop-down fields in the submission form.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=JOB_CUSTOM_Sample_Service`

Submit a job in a job group and go to the job group details page (serviceld=JOBGROUP_CUSTOM_Sample_Service)

This example demonstrates how to submit a job associated with a job group and display the job group details page after submission.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceld=JOBGROUP_CUSTOM_Sample_Service`

AJAX call and parameter passing (serviceld=ajaxcall)

This example demonstrates how to pass parameters entered on a form through an AJAX call and display the value of the passed parameters in a text field.

In the example code, the function `blah` is triggered through an AJAX call, and the parameters are passed through an array list named `arr`. Inside the `blah` function, the each name/value pair in `arr` is converted into an environment variable.

The Show Info button displays the text input values.

In the example source, this example has `serviceld=ajaxcall`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceld=ajaxcall`

Cache data and results with AJAX (serviceld=ajaxspooler)

This example demonstrates how to create a temporary spooler file with AJAX, writing data to a file, and retrieving data from the file.

The example writes the output of the `hostname` command in the file, and retrieves data from the file.

Click Save `hostname` to `name.txt` to create the spooler and write the `hostname` command output to file `name.txt`.

Click Show `name.txt` to display the contents of `name.txt`.

In the example source, this example has `serviceld=ajaxspooler`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceld=ajaxspooler`

Call a service and pass parameters (serviceld=callservice)

This page redirects to service with parameters. In the example, the variable `$PAC_SPOOLER_URI` is automatically generated by Platform Application Center when `${PAC_SPOOLER_DIR}` is defined in the `pac:spooler`.

Click the Jack link, and the parameter value will be sent to another service (`serviceld=receive`).

The redirected page receives the parameters from the `servicecallservice` and displays the input as follows:

`http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=receive&name=Jack`

The input value is displayed in a text element labelled Name.

In the example source, this example has `serviceId=callservice`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=callservice`

Update a page from the spooler (serviceId=fromspooler)

This example demonstrates how to retrieve information from the spooler and how refreshing a page with updated spooler data. Click Get Image to retrieve an image file from the spooler, and show it on the same page. In the example source, this example has `serviceId=fromspooler`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=fromspooler`

Trigger a job flow (serviceId=JOBFLOW_CUSTOM_Sample_Service)

This example demonstrates how to trigger a job flow through a job submission form and shows examples of submission form dropdown fields.

After setting the submission parameters on the page, click Trigger to trigger the job flow.

The application script must return a flow-details element. For example,

```
<pac:flow-details id="10" redirect='${JOBFLOW_DETAIL_URL?instance=${FLOWID}'} />
```

Submission sets `<pac:action>` to `<result="text/xml">`.

In the example source, this example has `serviceId=JOBFLOW_CUSTOM_Sample_Service`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?JOBFLOW_CUSTOM_Sample_Service`

Job data file list (serviceId=data)

This example demonstrates how to manage job data files in a list table. It implements a Delete button that executes a delete action on selected files. In the example source, the builtin action `/platform/service/getDataListAction.action` is supported to manage all job data.

In the example source, this example has `serviceId=data`.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=data`

Convert a string to a json object(serviceId=jsonStrToObj)

This example demonstrates how to convert a string to a json object and display the value in a text field.

Access the example at this URL: `http://pac_host:8080/platform/service/pageFactoryAction.action?serviceId=jsonStrToObj`

Integrate a custom page into Platform Application Center

To integrate a customized page into Platform Application Center, edit `PAC_TOP/gui/conf/navigation/pmc_tree.xml` and add a node:

```
<TreeItem id="jobsTreeSource">
  <Nodes>
    <Node id="workload" label="@{pmc.tree.node.jobs.label}" icon="" status="" url=""
    layoutSourceId="" tabGroup="" highlightTabId="">
      ...
    <Node id="SDKExamples" label="SDK Examples" icon="" status="" url="/platform/
    service/pageFactoryAction.action?serviceId=menu" layoutSourceId="" tabGroup=""
    highlightTabId="" />
  </Node>
  ...
</Nodes>
</TreeItem>
```

The following attributes must be specified:

Name	Description	Required
ID	Identifies a unique name in Platform Application Center.	True
Name	Name of this customized page	True
URL	Specifies the linked service. The URL must be: "/platform/service/pageFactoryAction.action?serviceId=menu" The value of service ID maps to the ID of <code><pac:service></code> defined in <code>services.xml</code> . For example: <code><pac:service id="menu"></code>	True

Page customization framework XML reference

<pac:agent>

This tag is the root tag for all the <pac:service> XML tags. It has the following attributes:

Name	Description	Required
id	Identifies a unique name for this agent within the Platform Application Center servers.	True
xmlns:pac	http://www.platform.com/pac/service/schema Binds the Platform Application Center name space.	True
Resources	Binds the i18n resource file.	False

It can include the following sub-tags:

Name	Description	Required
<pac:service>	Defines a new Platform Application Center service	False

This tag must be the root of the service tree, so it cannot be contained by any other tag.

Example:

```
<pac:agent id="hpc"
  xmlns:pac="http://www.platform.com/pac/service/schema"
  resources="resources">
  <pac:service id="formElement">
    [...]
  </pac:service >
</pac:agent>
```

<pac:spooler>

This tag describes the spooler owned by the user requesting the list of spoolers. The tag is included only if the spooler is not empty. It is optional and is the parent of the <pac:service> tag.

The <pac:spooler> tag has the following attributes:

Name	Description	Required
server	The path name under which spoolers will be created on the server	True

Name	Description	Required
ttl	Time-to-live: defines how long the spoolers will be accessible	True

<pac:service>

This tag describes a service, its options, description, and available actions. It has the following attributes:

Name	Description	Required
id	Identifies a unique name for this agent within the Platform Application Center servers.	true

It can include the following sub-tags:

Name	Description	Required
<pac:name>	Name of the service. The value will display on top of web page	true
<pac:info>	Description of the service. The message will display in front of the whole service contents	false
<pac:html>	Contains XHTML tags or other information that is displayed together with the form.	false
<pac:option>	The element that user will see in the browser and will become variables in the backend environment	false
<pac:action>	Specifies the script or command that is run when users submit the HTML form filled with the proper parameters	false

Example:

```
<pac:service id="hello_pac">
  <pac:option id="USER_NAME" label="@{userName}:" type="text" />
  <pac:action id="submit" label="@{submit}">
    PAC_TOP/service/samples/hello.sh hello_pac
  <pac:result type="text/html" />
</pac:action>
</pac:service>
```

<pac:name>

This tag describes the HTML page title string.

This tag is an extended name for the containing service node. The name appears in the list on the top of the HTML page.

This tag does not have attributes.

Example:

```
<pac: service id="hello_pac">
  <pac: name>Service Name</pac: name >
  [...]
</pac: service>
```

<pac:info>

This tag explains the purpose of a service to users.

This tag does not have attributes.

Example:

```
<pac: service id="hello_pac">
  <pac: info>Service information</pac: info >
  [...]
</pac: service>
```

<pac:html>

This tag contains XHTML tags or other information displayed with the form. Its usage is the same as <pac: info>, but it can be embedded anywhere in service.xml.


This tag does not have attributes.

Example:

```
<pac: service id="hello_pac">
  <pac: html >
    <![CDATA[
      <!-- BEGIN query result page -->
      <div id="showQueryResult" style="display: block">
        <label >
          @{requiredMsg}
        </label >
      </div>
    ] ] >
  </pac: html >
  [...]
</pac: service>
```

<pac:option>

This tag describes an option that users must specify in order to run a service. It has the following attributes:

Name	Description	Required
id	Identifies a unique name for this agent within the Platform Application Center.	True
type	Describes how the option should be presented to users. Possible values: label,text, list, radio, file, hidden, text area, time, checkbox; list ,radio and checkbox require a list of choices	True
label	Specifies a name for this option	False
defaultValue	Specifies the default value of an option	False
helpText	Specifies the help message for an option. The page displays this icon image: 	False
required	Specifies the value of option is optional or required	False
hidden	Specifies whether the option is visible or invisible	False
width	Specifies the width of an input field	False
readonly	Specifies that an input field should be read-only	False
multi	Specifies the option is a drop-down list or multiple selection	False
size	Specifies the number of visible options in a multiple selection	False
rows	Specifies the visible number of rows in a text area	False
cols	Specifies the visible width of a text area	False

Name	Description	Required
<i>html_event</i>	<p>Specifies an HTML event to trigger when the control is used.</p> <p>Supported events:</p> <ul style="list-style-type: none"> • onabort • onblur • onchange • onclick • ondblclick • onerror • onfocus • onkeydown • onkeypress • onkeyup • onload • onmousedown • onmousemove • onmouseout • onmouseover • onmouseup • onreset • onresize • onselect • onsubmit • onunload 	False

It can include the following sub-tags:

Name	Description	Required
<code><pac:option></code>	Specifies an optional tag if the parent <code><pac:option></code> requires a list of choices, like <code><pac:list></code> , <code><pac:radio></code> and <code><pac:checkbox></code>	False

`<pac:option>` can be contained in another `<pac:option>` tag, as an optional tag if the parent `<pac:option>` requires a list of choices.

If the `type` attribute is equal to one of the following, it is propagated in the final HTML code:

- `label`: add plain text to your submission form
- `text`: add fields to collect user input
- `list`: allow users to select items from a predefined list
- `radio`: add a set of exclusive choices as radio buttons
- `file`: upload local files
- `hidden`: add hidden fields
- `textarea`: add a field for multiline text input or display
- `time`: add fields to specify date and time

- checkbox: add fields with a yes/no value as checkboxes
- rfb: select files from the server

Examples:

A label option:

```
<pac:option id="label1" type="label" helpText="this is a label" />
```

A text option:

```
<pac:option id="DEPARTMENT" label="this is a text box"
type="text" width="25%" required="true">
    test
</pac:option>
```

A drop-down list option:

```
<pac:option id="cpu" label="Your CPU type" type="list">
<pac:option id="i386">386</pac:option>
<pac:option id="i586">Pentium</pac:option>
<pac:option id="ia64">Itanium</pac:option>
</pac:option>
```

A multi-selection option:

```
<pac:option id="cpu" label="Your CPU type" type="list" multi="true" size="4">
<pac:option id="i386">386</pac:option>
<pac:option id="i586">Pentium</pac:option>
<pac:option id="ia64">Itanium</pac:option>
</pac:option>
```

A radio box option:

```
<pac:option id="RADIO" label="this is a radio box:" type="radio">
    <pac:option checked="true">Y</pac:option>
    <pac:option>N</pac:option>
</pac:option>
```

A file box option:

```
<pac:option id="INP_INPUT_FILE" label="this is a file uploader" type="file"/>
```

A hidden option:

```
<pac:option id="hidden" label="hidden label" type="hidden">
    1
</pac:option>
```

A time option:

```
<pac:option id="timepicker" label="this is a time picker" hidden="false" type="time" />
```

A time option:

```
<pac:option id="timepicker" label="this is a time picker" hidden="false" type="time" />
```

A text area option:

```
<pac:option id="PCC_LSF_ORG_NOTE" label="this is a text area" type="textarea"
cols="18" rows="3" />
```

A checkbox option:

```

<pac:option id="CHECKBOX_1" label="this is a check box" type="checkbox">
  <pac:option label=" 111">1</pac:option>
  <pac:option label=" 222">2</pac:option>
</pac:option>

```

An option with an HTML event:

```

<pac:option id="QUEUE" label="Queues" type="list" onchange="alert(' You select '+this.value) ">
  <pac:action id="queues">
    <![CDATA[
      SGUI_CONFDIR/appl icati on/optio ns/queue. sh normal
    ] ]>
  </pac:action>
</pac:option>

```

<pac:action>

This tag describes the action (for example, a command) that is executed on the underlying operating system. Its child node contains a command or a script to be executed by Platform Application Center when users click a Submit button in the request form.

<pac:action> has the following attributes:

Name	Description	Required
id	Identifies a unique name for this agent within the Platform Application Center.	True
execute	If the value is true, the action will be executed on the server and the output will be embedded in the page.	False
label	The label text is placed on the button of the form used to proceed with the submission.	False
name	The value maps to a Java action which the action will forward to.	False
hidden	If true, the action is interpreted as an Ajax call.	False

Name	Description	Required
result	<p>Declares which kind of output the command is supposed to provide (plain text, HTML, XML).</p> <p>The default is <code>text/html</code>.</p> <p>If the result is <code>text/plain</code>, script output returns plain text, which will be preformatted in the page result.</p> <p>If the result is <code>text/html</code>, script output must return well-formed HTML.</p> <p>If the result is <code>text/xml</code>, script output must return well-formed XML output. Output containing <code><pac: ></code> tags are parsed again then the HTML page is generated.</p>	False
CDATA	<p>Character data containing the Linux bash shell script for the page action.</p>	True

It can include the following sub-tags:

`<pac: call >`

Used within `<pac: action>` to execute an action within an action.

`<pac: call >` has the following attributes:

Name	Description	Required
id	<p>Specifies the ID of the action to execute. The action must have been previously defined.</p>	True

`<pac: action>` can be triggered the following ways:

- **Submit button:**

If `id="submit"`, and the Submit button is clicked, the form is submitted in the current service definition, and the shell script inside `<pac: action>` is executed. The result is displayed in the next page. For example:

```
<pac: service id="flow_it10">
  <pac: option id="VERSION" type="text" value="1.0" />
  <pac: action label="Submit" id="submit" result="text/xml">
    <![CDATA[
      echo "<div>"
      pwd
      env > a.log
      now=`date +%Y-%m-%d`
      echo $VERSION
      echo "</div>"
    ] ]>
  </pac: action>
```



```
</pac: service>
```

- **Self-executing page:**

If `id` is not "submit", and `execute="true"`, the script inside `<pac: action>` is executed when the page defined by the service is loaded for the first time. The result is displayed in the next page. For example:

```
<pac: service id="viewsource">
  <pac: info>services.xml</pac: info>
  <pac: action execute="true" result="text/plain">
    <![CDATA[
      cat $GUI_CONFDIR/service/services.xml
    ] ]>
  </pac: action>
</pac: service>
```

- **Ajax call:**

If `hidden="true"`, `id` is not "submit", and `execute` is not "true", the action called by `pac_trigger_action('xxxx')` is interpreted as an Ajax call. The result of executing `<pac: action>` must be JavaScript. The result output must follow the JavaScript grammar. For example:

```
<pac: service id="fromspooler">
  <div id="msgarea"></div>
  <button type="button" onclick="pac_trigger_action
('put_my_gif_in_div');">Get Image</button>
  <pac: action id="put_my_gif_in_div" hidden="true">
    <![CDATA[
      echo "\${'msgarea'}.innerHTML='<img src=\"\$PAC_DOWNLOAD_URL?spooler=
\$PAC_SPOOLER_URI&amp;file=corp_logo.png\" alt=\"Company Logo\"/>';";
    ] ]>
  </pac: action>
</pac: service>
```

- `<pac: action>` embedded in a `<pac: option type="list">` tag:

If `id` is not "submit", and `execute` and `hidden` are not specified, the embedded `<pac: action>` provides an XML list for the `<pac: option>` tag. For example:

```
<pac: option id="QUEUE" type="list">
  <pac: action id="queues">
    <![CDATA[
      echo "<option value=\"night\">night</option>"
      echo "<option value=\"normal\">normal</option>"
    ] ]>
  </pac: action>
</pac: option>
```

- **Called in another action:**

If an action was defined, it can be used in another action with the tag `<pac: call>`. For example:

```
<pac: action id="1" hidden="true">
  <![CDATA[
    ls/tmp
  ] ]>
</pac: action>
<pac: action id="2" execute="true" result="text/plain">
  <![CDATA[
    echo "action 2"
  ] ]>
  <pac: call id="1">
</pac: action>
```

When users click a Submit button, all form input is set as environment variables for the submission script to handle. You must make sure you know what variables are exported into the execution environment when you set up your form.

The services are run in the following environment:

- The working directory is set to the newly created spooler area
- Environment variables are created for Platform Application Center parameters. Some special characters within such variable contents are parsed to avoid security problems.

The following Platform Application Center variables are exported in every execution script:

- GUI_CONFDIR: The directory for Platform Application Center configuration
- LSF_ENVDIR: The directory for LSF configuration
- PAC_SPOOLER_DIR: The directory for spoolers. Defined in repository.xml

```
<ParamConfs>
  <Confirgurat i on>
    <Reposi tory>
      <User>all</User>
      <Path>/home</Path>
    </Reposi tory>
  </Confirgurat i on>
</ParamConfs>
...
```

The script must use the Linux bash shell, and output defined results to `stdout`. `stderr` must redirect to an error page with an error message.

Examples:

- Create a dynamic list option:

```
<pac: opti on id="QUEUE" label="@{selectTag}" type="list">
  <pac: acti on id="queues">
    default="normal"
    bqueues -w | awk -v select="$default" ' $0 !~ /QUEUE_NAME/ {if ($1 == select)
    {print "<option value=\""$1\"\" selected=\"selected\">"$1"</option>"} else print
    "<option value=\""$1\"\">"$1"</option>"}'
  </pac: acti on>
</pac: opti on>
```

- Forward to a Java action:

```
<pac: acti on id="submit" name="LicenseOverviewAction" label="submit">
  <pac: resul t type="text/html" />
</pac: acti on>
```

<pac:result>

This tag specifies what kind of output Platform Application Center should expect from an action. Valid output is either text or HTML. The result of the action is parsed by an XHTML parser and returned as XHTML to the browser.

<pac: resul t> has the following attributes:

Name	Description	Required
type	Defines the kind of output for the action	True

Example:

```
<pac: action id="submit" label="submit job" >
  PAC_TOP/service/samples/jobStatusApp.xml
<pac: result type="text/html" />
</pac: action>
```

<pac:spooler-info>

This tag changes the name of the spooler identified by the `uri` attribute to the `info` attribute. This allows listing the spooler name in a user-friendly fashion.

<pac: spool er- i nfo> has the following attributes:

Name	Description	Required
uri	Specifies the spooler to remove	True
info	Specifies the spooler alias	True

When you define an alias for the job directory with <pac: spool er- i nfo>, the alias is displayed in the Job Data page instead of the job directory path. You can see this in:

- Jobs > Data By Job, Job Directory column
- Jobs > Data By Flow, Job Flow Directory column

<pac:reset-spooler>

This tag resets the time-to-live of the spooler identified by the `uri` attribute.

<pac: reset - spool er> has the following attributes:

Name	Description	Required
uri	Specifies the spooler to reset	True
ttl	Specifies the new time-to-live. This value can also be negative, thus aging the spooler time-to-live.	True

<pac:go-to-spooler>

This tag redirects to the job directory page specified by the `id` attribute (`id="\${PAC_SPOOLER_URI}\/`). Use it during job submission.

<pac: go- to- spool er> has the following attributes:

Name	Description	Required
id	Specifies the job directory page to redirect to	True

Example:

```
<pac: spool er ttl="1d" >
  <pac: service id="test" >
    <pac: action execute="true" >
      <![CDATA[
        echo "this is test log" > test.log
```

```

    ] ]>
  </pac: action>
  <pac: action label="Submit" id="submit" result="text/xml">
  <![CDATA[
    echo "<pac: go-to-spooler/>"
  ] ]>
  </pac: action>
</pac: service>
</pac: spooler>

```

<pac:flow-submit>

This tag has been deprecated since version 8.0.2. Use <pac:flow-details> instead.

<pac:flow-details >

This tag redirects to the Flow Details page. Use it after you triggered a flow to display results. Only available when Platform Process Manager is installed.

<pac:flow-details> has the following attributes:

Name	Description	Required
id	ID of the job. Use the variable \${JOBID}.	True
redirect	URL of the details page. Use the variable \$JOB_DETAIL_URL?jobId=\${JOBID} for the URL.	True

Example:

```

<pac: service id="test">
  ....
  <pac: action label="Submit" id="submit" result="text/xml">
  <![CDATA[
    echo "<pac: flow-details id='\${FLOWID}'\n"
    redirect=' $JOBFLOW_DETAIL_URL?insId=${FLOWID}' />"
  ] ]>
  </pac: action>
</pac: service>

```

<pac:job-submit>

This tag has been deprecated since version 8.0.2. Use <pac:job-details> instead.

<pac:job-details>

This tag redirects to the Job Details page. Use it after you submitted a job to display results.

<pac:job-details> has the following attributes:

Name	Description	Required
id	ID of the job. Use the variable \${JOBID}.	True
redirect	URL of the details page. Use the variable \$JOB_DETAIL_URL?insId=\${JOBID} for the URL.	True
jobName	Descriptive name for the job.	Optional for jobs. Required for job arrays.

Name	Description	Required
bsubH	If the job is submitted with the bsub -H option, this attribute must be set to true. When a job is submitted with the bsub -H option, LSF holds the job in the Suspended state when the job is submitted.	Default value is false.

Example:

```
<pac: service id="test">
  ....
  <pac: action label="Submit" id="submit" result="text/xml">
    <![CDATA[
      echo "<pac:job-details id=\"${JOBID}\"
        redirect=' $JOB_DETAIL_URL?jobid=${JOBID}' bsubH="false" jobName=\"${JOBNAME}\""/>"
    ] ]>
  </pac: action>
</pac: service>
```

<pac:jobgroup-submit>

This tag has been deprecated since version 8.0.2. Use <pac:jobgroup-details> instead.

<pac:jobgroup-details>

This tag redirects to the Job Group Details page. Use it to display results after you submitted jobs associated with a job group.

<pac:jobgroup-details> has the following attributes:

Name	Description	Required
id	ID of the job. Use the variable \${JOBID}.	True
redirect	URL of the details page. Use the variable \$JOBGROUP_DETAIL_URL?groupName=\${groupName} for the URL.	True
bsubH	If the job is submitted with the bsub -H option, this attribute must be set to true. When a job is submitted with the bsub -H option, LSF holds the job in the Suspended state when the job is submitted.	Default value is false.

Example:

```
<pac: service id="JOBGROUP_CUSTOM_Sample_Service">
  ....
  <pac: action label="Submit" id="submit_job" result="text/xml">
    <![CDATA[
      JOBGROUPNAME="/test"
      JOB_COMMAND=`echo "$COMMAND" | awk -F": " '{ print $1}'`
      if [ "x$JOB_COMMAND"="x" ]; then
        echo "Job command was not specified " 1>&2
        exit 1
      fi
      LSF_OPT=""
      if [ "x$JOB_NAME" != "x" ]; then
        LSF_OPT="-J \"${JOBGROUPNAME}\""
      fi
      if [ "x$INP_INPUT_FILE" != "x" ]; then
        INPUT_FILE="\${INP_INPUT_FILE}\"
        LSF_OPT="$LSF_OPT -i $INPUT_FILE"
      fi
      JOBID=""
      for n in {1, 2, 3, 4, 5}; do
```

```

JOB_RESULT=`bin/sh -c "bsub -g ${JOBGROUPNAME} ${LSF_OPT} ${JOB_COMMAND}
2>&1" `
ID=`echo ${JOB_RESULT} | grep "<[0-9]*>" | sed 's/>. *//' | sed 's/^. */'`

#get the JOB ID, if it's "", then return error message
if [ "${ID}"="" ]; then
    echo "$JOB_RESULT" 1>&2
    exit 1
fi
#remember all sub job id into variable JOBIID
if [ "${JOBIID}"="" ]; then
    JOBIID="${ID}"
else
    JOBIID="${JOBIID} ${ID}"
fi
done
echo "<pac:jobgroup-details id=\"${JOBIID}\" groupName=\"${GROUPNAME}\"
redirect=' $JOBGROUP_DETAIL_URL?groupName=${GROUPNAME}' bsubH=' false' />"
] ]>
</pac:action>
</pac:service>

```

Built-in JavaScript functions

The following JavaScript functions are built-in functions. You can use them directly in services.xml.

pac_trigger_action('actionId', opt1, opt2, ...)

Ajax request that executes an action in the current service, and the result of the action is executed by JavaScript.

Result output must follow JavaScript grammar.

Parameters

Parameter	Description
actionId <String>	ID of the action element.
opts <Array>	Optional parameters to send to the action.

Example

```

<pac:info>
  <![CDATA [
    <script type="text/JavaScript">
      function getInfo(){
        var n = document.getElementById("name").value;
        var a = document.getElementById("gender").value;
        var arr = ['name='+n, 'gender='+a];
        pac_trigger_action('blah', arr);
      }
    </script>
  ] ]>
</pac:info>
<pac:option id="userInfo" type="textarea"></pac:option>

<pac:action id="blah" hidden="true">
  <![CDATA [
    name=$name;
    gender=$gender;
    msg=`echo 'Name: ' $name ' Gender: ' $gender`;
    echo "\$('userInfo').innerHTML=' $msg';"
  ] ]>
</pac:action>

```

pac_update_element('\${id}', text1, text2, ...)

Populates with text the innerHTML element of the specified ID.

Parameters

Parameter	Description
id <String>	ID of the element.
text <String>	Text value.

Example

```
<pac: info>
  <![CDATA [
    <script type="text/JavaScript">
      function getJsonInfo() {
        var n = document.getElementById("name").value;
        var a = document.getElementById("age").value;
        var jsonString="{ 'name': '" + n + "', 'age': '" + a + "' }";
        var jsonObj = pac_toJSONObject(jsonString);
        alert(jsonObj.name);
        alert(jsonObj.age);
        pac_update_element(' $(jsonData)', ' name:' + jsonObj.name + ', age:' +
jsonObj.age);
      }
    </script>
  ] ]>
</pac: info>
<pac: option id="jsonData" type="textarea"></pac: option>
```

pac_toJSONObject(obj)

Converts a JSON text to an object.

Parameters

Parameter	Description
obj <String>	The JSON style string.

Returns

The JSON object.

Example

Refer to the example in `pac_update_element()`.

CDATA section

CDATA is text data that should not be parsed by the XML parser. For example, characters like "<" and "&" are not valid in XML elements. CDATA can include JavaScript or a Linux shell script as code segments. It exists within option tags `<pac: action>`, `<pac: info>` and `<pac: html >`.

Built-in environment variables

You use built-in environment variables when you want the content of the variable to be automatically interpreted by Platform Application Center.

- PAC_SPOOLER_DIR: The directory for spoolers. Defined in repository.xml :

```
<ParamConfs>
  <Configuration>
    <Repository>
      <User>all</User>
      <Path>/home</Path>
    </Repository>
  </Configuration>
</ParamConfs>
```

```
...
Example:
<pac:spooler server="{PAC_SPOOLER_DIR}">
...
```

- PAC_SPOOLER_URI: The current directory in which spoolers are running. Example:

```
<pac:option id="INP_INPUT_FILE" type="file" />
<pac:action label="Submit" id="submit">
  <![CDATA[
    if [ -z $INP_INPUT_FILE ]; then
      echo "You must upload a local file."
    else
      echo "You uploaded this file: <br/>"
      echo "SPAC_SPOOLER_URI/$INP_INPUT_FILE"
    fi
  ] ]>
</pac:action>
```

- PAC_DOWNLOAD_URL: The URL for displaying an image or downloading a file.

Example:

```
<pac:action id="put_my_gif_in_div" hidden="true">
  <pac:action label="Submit" id="submit">
    <![CDATA[
      echo "\${'msgarea'}.innerHTML='<img
      src=\"\$PAC_DOWNLOAD_URL?spooler=\$PAC_SPOOLER_URI&amp;file=x.png\" alt=
      \"Platform Computing logo\"/>';"
    ] ]>
  </pac:action>
```

- JOB_FLOW_DETAIL_URL: URL for the Flow Details page, to use after a flow is triggered.

Example:

```
echo "<pac:flow-details id=\"\${FLOWID}\" redirect=' \$JOBFLOW_DETAIL_URL?insId=\$
{\$FLOWID}' />"
```

- JOB_DETAIL_URL: URL for the Job Details page, to use after a job is submitted.

Example:

```
echo "<pac:job-details id=\"\${JOBID}\" redirect=' \$JOB_DETAIL_URL?jobId=\${JOBID}' />"
```

- JOBGROUP_DETAIL_URL: URL for the Job Group Details page, to use after a job is submitted to a job group.

Example:

```
echo "<pac:jobgroup-details id=\"\${JOBID}\" groupName=\"\${GROUPNAME}\"
redirect=' \$JOBGROUP_DETAIL_URL?groupName=\${GROUPNAME}' />"
```

XML internationalization (i18n)

The Platform Application Center page customization framework supports an internationalization (i18n) feature for services.xml. The i18n resource file is defined in resource[_xx_xx].xml under PAC_TOP/service.

Example:

```
<pac:resource name="Monday">Monday</pac:resource>
```

The resource name in the resource[_xx_xx].xml file maps to the same one surrounded by @{} in the services.xml file.

Example:

```
<pac:option id="@{Monday}">@{Monday}</pac:option>
```

Create Custom Pages with the SDK

Platform Application Center Reference

perfadmin

Administer the LSF Reports (PERF) services.

Synopsis

perfadmin start *service_name* | **all**

perfadmin stop *service_name* | **all**

perfadmin [**list** | **-h**]

Description

Caution:

This command can only be used by LSF administrators.

Starts or stops the PERF services, or shows status.

Run the command on the PERF host to control the following PERF services: loader controller (plc), job data transformer (jobdt), and data purger (purger). Run the command on the Derby database host to control the Derby database service (derbydb).

If PERF services are controlled by EGO, let the EGO service controller start and stop the PERF services.

Options

start *service_name* | **all**

Starts the PERF services on the local host. You must specify the service name or the keyword **all**. Do not run this command on a host that is not the PERF host or Derby database host, you should only run one set of services per cluster.

stop *service_name* | **all**

Stops the PERF services on the local host. You must specify the service name or the keyword **all**.

list

Lists status of PERF services. Run this command on the PERF host or Derby database host.

-h

Outputs command usage and exits.

Output

Status information and prompts are displayed in your command console.

SERVICE

The name of the PERF service.

STATUS

STARTED

	Service is running.
STOPPED	
	Service is not running.
UNKNOWN	
	Service status is unknown. The local host may not be the PERF host or Derby database host.
HOSTNAME	
	Name of the host.
WSM_PID	
	Process ID of the running service.

See also

- `pmcadmin` command: administer the Platform Application Center
- `perfsetrc` command: enable automatic startup of PERF services on a host
- `perfremoverc` command: disable automatic startup of PERF services on a host

perfremoverc.sh

Prevents automatic startup of the LSF Reporting (PERF) daemons on a UNIX host.

Synopsis

perfremoverc.sh

Description

This is an administrative command. You must be logged on as `root` to issue this command.

Prevents automatic startup of PERF daemons on a UNIX host when a system reboot command is issued. After this script/command is issued, PERF daemons no longer start automatically if the host gets rebooted. In such a case, you must manually start daemons after the host has started up.

Removes the file `perf` created in the system startup directory by `perfsetrc.sh`.

Run the command on the PERF host to control the following PERF services: loader controller (`plc`), job data transformer (`jobdt`), and data purger (`purger`). Run the command on the Derby database host to control the Derby database service (`derbydb`).

perfsetrc.sh

Configures automatic startup of the LSF Reporting (PERF) daemons on a UNIX host.

Synopsis

perfsetrc.sh

Description

This is an administrative command. You must be logged on as `root` to issue this command.

Configures a UNIX host to allow automatic startup of PERF daemons on the machine when a system reboot command is issued. Creates the file `perf` under the system startup directory.

For ease of administration, you should enable automatic startup. This starts PERF daemons automatically when the host restarts.

If you do not configure hosts to start automatically, PERF daemons must be started manually.

Run the command on the PERF host to control the following PERF services: loader controller (`plc`), job data transformer (`jobdt`), and data purger (`purger`). Run the command on the Derby database host to control the Derby database service (`derbydb`).

pmcadmin

Administer Platform Application Center.

Synopsis

pmcadmin start | stop | list | https enable | https disable | -h

Description

This command can only be used by LSF administrators.

Always run this command on the host that runs Platform Application Center.

This command is used to administer Platform Application Center.

Options

start

Starts Platform Application Center on the local host.

stop

Stops Platform Application Center on the local host.

list

Status of the Platform Application Center service on the local host.

https enable | https disable

This command must be run as root.

- The `enable` option stops Platform Application Center, automatically generates the required SSL key, and restarts Platform Application Center.
- The `disable` option stops Platform Application Center, completes the necessary configuration to disable HTTPS, and restarts Platform Application Center.

-h

Outputs command usage and exits.

Output

Output of `pmcadmin list`:

SERVICE

WEBGUI, the name of the Platform Application Center service

STATUS

STARTED

Platform Application Center is running.

STOPPED

Platform Application Center is not running.

UNKNOWN

Platform Application Center status is unknown. The local host may not be the Platform Application Center host.

HOSTNAME

Name of the host.

WSM_PID

Process ID of the service.

PORT

Web server port.

See also

- `perfadmin` command: administer reporting services
- `pmcsetrc` command: enable automatic startup of Platform Application Center on a host
- `pmcremovec` command: disable automatic startup of Platform Application Center on a host

pmcremovec.sh

Prevents automatic startup of the Platform Application Center on a UNIX host.

Synopsis

pmcremovec.sh

Description

This is an administrative command. You must be logged on as `root` to issue this command.

Prevents automatic startup of the Platform Application Center on a UNIX host when a system reboot command is issued. After this command is issued, the Platform Application Center no longer starts automatically if the host gets rebooted. In such a case, you must manually start the Platform Application Center after the host has started up.

Removes the file `pmc` created in the system startup directory by `pmcset rc.sh`.

pmcsetrc.sh

Configures automatic startup of the Platform Application Center on a UNIX host.

Synopsis

pmcsetrc.sh

Description

Configures a UNIX host to allow automatic startup of the Platform Application Center on the machine when a system reboot command is issued. Creates the file `pmc` under the system startup directory.

This is an administrative command. You must be logged on as `root` to issue this command.

For ease of administration, you should enable automatic startup. This starts the Platform Application Center automatically when the host restarts.

If you do not configure hosts to start automatically, Platform Application Center must be started manually.

