Platform Application Center
Version 9 Release 1

Administering Platform Application Center

SC22-5396-01

Platform Application Center
Version 9 Release 1

Administering Platform Application Center

SC22-5396-01

Note

Before using this information and the product it supports, read the information in "Notices" on page 177.

First edition

This edition applies to version 9, release 1 of IBM Platform Application Center (product number 5725G88) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Overview

Differences between IBM Platform Application Center Editions

IBM® Platform Application Center(Platform Application Center) is available in two editions: Basic Edition and Standard Edition.

IBM Platform Application Center Basic Edition provides basic job submission, and job and host monitoring.

IBM Platform Application Center Standard Edition provides not only basic job submission and job and host monitoring, but also default application templates, role-based access control, reporting, customization, and remote visualization capabilities.

The following table highlights the differences between the editions.

Feature	**IBM Platform Application Center Basic Edition**	**IBM Platform Application Center Standard Edition**
Generic job submission	•	•
Job and host monitoring	•	•
Built-in reporting		•
Specific application job submission		•
Built-in application templates		•
Custom applications and templates		•
Custom job data repositories		•
SDK to extend application templates and create custom pages		•
Notifications when job status changes		•
Branding customization capabilities		•
Support for remote 2D/3D visualization		•
Web Services API		•
Role-based access control		•
LSF® Scheduler Dashboard		•
Integration with IBM Platform Process Manager for web-based flow submission, monitoring and control		•
Integration with IBM Platform Analytics reports for advanced web-based analysis and reporting on LSF data		•

Feature	IBM Platform Application Center Basic Edition	IBM Platform Application Center Standard Edition
Integration with IBM Platform License Scheduler for web-based license usage monitoring	•	•

Open Platform Application Center for the first time

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

About this task

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.

Procedure

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:

     pmcadmin list
2. Open Platform Application Center in a supported Web browser.

   `http://pac_host:8080/platform`
3. Log on with your user name and password.

   To see administrator features in Platform Application Center, you must log in as an administrator.
4. If the log on failed or the URL could not be found, 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.

   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

About this task

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.

Procedure

1. For Internet Explorer, 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.

View the Platform Application Center version

Procedure

To view the Platform Application Center version and the build number, use the command **pmcadmin -V**.

Chapter 2. Jobs and Job Data

Job directory, repository, 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.

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.

Note: If root squash is used at your site, permissions on the job repository directory must be set to 777.

Job Directory

The job directory is the directory in which job data is located. The job directory is created automatically when the user submits the job.

By default, if the job is submitted with a job name, the job directory is named with the job name, the timestamp, and five random characters in the form: *jobname_timestampfiverandomcharacters*.

By default, if the job is submitted without a job name, the job directory is named with the application name, the timestamp, and five random characters in the form: *applicationname_timestampfiverandomcharacters*.

The administrator can configure an application template to allow users to specify a custom job directory. In such a case, the job directory is the directory specified by the user.

For more details on custom job directories, see "Configure an application template to allow users to specify a custom job directory" on page 32.

Configure Repository.xml

About this task

This file defines the default repository location. If the application definition file (`application_name.xml`) does not have the <jobdata> element present, Platform Application Center uses `Repository.xml` to determine the repository location.

The repository for each user is `repositorylocation/username`.

Procedure

1. Edit `$GUI_CONFDIR/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.

   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.

3. 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.

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.

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

   For example:

   `$GUI_CONFDIR/application/draft/applicationname/applicationname.xml`

2. Add the <jobdata> element is nested under the <repository> element, as shown:

```
<agent>
 <repository>
 <jobdata>
 ...
 </jobdata>
 </repository>
</agent>
```

3. The <jobdata> 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 <jobdata> element, the last location specified is used.

   Syntax:

   <location path="repository_location" user="user_name ..." />

   Special Keywords:

   - all: If "all" is the user, it indicates all users not explicitly defined elsewhere within the <jobdata> element.

4. Restart Platform Application Center.

Example:

```
<jobdata>
<location path="/data" user="all" />
<location path="/abc" user="user1" />
<location path="/xyz" user="user2,user3" />
</jobdata>
```

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 `/data`.

Shared Directories

An administrator can set up shared directories for users. Shared directories for users 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.

Procedure

1. Edit `$GUI_CONFDIR/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.

Example

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>
  <Configuration>
        <ShareDirectory>
                 <Path>/shared/common/</Path>
        </ShareDirectory>
        <ShareDirectory>
                 <Path>/shared/testing/$USER</Path>
        </ShareDirectory>
  </Configuration></ParamConfs>
```

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

- user1: `/shared/common/` and `/shared/testing/user1/`
- user2: `/shared/common/` and `/shared/testing/user2/`

Enable the 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.

In order for you to view and manage the remote job directory though Platform Application Center, the submission script must generate the `.rspooler` file in the job directory. The `.rspooler` file in the job directory specifies to Platform Application Center the remote directories used by the job on execution hosts.

The `.rspooler` file has one remote job directory per line and can have multiple lines. The file format is:

remote_host_name:/*remote_directory*

For example:

```
host1:/tmp/usr1/job1directory
host2:/stratch/dev/shared/job2directory
```

Once the `.rspooler` file is generated in the job directory, if the parameter DELETE_REMOTE_DIR=y in `$GUI_CONFDIR/pmc.conf`, you will be able to view and delete any remote job directories through Platform Application Center, and automatic data purging will also be performed on the remote job directories.

For example, you can edit the ABAQUS application template 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 $NCPU_OPT"
echo "#BSUB $OUTPUT_OPT"
echo "#BSUB $MEMARC_OPT"
echo "#BSUB $ABQ_LIC"
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_CONFDIR}/application/job-result.sh
```

Job data purging

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. You can define how long to keep job data and job information for Done and Exited jobs before the data is automatically purged.

When data is purged

Data is purged daily according to the time defined with the parameter AutoPurgeTime in `$GUI_CONFDIR/pmc.conf`. The default is daily at 3:00 am.

How data purging works

Platform Application Center runs the purger task daily, according to the configurable parameter AutoPurgeTime in `$GUI_CONFDIR/pmc.conf`. You can view the scheduled time for automatic purging of a job's data by selecting **Job Data** > **By Job** and looking at the **Purge Time** column.

The purger task checks all job data to see whether it is expired. If it is expired, the purger removes the job data.

Data expiry time is calculated as:

```
job data expiry time = job completion time + FINISH_JOB_TIME_TO_LIVE
```

The parameter in FINISH_JOB_TIME_TO_LIVE in `$GUI_CONFDIR/pmc.conf` is a system-wide parameter for all applications. It defines how many days job data and job information is kept in the system for all Done and Exited jobs before it is automatically purged. The default is 14 days.

This parameter takes effect ONLY if an application does not have a defined time-to-live (`repository ttl` parameter) in its xml application definition file (published applications: `$GUI_CONFDIR/application/published/`*`application_name`*`/`*`application_name`*`.xml`, unpublished applications: `$GUI_CONFDIR/application/draft/`*`application_name`*`/`*`application_name`*`.xml`), or a job is submitted outside of IBM Platform Application Center (such as with the LSF command line).

If there is a defined `repository ttl` parameter defined for an application, the application-specific setting overrides the system FINISH_JOB_TIME_TO_LIVE setting.

If there is a defined `repository ttl` parameter for an application, data expiry time is calculated as follows for that application:

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

How parameter settings affect what users see

The settings of the parameters FINISH_JOB_TIME_TO_LIVE and `repository ttl` for an application affect what users see in the **Jobs** tab, **Jobs** page, and in the **Job Data** > **By Jobs** page. For example, if you define that job data and job information is to be kept for 10 days, users will be able to see job information and job data for all pending and running jobs, and also for jobs that have ended in the past 10 days. Job data and job information for jobs that have an end time earlier than the past 10 days will be purged and will not be visible.

Configure when automatic data purging occurs

Procedure

1. Log on to the Platform Application Center server host as root.
2. Edit the file:`$GUI_CONFDIR/pmc.conf`.
3. Set the AutoPurgeTime parameter (specify the number of hours):

   ```
   AutoPurgeTime=3
   ```
4. 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.

Configure how long to keep job information and data for an application

Procedure

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

   If the application is published:

   `$GUI_CONFDIR/application/published/`*`application_name/application_name`*`.xml`

   If the application is unpublished:

   `$GUI_CONFDIR/application/draft/`*`application_name/application_name`*`.xml`
3. Edit the file, find and add the ttl attribute to the `<repository>` element.

   For example: `<repository  ttl="60">`

   In this example. the application job data will be kept for 60 days after the job completes.
4. 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.

Add a custom column to the Jobs page

You can add custom columns to the **Jobs** page for jobs, job array elements, jobs that belong to job groups, and flows with the configuration file `$GUI_CONFDIR/customJobColumn.xml`. You create a script that uses the job IDs and job types passed from Platform Application Center, you pass predefined input keywords to your script to get the values from Platform Application Center, perform any required calculations, and generate output in XML format. You can

then choose to display the columns from the **Jobs** page, by selecting the **Options** button and choosing your new columns to display.

Step 1: Configure the custom column in customJobColumn.xml

You add custom columns to the **Jobs** page for jobs, job array elements, jobs that belong to job groups, and flows with the configuration file `$GUI_CONFDIR/customJobColumn.xml`.

Procedure

1. Edit the file `$GUI_CONFDIR/customJobColumn.xml`.
2. Add a `<JobColumn>` section for your custom column.

   In the following example, we add the column Application Status, taking as input the predefined values for memory usage and job priority:

```
<JobColumnConf>
<JobColumnGroup id="customColumn" name="">
<JobColumn id="appStatus">
    <Command type="CLI">$GUI_CONFDIR/application/getstate.sh/Command>
    <ColumnName>Application Status</ColumnName>
    <Tooltip>Application state reported by application library </Tooltip>
    <InputParams> MEMORY;PRIORITY  </InputParams>
  </JobColumn>
</JobColumnGroup>
</JobColumnConf>
```

3. Save the file.
4. Validate the XML file against the xsd file `$GUI_CONFDIR/customJobColumn.xsd`.

customJobColumn.xml Reference

Configuration file to add custom columns to the Jobs page. You can validate agains the xsd file `$GUI_CONFDIR/customJobColumn.xsd`.

File Format:

```
 <JobColumnConf>
<JobColumnGroup id="customColumn" name="">
  <JobColumn id="myid">
    <Command type="CLI">.../Command>
    <ColumnName>...</ColumnName>
    <Tooltip>... </Tooltip>
    <InputParams> KEYWORD;KEYWORD;KEYWORD; </InputParams>
  </JobColumn>
  <JobColumn id="myid2">
    <Command type="CLI">.../Command>
    <ColumnName>...</ColumnName>
    <Tooltip>... </Tooltip>
    <InputParams> KEYWORD;KEYWORD; </InputParams>
  </JobColumn>
   ...
 </JobColumnGroup>
</JobColumnConf>
```

Example:

```
<?xml version=<i>"1.0" encoding="UTF-8">
<JobColumnConf xmlns:xsi ="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation ="customJobColumn.xsd">
 <JobColumnGroup id="customColumn" name="CustomColumn">
   <JobColumn id="concatenate">
     <Command type="CLI">$GUI_CONFDIR/application/catenate.sh</Command>
     <ColumnName>CONCATENATE MEM PRI</ColumnName>
     <Tooltip>Concatenate memory and priority</Tooltip>
```

```
    <InputParams>MEMORY;PRIORITY </InputParams>
  </JobColumn>
 </JobColumnGroup>
</JobColumnConf>
```

JobColumn:
Required. Defines the new custom column. There is one `JobColumn` element per new column.

Attributes:

Name	Description	Required

id	ID of the new column. This ID must be unique, alphanumeric, and a maximum of 32 characters.
	
	If multiple entries with the same ID are defined, Platform Application Center uses the first one defined and ignores any subsequent ones.
	
	Some IDs are reserved for system use. If a reserved ID is used, an error message is displayed in the **Jobs** page.
	
	The following are reserved IDs for system use. Do not use them as your column IDs:
	`activeGraphicJob`
	`app`
	`appTemplateType`
	`appType`
	`askedHosts`
	`command`
	`cwd`
	`deleteTime`
	`description`
	`done`
	`endTime`
	`estimatedRunTime`
	`execCwd`
	`exHosts`
	`exited`
	`extStatus`
	`fromHost`
	`fromOwnFlowDefinition`
	`graphicJob`
	`hasControlPermission`
	`id`
	`indexId`
	`infile`
	`jobGroup`
	`jobId`
	`jobName`
	`jobNotification`
	`jobStatus`
	`jobType`
	`jobUniqId`
	`mem`
	`name`
	`nthreads`
	`numProcessors`
	`outfile`
	`path`
	`pathAlias`
	`pend`
	`pgid`
	`pid`
	`priority`
	`projectName`
	`queue`
	`remainingTime`
	`run`
	`runTime`
	`startTime`
	`status`
	`submitTime`
	`susp`
	`swap`
	`type`
	`user`
		Yes

Command:
Required. Path to the script file to run to retrieve a value to populate the custom column.

The script name must be named according to UNIX file and path naming conventions.

The script can be any executable as long as it meets the required input and output.

The directory in which the script is located can be any directory accessible by the web server.

All users who will be viewing the **Jobs** page require Read and Execute permission to the directory in which the script is located. The script runs as the user who is logged on and viewing the Jobs page.

Attributes:

Name	Description	Required
type	Must always be set to type CLI.	Yes

ColumnName:
Required.

Name of the column as it will be displayed in Platform Application Center in the **Jobs** page. The name of the column must be alphanumeric with a maximum of 32 characters. Dashes (-) and underscores(_) are allowed.

Tooltip:
Optional.

Descriptive text to be displayed about the column when the user hovers on the column name. The maximum number of characters is 250.

You cannot use the special characters < > / inside the tooltip. If you need to use these characters, you must encrypt the tooltip.

InputParams:
Optional.

Values to be passed to your command script as input parameters. You specify predefined keywords and Platform Application Center passes the actual values of the keywords to your command script.

Important: Keywords must be specified in capital letters and must also be used in your script in capital letters.

Separate multiple keywords with a semi-colon(;).

Valid keywords:

Keyword	Description
APPLICATION_TYPE	Name of application template used to submit the job.
COMMAND	The job command specified during job submission.
CONSOLE_SUPPORT	Indicates whether the job was submitted with remote console support enabled.
CURRENT_WORKING_DIRECTORY	The current working directory on the submission host.
ENDED_TIME	For done or exited jobs: when the job finished. For running jobs, when the job is estimated to finish. Calculated as Current® time + Time Remaining.

Keyword	Description
ESTIMATED_RUN_TIME	Estimated running time for the job specified by the user during job submission, with the LSF command **bsub -We**.
EXECUTION_HOST	The name of one or more hosts on which the job is running.
EXECUTION_WORKING_DIRECTORY	Current working directory where job is running.
EXTERNAL_STATUS	External status information for a job, specified with the LSF command `bpost -d`.
INPUT_FILE	Input files specified during job submission.
JOB_DESCRIPTION	The job description assigned to the job during job submission. For flows, this is the description specified with the input variable JS_FLOW_DESCRIPTION.
JOB_DIRECTORY	Directory in which input and output files are located.
JOB_ID	The job ID that LSF assigned to the job.
JOB_NAME	Name of job specified during job submission.
JOB_NOTIFICATION	Indicates whether the job was submitted with notification for job status changes enabled.
JOB_STATUS	Current status of job.
JOB_TYPE	Type of workload. Possible values: are job, job array, or flow.
MEMORY	Total resident memory usage in MB, of all processes in a job. For host-based resource usage, the total resident memory usage of all processes in a job running on a host.
OUTPUT_FILE	Output files created by the job.
PENDING_REASON	The reason the job is in the Pending state.
PRIORITY	Priority specified for the job during job submission.
PROCESS_GROUP_ID	Currently active process group ID in a job.
PROCESS_ID	Currently active processes in a job.
PROFILE_NAME	The application profile the job was submitted to.
PURGE_TIME	Date and time job data was automatically purged by the system.
PROJECT_NAME	The project the job was submitted from.
QUEUE	The name of the job queue to which the job was submitted.
REMAINING_TIME	Time remaining to complete the job. Calculated as Estimated Runtime – Runtime. Available only for jobs submitted with an estimated runtime with the LSF command **bsub -We**.
REQUESTED_HOST	All the resource requirement strings you specified during job submission.
REQUIRED_PROCESSOR_NUMBER	Number of processors specified as required to run the job during job submission.
RUN_TIME	How long the job has been running.
STARTED_TIME	When the job started running.
SUBMISSION_HOST	Host from which the job was submitted.
SUBMISSION_USER	The user who submitted the job.
SUBMITTED_TIME	When the job was sent to the system.
SWAP	Total virtual memory usage in MB, of all processes in a job. For host-based resource usage, the total virtual memory usage of all processes in a job running on a host.
THREAD_NUMBER	Number of currently active threads of a job.

Step 2: Create your script to populate your custom column

Create your script to perform calculations to populate your custom column. It is recommended to use one script per custom column for easier debugging. An example script `catenate.sh` is provided in `$GUI_CONFDIR/application/`.

Create your script

Procedure

1. Write your script.

   Ensure your script meets the script requirements.
2. Test and debug your script.

   It is important your script work properly before you integrate it with Platform Application Center.
3. Copy your script to a location that is accessible to the web server.

   Ensure that the directory is readable and executable by all users who will be viewing the **Jobs** page.

Script requirements

Script name

The script name must be named according to UNIX file and path naming conventions.

Script type

The script can be any executable as long as it meets the required input and output.

Directory in which script is located: The directory in which the script is located can be any directory accessible by the web server.

All users who will be viewing the **Jobs** page require read and execute permission to the directory in which the script is located. The script runs as the user who is logged on and viewing the **Jobs** page.

Input

Required environment variables

JOB_IDS and JOB_TYPES are environment variables that must be passed to the command script. These must be called within your script.

For example:

```
ids=`echo $JOB_IDS | tr ';' ' '`
types=`echo $JOB_TYPES | tr ';' ' '`
```

JOB_IDS

The ID corresponds to jobs, job flows, and job array elements. For job groups, only the IDs of jobs within the group are returned. For job arrays, the ID of array elements is returned. IDs are separated with a semi-colon (;).

Example of IDs returned for jobs and job flows.

```
JOB_IDS="102;103;105"
```

Example of IDs returned for job arrays (*job ID_index*). For example, for job array 303, and elements 6, 7, 8:

```
JOB_IDS="303_6;303_7;303_8"
```

JOB_TYPES

Items returned in JOB_TYPES indicate whether the ID returned is a job, a job belonging to a job group, job array element, or a flow. Types returned are JOB, ARRAY, GROUP, FLOW.

The types correspond to the JOB_IDS returned and are returned in the same order and number as the corresponding ID.

For example, ID 102 is a job, 103 is a job within a job group, 105 is a flow, and the other IDs are for job array elements. The environment variables are set by Platform Application Center in this way:

```
JOB_IDS="102;103;105;303_6;303_7;303_8"
JOB_TYPES="JOB;GROUP;FLOW;ARRAY;ARRAY;ARRAY"
```

Input parameters

Input parameters passed to the command script are predefined keywords recognized by Platform Application Center and are defined in the `$GUI_CONFDIR/customJobColumn.xml` file with the element `<InputParams>`. Keywords must be used within the script in capital letters.

Output

The output of the command must be to `stdout`. You must handle every single job ID that is returned by Platform Application Center, even if you do not have values to attach to those jobs. Return a dash (-) for job IDs to which no values apply.

Output must be in the following XML format. You can validate against the xsd file `$GUI_CONFDIR/customJobColumnOutput.xsd`.

```
<JOB_COLUMNS>
<JOB_COLUMN JOB_ID="job_ID" JOB_TYPE="job_type">
your_value
</JOB_COLUMN>
</JOB_COLUMNS>
```

There must be one XML `<job>` element returned for each job. For example:

```
<JOB_COLUMN JOB_ID="45657" JOB_TYPE="FLOW">
99</p><p>
</JOB_COLUMN>
```

Example output for several jobs:

```
<JOB_COLUMNS>
  <JOB_COLUMN JOB_ID="45657" JOB_TYPE="FLOW">
  99
  </JOB_COLUMN>
  <JOB_COLUMN JOB_ID="90123_6" JOB_TYPE="ARRAY">
  119
  </JOB_COLUMN>
</JOB_COLUMNS>
```

Exit codes

On success, the command script must return an exit code of 0.

On failure, the command script must return any value other than 0. Error messages are displayed at the top of the **Jobs** page. If additional information is needed, look for the `customJobColumn.xml` keyword in `$GUI_CONFDIR/../logs/catalina.out`.

Step 3: Display your custom job column

Once you created the script and added your custom column to the configuration file `$GUI_CONFDIR/customJobColumn.xml`, you can display your new column in the **Jobs** page.

Procedure

1. In the **Jobs** tab, select **Jobs**.
2. Click the **Options** button to display available columns.

   Important: For performance reasons, only select the new column for display for users who require it. This is because your custom script is run every time a user displays the **Jobs** page if the new column is selected to be displayed.
3. In the **Set Columns** section, click to select your new column and click **Apply**.

   You should be able to see your new column displayed.

   If any errors occur, error messages are displayed at the top of the **Jobs** page. If additional information is needed, look for the `customJobColumn.xml` keyword in `$GUI_CONFDIR/../logs/catalina.out`.

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 `sderr` 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:

```
$GUI_CONFDIR/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 the 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

the path of the job action script or command

- **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)

By default, it is:

`1:0`

- `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)

By default, it is ALL.

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.

MultipleJobs (optional)

By default, it is true.

true means the job action is available when multiple jobs are selected in job list page.

Application (optional)

By default, it is ALL.

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.

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>
```

Chapter 3. Application Templates

About application templates

Application templates are used to integrate applications.

You can use the built-in templates to immediately submit jobs to the specific applications.

Generally, the name of the template indicates the name of the application to which jobs can be submitted.

Submission forms and submission scripts

Application templates are composed of:

- A Submission Form—The job submission form is composed of fields. Each field has a unique ID associated with it. IDs must be unique within the same form.
- A Submission Script—The job submission script uses the same IDs as the submission form to pass values to your application.

Application template states

Application templates have the following states:

State	Description
Published	When an application template has the status Published, all users that have been granted access to this template can submit jobs to the application with the job submission form.Application templates with the status Published cannot be modified. They must be set to Unpublished before any changes can be made to the template.
Unpublished	When an application template has the status Unpublished, it is not visible to any users. It is only visible to the administrator. The application template can be modified and customized as desired. When an application template that has the status Published is set to Unpublished, it is no longer available to any users.

Customizing application templates

You can customize application templates by adding or removing fields, rearranging the order of fields (through drag and drop), and entering default values for fields.

You can also change field names and add help text for fields.

In addition, you can create hidden fields - fields that are only visible to you, the administrator, but that can hold default values for the submission forms. Users cannot see hidden fields in their forms.

Creating new templates

The easiest way to create a new template is to copy another template, then modify the job submission form and script. You can then test your new template by submitting a test job.

Access permissions on new templates

Once you have tested your template, you can then set the template status to Published. By default, your template is visible and usable to all users once you set its status to Published.

Modify an application template

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.

Procedure

1. Log in to Platform Application Center as administrator.
2. Choose **Resources** > **Submission Templates** > **Application Templates**.
3. Select an unpublished application (for example, ANSYS).

   If an application is already published, select it and click **Unpublish**.
4. Choose **Modify** and make the desired changes to the application submission form and submission script.

   Whenever you add, delete, or edit input fields, remember to make the corresponding changes to the code in the submission script. Use the field ID to match fields on the form with parameters in the script.
5. Click **Submit Test Job** to make sure the application works properly.
6. Click **Save** or **Save As** to save your changes.
7. Choose **Close** to close the modify window or click **Publish** and confirm as prompted.

   Only published applications are available to users.

Prepare and publish an application template

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.

Procedure

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. Click **Publish**. Fill in the required information and click **Confirm** as prompted.

   Only published applications are available to users.

Copying application templates between clusters with the application directory

Each application has a dedicated directory for configuration files, and you may create subdirectories for other scripts and data. This allows you to copy a working application from one cluster to another simply by copying all contents of the directory.

Application directory

The $APPLICATION_TOP variable points to the application directory and you can use it in scripts and application definitions instead of hard-coding the path.

In this example, the application file is the published FLUENT application, and Platform Application Center is installed under `/usr/share/pac`. Therefore, the application directory ($APPLICATION_TOP) is:

`/usr/share/pac/gui/conf/applicaton/published`.

If the application is unpublished, the value of $APPLICATION_TOP changes to:

`/usr/share/pac/gui/conf/application/draft`.

Create a new application template

You can create a new application based on a copy of an existing application template.

Procedure

1. Log in to Platform Application Center as administrator.
2. Choose **Resources** > **Submission Templates** > **Application Templates**.
3. Select an application and choose **New**.
4. Select an existing application as a base and enter a name for the new application template.
5. Edit the template submission form and template submission script as desired in Platform Application Center.

Defining dependencies between fields

You can define a dependency between two fields so that when you choose a value in a dropdown or radio button field you override another field's settings for:

- Visibility—Whether the field is displayed or hidden.
- Default value—The default value defined for the field.
- Content—The available values in the field.

You do this in Platform Application Center by:

1. Adding the fields to your application template.
2. Writing a script for the desired behavior and testing it outside of Platform Application Center.
3. Editing the dependent field in the application template, enabling dependencies in this field, and linking this field to the field that affects it with the corresponding script file.

Field requirements

- Fields that affect other fields can only be dropdown fields or radio button fields.
- Fields that affect other fields must be located in the application template before dependent fields.
- If the field affected by another field is an input file field, only input files from the web server can be selected. Selection of local input files is not supported.

Script requirements

The script file must:

- Be an executable. Can be written in any scripting language.
- Be tested to ensure that it works before it is used in Platform Application Center.
- Take as input value the environment variable indicating the field ID of the field that affects another field. The variable name is the field ID.
- Output a fixed-format XML message through `stdout`. For example:

```
echo "<field-control visible=\"true\" type=\"content\">"
echo "<option value=\"1\" selected=\"true\" /> "
echo "<option value=\"2\" />"
```

- Be located in your UNIX home directory or in a shared directory accessible by the Platform Application Center web server.

Script format

```
<field-control visible="true|false" type="content|default">
<option value="value" selected="true | false"/>
     ...
</field-control>
```

`<field-control>` attributes:

Name	Description	Valid values
visible	Required. Specifies whether the field is displayed or hidden. Specify true if the field is to be displayed in the submission form, false if the field is to be hidden.	true | false
type	Required. Specifies whether the script changes the content of the field or only the default value.	content | default

`<option>` attributes:

Name	Description	Valid values
value	Required. Specifies the value that this option contains.	Alphanumeric values.
selected	You can omit this attribute for options that do not contain the default value.Specifies whether this value is the default value for the field.Specify true if this value is the default value for the field, false if it is not.	true | false

Examples

When a specific queue is selected, a different number of CPUs is displayed

When a user selects the queue NORMAL, the number of CPUS available in the CPUS field is 1, 2, 4. The default number of CPUS selected is 1.

When a user selects the queue PRIORITY, the number of CPUS available in the CPUs field is 2, 6, 8, 12. The default number of CPUS selected is 6.

The ID of the queue field is QUEUE.

Script example:

Note that in is script, we also have a setting if any other queue is selected. If any other queue is selected, then the drop-down list of available CPUs is the number already defined for the field but the default value selected is 2.

```
#!/bin/sh

if [ $QUEUE="normal" ];then
   echo "<field-control visible=\"true\" type=\"content\">"
   echo "<option value=\"1\" selected=\"true\" /> "
   echo "<option value=\"2\" />"
   echo "<option value=\"4\" />"
   echo "</field-control>"
elif [ $QUEUE="priority" ]; then

   echo "<field-control visible=\"true\" type=\"content\">"
   echo "<option value=\"2\" />"
   echo "<option value=\"6\" selected=\"true\" /> "
   echo "<option value=\"8\" />"
   echo "<option value=\"12\" />"
   echo "</field-control>"

else
   echo "<field-control visible=\"true\" type=\"default\">"
   echo "<option value=\"2\" />"
   echo "</field-control>"
fi
```

When Batch mode is selected, display a field to select memory architecture

You have a drop-down field called MODE with the values Batch and Sequential. When a user selects Batch, you want to display the field Memory so that the user selects memory architecture.

The ID of the MODE field is S_MODE.

Script example:

In this example, if the value of the S_MODE field is Batch, then the Memory field is visible. If any other value is selected, the Memory field is not displayed.

```
#!/bin/sh

if [ x$S_MODE="xBatch" ];then
   echo "<field-control visible=\"true\">"
   echo "</field-control>"
```

```
else
  echo "<field-control visible=\"false\">"
  echo "</field-control>"
fi
```

Configure dependency between fields

About this task

Prerequisites:

- Both the dependent field and the field that affects it must have already been defined in the application template.
- Make note of the ID of the field that affects another field.
- You have a script to associate. Refer to the script requirements, format, and examples for details on creating your own script.
- The script is located in your UNIX home directory or on a shared directory accessible by the web server.

For these instructions, we are going to use the queue example, but you can apply the instructions to any script and field you choose. In the queue example:

- When a user selects the queue NORMAL, the number of CPUS available in the CPUS field is 1, 2, 4. The default number of CPUS selected is 1.
- When a user selects the queue PRIORITY, the number of CPUS available in the CPUs field is 2, 6, 8, 12. The default number of CPUS selected is 6.
- The ID of the queue field is QUEUE.

Procedure

1. Edit the application template that contains the field for which you want to configure dependencies.
   a. Choose **Resources** > **Submission Templates** > **Application Templates**.
   b. Select the application.

      If the application is already published, select it and click **Unpublish**.
   c. Choose **Modify**.

      The Modify Template window is displayed.
2. Select the dependent field and click **Edit**.

   For example, for the queue example, you would select the CPUS field.

   The Edit Field window is displayed.
3. Click **Enable Dependencies**.

   The Dependency table is displayed.
4. Configure the dependency.
   a. In the **Linked Field** column, from the dropdown list select the ID of the field upon which this field depends. This is the field that affects the current field.

      For example, for the queue example, select QUEUE.
   b. In the **This script overrides** column, check all areas that your script affects.

      You can choose any or all, depending what your script does.

      For example, for the queue example, since the script replaces the number of CPUs depending on what you select, select Content. The script also sets the default value for queues other than NORMAL and PRIORITY, so also select Default.

Note:

What you select in this column is for information purposes only so that you know what your script affects, but does not enable or disable any functionality. It is a summary for your convenience so you do not need to read your script to know what it does.

c. In the **Dependency Script** column, click **Browse** and select your script.

Your script must be located in your UNIX home directory or on a shared directory accessible to the web server.

Important:

Make sure you thoroughly test your script externally to Platform Application Center to make sure it functions as desired.

d. Click **OK**.

A dependency between the two fields is created and the script you specified is copied to your application directory.

For example, if you were modifying ABAQUS, the script would be copied to `$GUI_CONFDIR/application/published/ABAQUS`.

5. Publish your application and test your form.

Now that the dependency has been created, you can publish your application and test your submission form.

To test the Queue example, select the queue priority and see if the number of CPUs changes. Also select the queue normal and see if the number of CPUs changes. Select any other queue and see what the selected default value is.

Automatic detection and upload of included input files

When users submit jobs, they select which input files to use for the job. If the file is a file that is located on the user's desktop, the file is automatically uploaded to the job directory. If the file is located on the web server, it is either copied or linked to the job directory, depending on the setting of the parameter `ADD_SERVER_FILE_TYPE` in the file `$GUI_CONFDIR/pmc.conf`. In some cases, user input files are split into multiple files. The parent file contains include statements to the other files. You can configure automatic detection and upload of included input files.

To configure automatic detection and upload of included input files, in your application template you select the **Use include files** option in your **Input File Selection** type field and specify which keywords to search for within the files and the extension of input files in which to search.

How automatic detection of included input files works

When a user selects **Use include files** in the submission form and selects an input file, Platform Application Center scans the input file for any include keywords and displays the files that it finds. This is done recursively for all included files, until no more include keywords are found in the files. The user then decides whether all or none of the included input files are to be uploaded.

- Recursion is unlimited. The system scans included files for the configured include keywords until no more include keywords are found in the files.
- Include statements in the input files can contain absolute or relative paths. Relative paths are considered to be relative paths to the job directory.
- Subdirectories are automatically created within the job directory.

- If an include statement only contains a path to a directory, all files within that directory that match the configured file extensions in the application template are uploaded to the job directory.
- Input file names cannot contain any of the following characters:, " | / \ : < > ?

Supported patterns for include statements are:

- The include keyword must start at the beginning of a new line. For example:

  ```
  *INCLUDE "D:\mydata\file1.inp"
  ```

- Include keyword and file path on one line, separated by a space. For example:

  ```
  @INC "S:\MyGroup\MyProject\file.inp"
  @INC "S:\MyGroup\MyProjectB"
  ```

- Include keyword and file path on two lines:

  ```
  *INCLUDE
  file1.inp
  *INCLUDE
  S:\MyGroup\MyProjectB
  ```

- Include keyword and path to a folder, without naming of specific files. For example:

  ```
  *INCLUDE S:\MyGroup
  ```

How automatic upload of included files works

When the user submits the job, Platform Application Center uploads to the job directory the file specified by the user and all detected included files. Paths in the included files are automatically updated.

Example: Include statements without subdirectories

For example, you have an input file `myfile.txt` that contains the following include statements:

```
*INCLUDE C:\data\f1.txt
*INCLUDE D:\engine\f2.txt
```

Files are uploaded to the job directory in this location and the include statements in `myfile.txt` are also updated to reflect this:

job_directory/myfile.txt
job_directory/f1.txt
job_directory/f2.txt

Example: Include statements with subdirectories

For example, you have an input file `myfile.txt` that contains the following include statements:

```
*INCLUDE C:\data\f1.txt
*INCLUDE D:\engine\f2.txt
```

Inside file `f1.txt`, you have the following include statement:

```
*INCLUDE myfiles\f3.txt
```

Files are uploaded to the job directory in this location and the include statements in `myfile.txt` and `f1.txt` also updated to reflect this:

job_directory/myfile.txt
job_directory/f1.txt
job_directory/f2.txt
job_directory/myfiles/f3.txt

Example: Include statements to a directory, no files specified

For example, you have an input file `myfile.txt` that contains the following include statements to a directory but not to individual files. Within `c:\data`, you have the files `a1.txt`, and `a2.txt`.

```
*INCLUDE C:\data
```

All files that match the extension criteria defined in the application template are uploaded to the job directory. In addition, those files are searched for include statements. Files are uploaded to this location and the include statement in `myfile.txt` is also updated to reflect this:

```
job_directory/myfile.txt
job_directory/a1.txt
job_directory/a2.txt
```

How path mapping works between Windows clients and the Linux web server

For include statements referring to Windows drive mappings on user's local machines, you can configure a mapping of local Windows paths to Linux paths on the web server with the file `$GUI_CONFDIR/include_path_map.conf`.

The paths are automatically replaced in the include statements. Upload location within the job directory depends on path mappings. Files that already exist on the web server are not uploaded.

When files are not uploaded because they already exist on the web server, the files are copied or linked to the job directory, depending on the setting of the parameter `ADD_SERVER_FILE_TYPE` in the file `$GUI_CONFDIR/pmc.conf`.

The following is an example `include_path_map.conf` file mapping the local Windows path `S:\MyGroup` to the Linux path `/data/group1`, and `S:\dev` to `/home/dev`. Whenever the system finds the path `S:\MyGroup` in an include statement, it replaces it with `/data/group1`.

In the file, specify one mapping per line and use `==` to map. For example:

```
S:\MyGroup ==/data/group1
S:\dev == /home/dev
```

Example: Mapping for include statements without subdirectories

For example, your `include_path_map.conf` contains the following mapping:

```
S:\MyGroup ==/data/group1
```

You have an input file `myfile.txt` that contains the following include statements:

```
*INCLUDE S:\MyGroup\mydata\f1.txt
*INCLUDE S:\MyGroup\f2.txt
```

Files are uploaded to the job directory. The files `/data/group1/mydata/f1.txt` and `/data/group1/f2.txt` are copied or linked to the job directory. The include statements in `myfile.txt` are updated to reflect this:

```
job_directory/myfile.txt
job_directory/f1.txt
job_directory/f2.txt
```

Example: Include statements with relative paths and no mapping

For example, your `include_path_map.conf` contains the following mapping:

```
S:\MyGroup ==/data/group1
```

You have an input file `myfile.txt` that contains the following include statements:

```
*INCLUDE S:\MyGroup\f1.txt
*INCLUDE D:\data\f3.txt
```

Inside file `f1.txt`, you have the following relative include statement:

```
*INCLUDE myfiles\f2.txt
```

The file `myfile.txt` is uploaded to the job directory, `f1.txt` and `f2.txt` are copied or linked from the web server. Note that since there was no mapping for `D:\data`, file `f3.txt` is uploaded to the job directory.

```
job_directory/myfile.txt
job_directory/f3.txt
job_directory/f1.txt
job_directory/myfiles/f2.txt
```

Enable detection of included input files in your application template

To enable detection and upload of included input files, you select in the application template **Use include files** in an **Input File Selection** type field and specify which keywords to search for within the files and the extension of files in which to search.

Before you begin

- Unpublish your application template so that you can make changes.
- Make note of all the file extensions that will need to be searched for include statements. For example, .txt, .inp, etc. You will need this for configuration.
- Make a note of all keywords that are used to indicate include statements. For example, @INC, *INCLUDE, etc. You will need this for configuration.

Procedure

1. In **Resources** > **Submission Templates** > **Application Templates**, click the application template you want to modify.

   The modification window is displayed.
2. Select an **Input File Selection** field and click **Edit**, or click **Add** and add an **Input File Selection** field.
3. In the field definition, select **Allow users to use include files**.
4. In the **Include File Keywords** field, specify all the keywords used in your files to indicate include statements.

   Separate include keywords with a comma.

   For example, to enter the keywords INCLUDE and @INC, specify:

   ```
   INCLUDE,@INC
   ```
5. In the **Search in files with these extensions** field, specify the extensions of files in which the system is to search for include statements.
   - Files that do not have the specified extension will not be searched.
   - Separate multiple extensions with a comma.
   - You can use * to indicate partial matching of file names.

- For example, to specify to search in files that have the prefix 123 in their name and the extension .txt, and that all .inp files should also be searched for include statements, specify:

  ```
  123*.txt,.inp
  ```

6. Optional. In the **Help Text** field, enter any information users will need about file extensions or keywords that are searched.
7. Click **OK**.
8. If you added a new field, modify your submission script to include the new field ID.
9. Save and publish your application.
10. Test your configuration by selecting **Jobs** > **Submission Forms** and clicking on your application name.
    a. Go to the field in which you enabled automatic detection of include files. You should be able to see the checkbox **Use include files**.
    b. Click the **Add Local File** or **Add Server File** button and specify some input files that contain include statements. You should be able to see all included files listed.

Configure mapping of local Windows paths to server Linux paths for included files

You can configure a mapping of local Windows paths to Linux paths on the web server in the file `$GUI_CONFDIR/include_path_map.conf`.

Before you begin

- You must have enabled detection of included input files in your application template in your **Input File Selection** field.
- Make note of the Windows paths and the Linux paths for which you want to configure mappings.

About this task

The paths are automatically replaced in the include statements. Upload location within the job directory depends on path mappings. Files that already exist on the web server are not uploaded.

When files are uploaded, the files are copied or linked to the directory, depending on the setting of the parameter `ADD_SERVER_FILE_TYPE` in the file `$GUI_CONFDIR/pmc.conf`

Procedure

1. Edit the file `$GUI_CONFDIR/include_path_map.conf`.
2. In the file, list the Windows path, then `==`, then the Linux path.

   Each mapping must have a separate line.

   For example:

   ```
   S:\MyGroup ==/data/group1
   K:\Group2 ==/data/group2
   ```

3. Save the file.
4. Restart Platform Application Center to make your changes take effect.

Configure an application template to allow users to specify a custom job directory

When a user submits a job, the job directory is a temporary timestamped directory that is created. Job input files and output files are placed in this directory. In your application template, you can create a field to allow users to customize the location of their own job directory. When users specify their own job directory, no temporary directory is created and users can see their job directory in **Job Details**.

Before you begin

Requirements:

- Your application template must have the status Unpublished so that changes can be made.
- The type of field in the application template must be **Input Text** or **Simple Browse** and must have the ID OUTPUT_FILE_LOCATION.
- Users must have the operating system permissions of read and execute for the directories they specify. If users specify a directory for which they have no read or execute permission, job submission will fail.
- The directory specified by the user must be an absolute path on the web server.
- The environment variables $HOME and $USER can be used in the specified directory path.

Important: Custom job directories will not be purged when automatic data purge is performed by the system. Users will need to manually manage their custom job directories and files.

Procedure

1. In **Resources** > **Submission Templates** > **Application Templates**, click the application template you want to modify.

   The modification window is displayed.
2. Click **Add** to add a field and select the field type **Simple Browse** or **Input Text**.
3. In the field definition, enter the field ID OUTPUT_FILE_LOCATION.
4. Add help text for your field to indicate to users that they must have read and execute access to directories they specify or job submission will fail and any other additional information you may want to communicate.
5. Click **OK** to exit the modification window.
6. Click **Save** to save your changes.
7. Select the **Submission Script** tab and modify your submission script.
   a. Check that your submission script has the following line.

      By default, all built-in application templates have this line in the submission script.

      If you created a custom application template, check that the submission script has this line and add it to your script if needed.

      ```
      CWD_OPT="-cwd \"$OUTPUT_FILE_LOCATION\""
      ```
   b. Add **${CWD_OPT}** to the **JOB_RESULT** line right after the **bsub** command.

      For example:

```
JOB_RESULT=`/bin/sh -c "bsub ${CWD_OPT} ${LSF_OPT} ${ADVANCED_OPT} ${LIMITS_OPT} ${CUSTOMIZE_OPT} ${JOB_COMMAND} 2>&1"`
```

Note: Depending on your application template, the JOB_RESULT line may differ from the example. Ensure that ${CWD_OPT} is immediately after the **bsub** command.

8. Click **Save** to save your application.
9. Click **Publish** to publish your application.
10. Test your configuration.
    a. Select **Jobs** > **Submission Forms** and the application you just modified.
    b. Specify the job directory location in your new field.
    c. Submit the job.

    The job directory that is created should be the one you specified.

Attach a help page to your application submission form

In your application template, you can write help tooltips for each field in the submission form. You can also create an entire HTML help page for the submission form that is displayed as a popup window when users click on the help icon which is in the upper-right corner of the submission form.

Before you begin

Your application template must have the status Unpublished so that you can make changes.

Procedure

1. In **Resources** > **Submission Templates** > **Application Templates**, click the application template you want to modify.

   The modification window is displayed.
2. Click the **Help Documentation** tab.
3. Check the **Show help icon and enable help pages** field.
4. Enter a title for your help page and click the **New Section** button to add text to your help page.

   Write in plain text. It will be automatically converted to HTML.

   HTML href links to URLs are accepted as long as the target HTML files are on a web server accessible to Platform Application Center.
5. Click **Save** to save your changes.
6. Click **Publish** to publish your application.
7. Test your configuration.
   a. Select **Jobs** > **Submission Forms** and the application you just modified.
   b. Click the help icon which is in the upper-right corner of the submission form.

      Your help page is displayed in a popup window.

Chapter 4. Users and Security

Usage Statistics

Usage statistics allows you to track current and historical usage of Platform Application Center and manage compliance with the license agreement.

The Usage Statistics chart displays the number of concurrent users logged into Platform Application Center for the past 24 hours, past 2 days, and past week.

You can mouse over points in the chart to get specific information.

The data is cumulative and displays the peak number of concurrent users over the last 7 days, and the last 52 weeks.

By default, the sampling period is 5 minutes, and can be configured by editing the parameter `sampleCurrentUser_interval` in `/opt/pac/gui/conf/pmc.conf`.

View logon user statistics

Procedure

Choose **Settings** > **Usage Statistics**.

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`.

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	Any operating system user. All users are automatically assigned this role.	Jobs: • 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. If IBM Platform Process Manager is installed: • View Job Flow definitions If IBM Platform License Scheduler is installed: • View all licenses If Reports are installed: • View and run all reports If the IBM Platform Analytics add-on is installed, a user with the Normal user role can: • View past reports • Subscribe to a report, or unsubscribe from a report to receive emails when reports are updated
Cluster administrator	All LSF administrators defined in the LSF `lsf.cluster.`*`cluster_name`* file.Users and user groups are automatically assigned based on the configuration of your lsf.cluster.*cluster_name* file.	Full permission on all Platform Application Center pages and objects (view, control, configure).If the IBM Platform Analytics add-on is installed, a user with the Cluster administrator role can: • 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	Displayed as the name of the user group. Includes all user group administrators defined in the LSF lsb.users file.Users are automatically assigned based on the configuration of your lsb.users file.	• Same permissions as Normal user • View, control, configure all jobs submitted by users who are members of the user group.

User role	User/User group members	Summary of permissions
Flow administrator	All Process Manager administrators defined in the Process Manager configuration file `js.conf` .Users and user groups are automatically assigned based on the configuration of your js.conf file.	• Same permissions as Normal user • View, control, configure all job definitions and job flows.
Flow control administrator	All Process Manager control administrators defined in the the Process Manager configuration file `js.conf`.Users and user groups are automatically assigned based on the configuration of your js.conf file.	• Same permissions as Normal user • View, control, configure all job flows.
Report administrator	Used when the IBM Platform Analytics add-on is installed.Platform Application Center automatically loads Tableau Licensed users as Report administrator.	A user with the Report administrator role can: • 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	• 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	• View all application templates that exist • No actions on submission forms • View template details through web services	In submission form: • Submit • Save As • Revert Can submit a form through web services	• View all application templates on the list • Take all actions on application templates: • Publish • Unpublish • Save • Modify	All templates are not visible to the user.
Specific application templatesSpecific 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	• View Hosts in the navigation tree • View Hosts list table; no actions available on hosts • Can sort and filter hosts	• 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	• 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	All job control actions on all jobs in the system: • New • Suspend • Resume • Kill • Requeue • View output Control jobs through web servicesAll data control actions: • View • Download • Copy to • Move to • Delete • More actions All remote console actions • 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.
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 Details (User's own jobs)	User can view the Job Details pane for jobs he submitted.	N/A	N/A	User cannot view the Job Details pane for jobs he submitted.
Job Details (Other users' jobs)	User can view the Job Details pane for jobs submitted by other users.	N/A	N/A	User cannot view the Job Details pane for jobs submitted by other users.

Job Flows permissions

Only available in IBM Platform Application Center Standard Edition.

Applies to	View	Control	Configure	None
All job flows	• View Job Flows in the navigation tree • View Job Flows page; no actions	All job flow actions on all job flows in the system: • 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

Only available in IBM Platform Application Center Standard Edition.

Applies to	View	Control	Configure	None
All job flow definitions	• 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: • 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 IBM Platform License Scheduler(Platform License Scheduler) and IBM Platform Application Center Standard Edition 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	• 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

Only available in IBM Platform Application Center Standard Edition.

Applies to	View	Control	Configure	None
Reports	• View Reports on the navigation tree • View report details • Cannot generate or copy reports	Generate reports	• 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	• View Settings, Logos and Colors in navigation tree	N/A	• Reset changed values to saved settings • Apply • Revert	Settings is not displayed in the navigation tree
Page Settings	• 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)	• View Settings, Users in navigation tree • No actions		• Assign roles to users and user groups • Add, remove, edit roles • Define permissions for user roles.	Settings is not displayed in the navigation tree

Chapter 5. Remote Visualization

Enabling Remote Consoles

The remote console feature is disabled by default.

Platform Application Center supports remote consoles with:

- VNC
- TurboVNC
- NICE Desktop Cloud Visualization(DCV)
- HP Remote Graphics Software(RGS)
- Exceed onDemand(EoD)

For configuration details, refer to *Administering Platform Application Center*.

Using Platform Application Center with VNC

Before you begin

Prerequisites:

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

About this task

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

Procedure

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 template

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 VNCSesssion 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
${GUI_CONFDIR}/application/job-result.sh
```

Using Platform Application Center with NICE Desktop Cloud visualization(DCV) on Linux

You can configure Platform Application Center and Platform LSF to enable viewing of a 2D/3D Windows application from Platform Application Center by using NICE Desktop Cloud Visualization (DCV).

Users use Platform Application Center and Platform LSF to start their application and view the results remotely through DCV. Platform LSF schedules and allocates hosts that have the specific application installed.

Users do not need to know which hosts have the application installed or which hosts are available.

In this way, compute resources and application licenses can be shared, increasing resource efficiencies and reducing cost.

Platform Application Center provides a default AppDCVonLinux application template. You can create custom application templates to support additional applications.

Installation architecture

This integration supports both physical and virtual hosts as application hosts.

A typical installation includes the following:

- An LSF cluster with an LSF master host and LSF compute hosts. The LSF compute hosts act as application hosts.
- Platform Application Center is installed on the LSF master host.
- Users access Platform Application Center through a web browser on their workstations and are able to remotely visualize 2D/3D interactive applications.

The following diagram illustrates a typical installation in which the LSF master host is a physical Linux host, the application hosts (LSF compute hosts) are virtual Windows hosts, and the end user stations are Windows hosts.

Requirements

Platform LSF compute hosts (application hosts)

Item	Description
Operating system	• RHEL 6.2 64-bit Installed with Gnome Desktop Support
Machine type	• Virtual machine or physical machine
Software	• Platform LSF 9.1. • RealVNC Visualization Edition (VE) 4.5.1 Server is properly installed and configured • NICE DCV Server 2012.0-4557 is properly installed and configured
User accounts	• OS user account names are used.
Shared disk	• It is recommended to configure a shared disk accessible to all LSF compute hosts (application hosts). • File permissions on the shared disk are managed through OS user account permissions.

Platform LSF master and Platform Application Center web server

Item	Description
Operating system	• RHEL 6 or SUSE 10.x, 11 64-bit
Machine type	• Physical machine recommended
Software	• Platform Application Center 9.1.0.0 • Platform LSF 9.1.0.0
Additional requirements	• Install Platform Application Center on the same host as the LSF master host. • Refer to the Platform LSF and Platform Application Center installation documentation for more details.

DCV rendering hosts

DCV Rendering hosts must be properly installed and configured.

User workstations (DCV end station)

Item	Description
Operating system	• Windows
Machine type	• Virtual machine or physical machine
Software	• RealVNC Visualization Edition (VE) 4.5.1 Viewer is properly installed and configured • DCV Receiver is installed
Web browser	• Internet Explorer 8, 9 • Mozilla Firefox 9, 10-17

1. Configure Platform LSF for DCV

Procedure

1. Log on to the LSF master host as the LSF administrator.
2. Set your LSF environment.

   For csh or tcsh:

   ```
   % source LSF_TOP/conf/cshrc.lsf
   ```

   For sh, ksh, or bash:

   ```
   $ . LSF_TOP/conf/profile.lsf
   ```
3. Edit the file `$LSF_ENVDIR/lsf.shared` and add the resource `dcv_linux` as follows:

```
Begin Resource
RESOURCENAME  TYPE    INTERVAL INCREASING  DESCRIPTION        # Keywords
dcv_linux      Boolean ()       ()          (dcv_linux resource type)
End Resource
```

4. Edit the file `$LSF_ENVDIR/lsf.cluster` and add the `dcv_linux` resource to each LSF compute host (application host).

   For example, your application hosts are dcv_vm1 and dcv_vm2:

   ```
   Begin   Host
   HOSTNAME  model    type        server r1m  mem  swp  RESOURCES   #Keywords
   dcv_vm1   !   !   1   3.5   ()   ()   (dcv_linux)
   dcv_vm2   !   !   1   3.5   ()   ()   (dcv_linux)
   End Host
   ```
5. Configure LSF so that each LSF compute host (application host) only has 1 job slot. Complete this step only if LSF compute hosts (application hosts) have more than 1 CPU.

   This is required so that only one application can run at the same time on the host and DCV works properly.

   Edit the file `$LSF_ENVDIR/lsb.hosts` and set `MXJ` to 1 for each LSF compute host.
6. Run **`lsadmin reconfig`** and **`badmin reconfig`** to make your changes take effect.

2. Configure the LSF compute hosts (application hosts)

Repeat the following steps on each LSF compute host (application host).

Procedure

1. Install RealVNC VE.
2. 3. Run the VNC server for the first time.

   The VNC server is set with the default password after installation.

   Run the **`vncserver`** command and enter your system password.

   If you see the prompt “`Couldn't open RGB_DB '/usr/X11R6/lib/X11/rgb`” when running the command, do the following:

   ```
   mkdir /usr/X11R6/lib
   cd /usr/X11R6/lib
   ln -s /usr/share/X11 X11
   ```

3. Create a custom application for DCV

Procedure

1. Log on to Platform Application Center as the LSF administrator, select **Resources** > **Submission Templates** > **Application Templates** and select **AppDCVonLinux**.

2. Click the **Save As** button and save the application with a new name.
3. Publish your new application template so that it is accessible to users.

4. Test

Procedure

1. Log on to Platform Application Center, select **Jobs** > **Submission Forms** and select your application name.

   The job submission form is displayed.
2. Complete any required parameters, select any required input files, and click the **Submit** button.

   The job is submitted and job details are displayed.
3. Click the **Visualize** icon to view the 2D/3D application in DCV.

Using Platform Application Center with NICE Desktop Cloud visualization(DCV) on Windows

You can configure Platform Application Center and Platform LSF to enable viewing of a 2D/3D Windows application from Platform Application Center by using NICE Desktop Cloud Visualization (DCV).

Users use Platform Application Center and Platform LSF to start their application and view the results remotely through DCV. Platform LSF schedules and allocates hosts that have the specific application installed.

Users do not need to know which hosts have the application installed or which hosts are available.

In this way, compute resources and application licenses can be shared, increasing resource efficiencies and reducing cost.

Platform Application Center provides a default CATIA application template. You can create custom application templates to support additional applications.

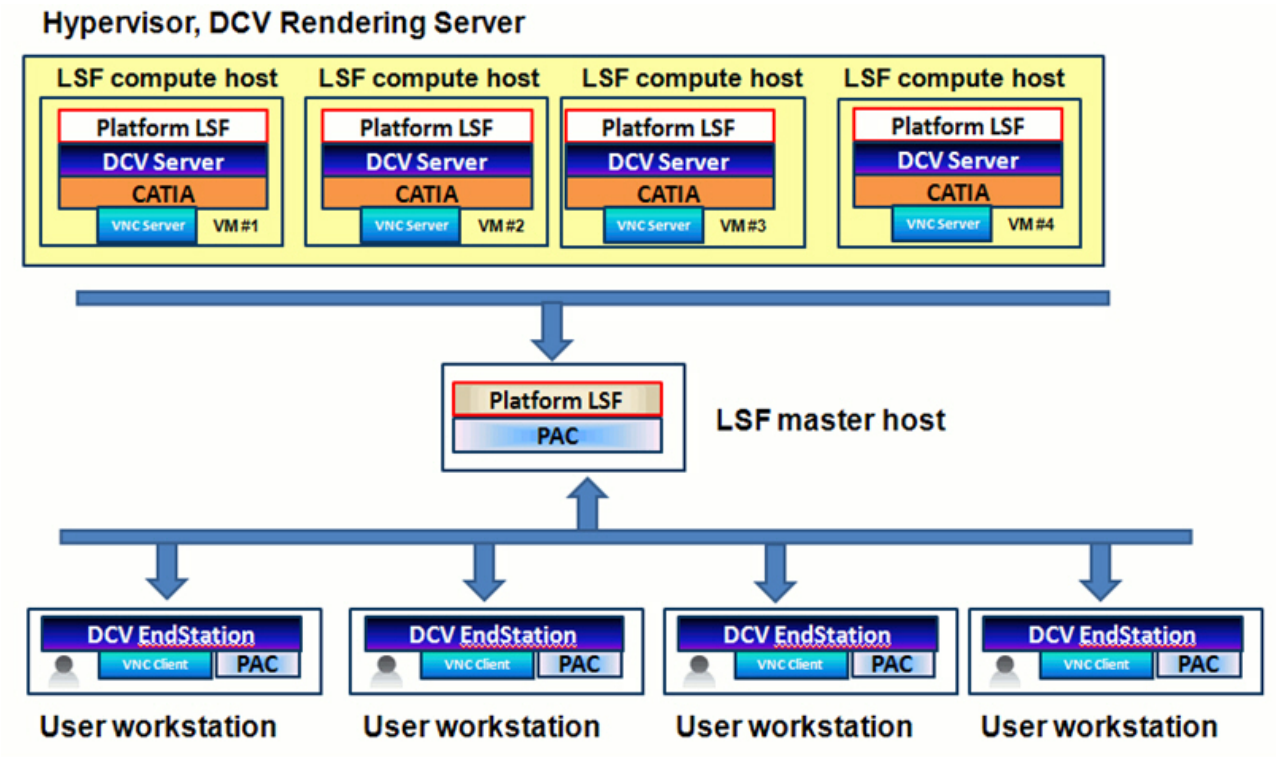
Installation architecture

This integration supports both physical and virtual hosts as application hosts.

A typical installation includes the following:

- An LSF cluster with an LSF master host and LSF compute hosts. The LSF compute hosts act as application hosts.
- Platform Application Center is installed on the LSF master host.
- Users access Platform Application Center through a web browser on their workstations and are able to remotely visualize 2D/3D interactive applications.

The following diagram illustrates a typical installation in which the LSF master host is a physical Linux host, the application hosts (LSF compute hosts) are virtual Windows hosts, and the end user stations are Windows hosts.



Requirements

Platform LSF compute hosts (application hosts)

Item	Description
Operating system	• Windows 7 Enterprise SP1 English Edition
Machine type	• Virtual machine or physical machine
Software	• Platform LSF 9.1. • RealVNC Visualization Edition (VE) 4.5.1 Server is properly installed and configured • NICE DCV Server 2012.0-4557 is properly installed and configured • CATIA V5R20 is properly installed and configured (if using the CATIA application)
User accounts	• User accounts are configured in a central user database such as Active Directory. • The same OS user account name is configured on both Linux and Windows. • The user accounts in the central user database have local administrator privileges on the LSF compute hosts (application hosts).
Shared disk	• It is recommended to configure a shared disk accessible to all LSF compute hosts (application hosts). • File permissions on the shared disk are managed through OS user account permissions.

Platform LSF master and Platform Application Center web server

Item	Description
Operating system	• RHEL 6 or SUSE 10.x, 11 64-bit
Machine type	• Physical machine recommended
Software	• Platform Application Center 9.1.0.0 • Platform LSF 9.1.0.0
Additional requirements	• Install Platform Application Center on the same host as the LSF master host. • Refer to the Platform LSF and Platform Application Center installation documentation for more details.

DCV rendering hosts

DCV Rendering hosts must be properly installed and configured.

User workstations (DCV end station)

Item	Description
Operating system	• Windows English Edition
Machine type	• Virtual machine or physical machine
Software	• RealVNC Visualization Edition (VE) 4.5.1 Viewer is properly installed and configured • DCV Receiver is installed
Web browser	• Internet Explorer 8, 9 • Mozilla Firefox 9, 10-17

1. Configure Platform LSF for DCV

Procedure

1. Log on to the LSF master host as the LSF administrator.
2. Set your LSF environment.

   For csh or tcsh:

   ```
   % source LSF_TOP/conf/cshrc.lsf
   ```

   For sh, ksh, or bash:

   ```
   $ . LSF_TOP/conf/profile.lsf
   ```
3. Edit the file `$LSF_ENVDIR/lsf.shared` and add the resource `dcv` as follows:

```
Begin Resource
RESOURCENAME  TYPE    INTERVAL INCREASING  DESCRIPTION       # Keywords
dcv      Boolean        ()       ()          (dcv resource type)
End Resource
```

4. Edit the file `$LSF_ENVDIR/lsf.cluster` and add the `dcv` resource to each LSF compute host (application host).

   For example, your application hosts are dcv_vm1 and dcv_vm2:

```
Begin   Host
HOSTNAME  model    type        server r1m  mem  swp  RESOURCES   #Keywords
dcv_vm1   !         !             1   3.5   ()   ()   (dcv)
dcv_vm2   !         !             1   3.5   ()   ()   (dcv)
End Host
```

5. Configure LSF so that each LSF compute host (application host) only has 1 job slot. Complete this step only if LSF compute hosts (application hosts) have more than 1 CPU.

   This is required so that only one application can run at the same time on the host and DCV works properly.

   Edit the file `$LSF_ENVDIR/lsb.hosts` and set `MXJ` to 1 for each LSF compute host.

6. Edit the file `$LSF_ENVDIR/lsf.conf` and define the `LSF_USER_DOMAIN` parameter.
   - If LSF compute hosts are configured as a Windows Workgroup, set `LSF_USER_DOMAIN=.`
   - If LSF compute hosts are configured in a domain, set `LSF_USER_DOMAIN=`*`domain_name`* .

     For example, if your domain name is mydomain, set `LSF_USER_DOMAIN=mydomain`

7. Run **`lsadmin reconfig`** and **`badmin reconfig`** to make your changes take effect.

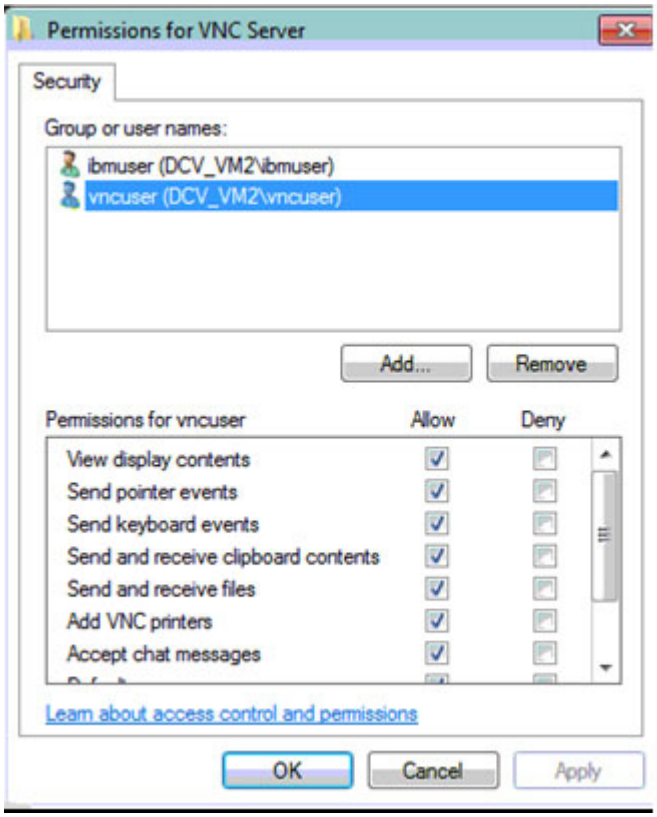
2. Configure the LSF compute hosts (application hosts)

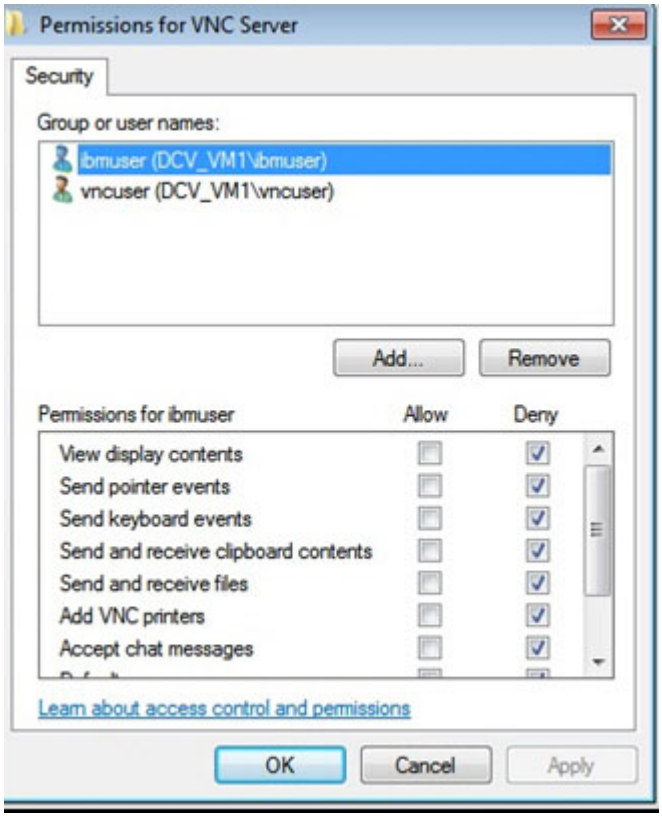
Repeat the following steps on each LSF compute host (application host).

Procedure

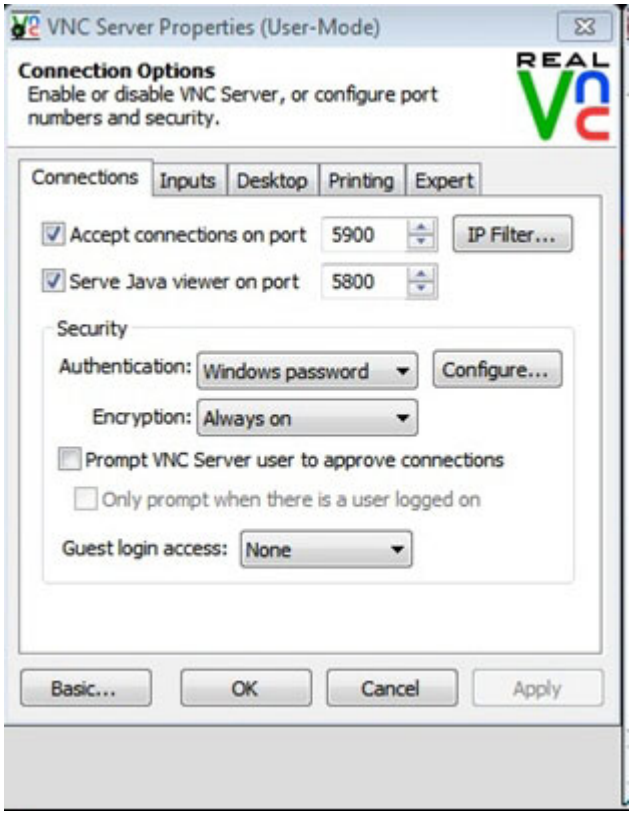
1. Configure VNC.
   a. Stop and disable the VNC Windows service.
   b. Open VNC: **Start** > **Run** `C:\Program Files\RealVNC\VNC4\vncconfig.exe`
   c. Configure VNC to only allow the vncuser to connect.

      The following images display configuration for the vncuser and configuration for another user, ibmuser. Notice that only the vncuser has permissions.


Permissions for VNC Server
Security
Group or user names:
ibmuser (DCV_VM2\ibmuser)
vncuser (DCV_VM2\vncuser)
Add...
Remove
Permissions for vncuser
Allow
Deny
View display contents
Send pointer events
Send keyboard events
Send and receive clipboard contents
Send and receive files
Add VNC printers
Accept chat messages
Learn about access control and permissions
OK
Cancel
Apply


Permissions for VNC Server
Security
Group or user names:
ibmuser (DCV_VM1\ibmuser)
vncuser (DCV_VM1\vncuser)
Add...
Remove
Permissions for ibmuser
Allow
Deny
View display contents
Send pointer events
Send keyboard events
Send and receive clipboard contents
Send and receive files
Add VNC printers
Accept chat messages
Learn about access control and permissions
OK
Cancel
Apply

d. In **Connection Options**, for **Authentication** select the **Windows password** method.



2. Configure screen resolution
   a. Download `ResSwitch` to dynamically change the screen resolution on LSF compute hosts.

      Download `ResSwitch.exe` from: http://www.naughter.com/qres.html
   b. Copy `ResSwitch.exe` to `C:\LSF_9.1\9.1\bin`.
3. Configure local administrator privileges.
   a. Select **Start** > **Run** and, type `secpol.msc` to open the **Local Security Settings console**.
   b. Select **Local Policies** > **User Rights Assignment**, and check that the local administrator (automatic logon user) has been assigned the following policies:
      - **Act as part of the operating system**
      - **Debug programs**
      - **Log on as a service**
      - **Log on as a batch job**
      - **Replace a process-level token**
4. Set the Windows environment by running `setenv.bat`.
   a. Copy the `setenv.bat` file from the Platform Application Center web server `$GUI_CONFDIR/application/vnc/` to `C:\` on the LSF compute host.
   b. Open a command prompt, type `cd C:\` and press **Enter**.
   c. Run setenv.bat by specifying the local administrator password.

      For example:

```
setenv.bat myadminpassword
```

5. Change the Windows local administrator user password to match the password used to log on to Platform Application Center.

3. Configure application templates

Configure the CATIA application

Procedure

1. Log on to the Platform Application Center web server host as the LSF administrator.
2. Create the rendering server map file `$GUI_CONFDIR/application/vnc/app_render_host.map` and map each LSF compute host (application host) to its Rendering Server.

   Note: The Rendering Server is generally configured as the virtual machine's hypervisor for better performance. Check the DCV documentation for details on selecting and setting up the Rendering Server.

   Each line in the file maps an LSF compute host (application host) to the IP address of its rendering server in the format:
   LSF_compute_host_name=rendering_server_IP_address

   For example:

   ```
   dcv_vm1=10.10.100.1
   ```

3. Complete this step ONLY if you did not install Platform LSF or RealVNC in the default locations.

   The CATIA template uses the default installation location for the RealVNC server, LSF and applications.

   If you did not use the default location at installation, modify the following files and specify the correct location:

   - `$GUI_CONFDIR/application/vnc/prepare_dcvjob.sh`
   - `$GUI_CONFDIR/application/vnc/start_catia.bat`

4. Update the WIN_DOMAIN variable with your Windows domain name in the file: `$GUI_CONFDIR/application/vnc/CATIA.cmd`
5. Log on to Platform Application Center as the LSF administrator and publish the CATIA application template so that it is accessible to users.

Create a custom application for DCV

Procedure

1. Log on to Platform Application Center the LSF administrator, select **Resources** > **Submission Templates** > **Application Templates** and select **CATIA**.
2. Click the **Save As** button and save the application with a new name.
3. Create the rendering server map file `$GUI_CONFDIR/application/vnc/app_render_host.map` and map each LSF compute host (application host) to its Rendering Server.

   Note: The Rendering Server is generally configured as the virtual machine's hypervisor for better performance. Check the DCV documentation for details on selecting and setting up the Rendering Server.

   Each line in the file maps an LSF compute host (application host) to the IP address of its rendering server in the format:
   LSF_compute_host_name=rendering_server_IP_address

   For example:

```
dcv_vm1=10.10.100.1
```

4. Complete this step ONLY if you did not install Platform LSF or RealVNC in the default locations.

   The CATIA template uses the default installation location for the RealVNC server, LSF and applications.

   If you did not use the default location at installation, modify the following files and specify the correct location:

   - `$GUI_CONFDIR/application/vnc/prepare_dcvjob.sh`
   - `$GUI_CONFDIR/application/vnc/start_catia.bat`
5. Update the WIN_DOMAIN variable with your Windows domain name in the file: `$GUI_CONFDIR/application/vnc/CATIA.cmd`
6. Log on to Platform Application Center as the LSF administrator and publish the CATIA application template so that it is accessible to users.

4. Test

Procedure

1. Log on to Platform Application Center, select **Jobs** > **Submission Forms** and select your new application name.

   For example, CATIA.

   The job submission form is displayed.
2. Complete any required parameters and click the **Submit** button.

   The job is submitted and job details are displayed.
3. Click the **Visualize** icon to view the 2D/3D application in DCV.

Using Platform Application Center with HP Remote Graphics Software(RGS)

You can configure Platform Application Center and Platform LSF to enable viewing of a 2D/3D Windows application from Platform Application Center by using HP Remote Graphics Software(RGS).

Users use Platform Application Center and Platform LSF to start their application and view the results remotely through RGS. Platform LSF schedules and allocates hosts that have the specific application installed.

Users do not need to know which hosts have the application installed or which hosts are available.

In this way, compute resources and application licenses can be shared, increasing resource efficiencies and reducing cost.

Platform Application Center provides a default HP-RGS application template. Users can request a remote desktop by using the HP-RGS submission form.

Installation architecture

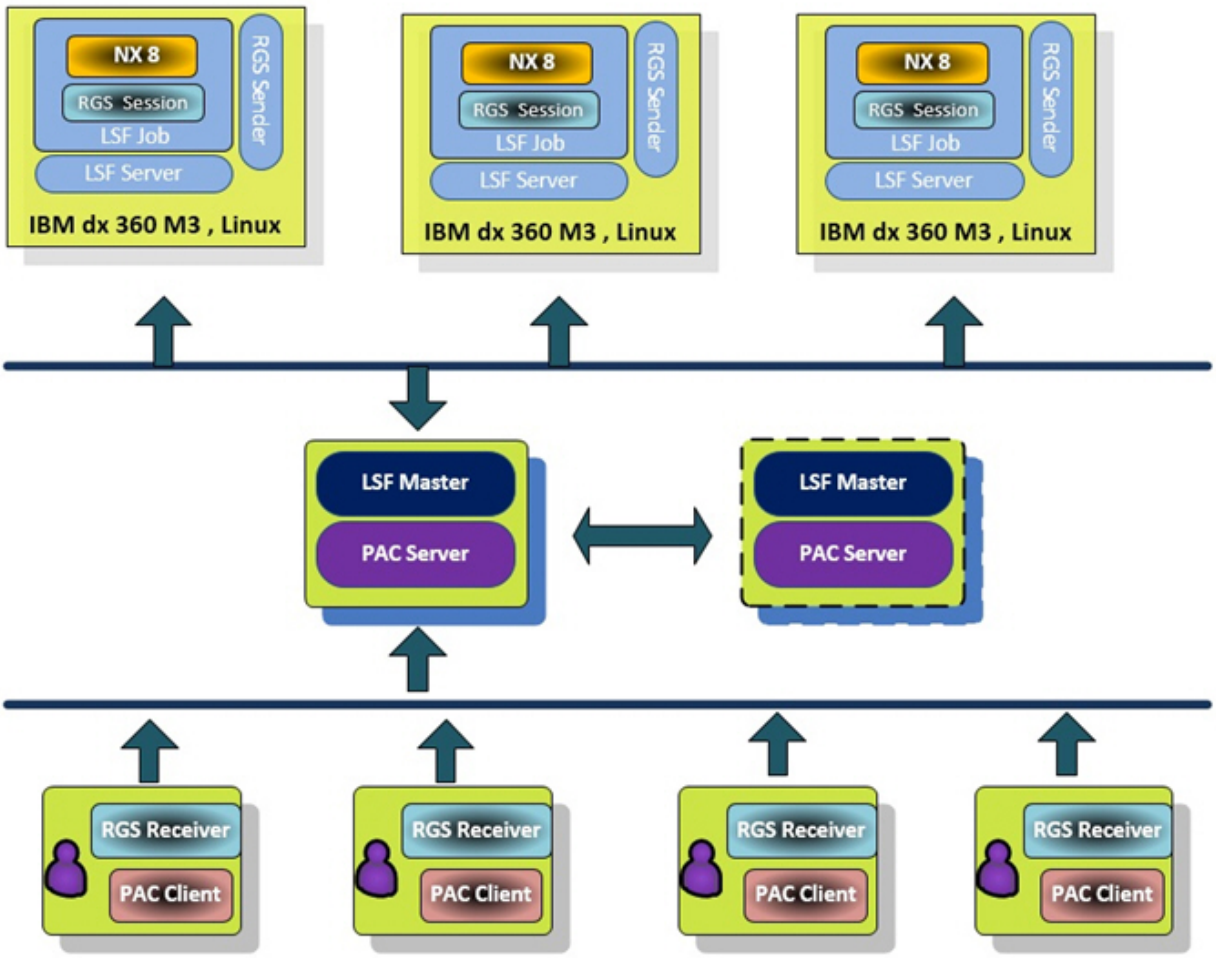
This integration supports both physical and virtual hosts as application hosts.

A typical installation includes the following:

- An LSF cluster with an LSF master host and LSF compute hosts. The LSF compute hosts act as application hosts.
- Platform Application Center is installed on the LSF master host.

- Users access Platform Application Center through a web browser on their workstations and are able to remotely visualize 2D/3D interactive applications.

The following diagram illustrates a typical installation in which the LSF master host and compute hosts are physical Linux hosts, and the end user stations are Windows hosts. Each compute host has an Nvidia graphic card installed.



Requirements

Platform LSF compute hosts (application hosts)

Item	Description
Operating system	• RHEL 5/6 64-bit Installed with Gnome Desktop Support
Machine type	• Physical machine with Nvidia graphic card installed
Software	• Platform LSF 9.1. • HP Remote Graphic Software 5.4.7/5.4.8 is properly installed and configured
User accounts	• OS user account names are used.

Item	Description
Shared disk	• It is recommended to configure a shared disk accessible to all LSF compute hosts (application hosts). • File permissions on the shared disk are managed through OS user account permissions.

Platform LSF master and Platform Application Center web server

Item	Description
Operating system	• RHEL 5/6 or SUSE 10.x, 11 64-bit
Machine type	• Physical machine recommended
Software	• Platform Application Center 9.1.0.0 • Platform LSF 9.1.0.0
Additional requirements	• Install Platform Application Center on the same host as the LSF master host. • Refer to the Platform LSF and Platform Application Center installation documentation for more details.

User workstations (RGS Receiver Host)

Item	Description
Operating system	• Windows 7/XP
Machine type	• Virtual machine or physical machine
Software	• RGS Receiver is installed
Web browser	• Internet Explorer 8, 9 • Mozilla Firefox 9, 10-17

1. Configure Platform LSF for RGS

Procedure

1. Log on to the LSF master host as the LSF administrator.
2. Set your LSF environment.

   For csh or tcsh:

   ```
   % source LSF_TOP/conf/cshrc.lsf
   ```

   For sh, ksh, or bash:

   ```
   $ . LSF_TOP/conf/profile.lsf
   ```
3. Edit the file `$LSF_ENVDIR/lsf.shared` and add the resource `rgs_linux` for the RGS sender host as follows:

```
Begin Resource
RESOURCENAME  TYPE    INTERVAL INCREASING  DESCRIPTION       # Keywords
rgs_linux      Boolean ()       ()     (rgs_linux resource type)
End Resource
```

4. Edit the file `$LSF_ENVDIR/lsf.cluster` and add the `rgs_linux` resource to each LSF compute host (application host).

   For example, your application hosts are compute1 and compute2:

```
Begin   Host
HOSTNAME  model    type        server r1m  mem  swp  RESOURCES   #Keywords
compute1   !   !   1   3.5   ()   ()   (rgs_linux)
compute2   !   !   1   3.5   ()   ()   (rgs_linux)
End Host
```

5. Edit the file `$LSF_ENVDIR/lsbatch/`*`cluster_name`*`/configdir/lsb.queues` and define a new queue for RGS jobs.

```
Begin Queue
QUEUE_NAME=rgs_job_queue
RES_REQ       = rgs_linux
EXCLUSIVE     = Y
...
End Queue
```

6. Run **`lsadmin reconfig`** and **`badmin reconfig`** to make your changes take effect.

2. Configure the LSF compute hosts (application hosts)

Repeat the following steps on each LSF compute host (application host).

Procedure

1. Install HP RGS rgsender 5.4.7 or 5.4.8.
2. Start `rgssender`.

```
#init 5
```

OR

```
#startx
```

3. Configure user workstations(RGS receiver hosts)

These steps apply to Windows 7 and windows XP 32-bit and 64-bit.

Procedure

1. Install HP RGS Receiver 5.4.7 or 5.4.8.
2. To set up the client environment, copy the file `$GUI_CONFDIR/application/vnc/set_rgs_client.bat` from the Platform Application Center web server to the user workstation.
3. Log on to the workstation as local administrator and run `set_rgs_client.bat`.

4. Configure and publish the HP-RGS application template

Procedure

1. Log on to Platform Application Center the LSF administrator, select **Resources** > **Submission Templates** > **Application Templates** and select **HP-RGS**.
2. Double-click on the template name and in the **Domain** field, enter the Windows user domain name of all the workstations, and click the **Save** button to save your changes.
3. Click the **Publish** button and select the **rgs_job_queue** as the queue name in the dialog.

5. Test

Procedure

1. From a workstation, log on to Platform Application Center, select **Jobs** > **Submission Forms** and select HP-RGS.

   The job submission form is displayed.
2. Complete any required parameters, select any required input files, and click the **Submit** button.

The job is submitted and job details are displayed.

3. Click the **Visualize** icon to view the RGS desktop.
4. Log on to the RGS desktop using the `gdm` user and password.
   The RGS desktop console is displayed.
5. Start a graphic application in the RGS desktop console.

Using Platform Application Center with Exceed OnDemand

You can configure Platform Application Center and Platform LSF to enable viewing of a 2D/3D Windows application from Platform Application Center by using Exceed On Demand(EoD).

Users use Platform Application Center and Platform LSF to start their application and view the results remotely through EoD. Platform LSF schedules and allocates hosts that have the specific application installed.

Users do not need to know which hosts have the application installed or which hosts are available.

In this way, compute resources and application licenses can be shared, increasing resource efficiencies and reducing cost.

Platform Application Center provides application templates for two modes of operation.

Mode 1:

- Template names: EXCEED_Xterm, EXCEED_Desktop
- Template type : EXCEED_Xterm, EXCEED_Desktop

Mode 2:

- Template names: EXCEED_Xterm_Enterprise, EXCEED_Desktop_Enterprise
- Template type : EXCEED_Xterm_Enterprise, EXCEED_Desktop_Enterprise

Note: The template type must be the same as the template name.

You can create custom application templates to support additional applications.

Installation architecture

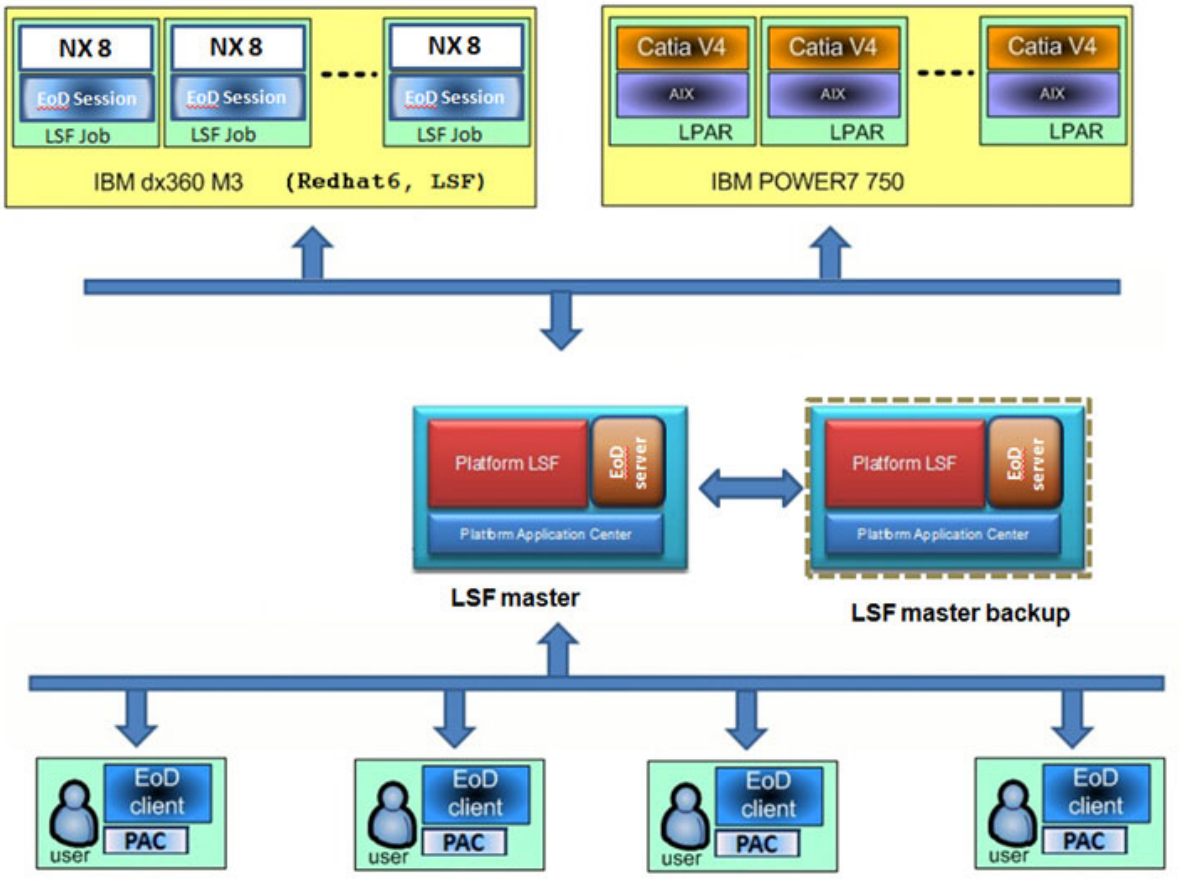
This integration supports both physical and virtual hosts as application hosts.

Mode 1

A typical installation includes the following:

- An LSF cluster with an LSF master host and LSF compute hosts. The LSF compute hosts act as application hosts.
- One EoD server serves all the application hosts managed through LSF.
- Platform Application Center is installed on the same host as the EoD server.
- LSF/Platform Application Center/EoD high availability is supported when all three products are installed with failover and the LSF master, Platform Application Center web server, and EoD server are on the same host.
- Users access Platform Application Center through a web browser on their workstations and are able to remotely visualize 2D/3D interactive applications.

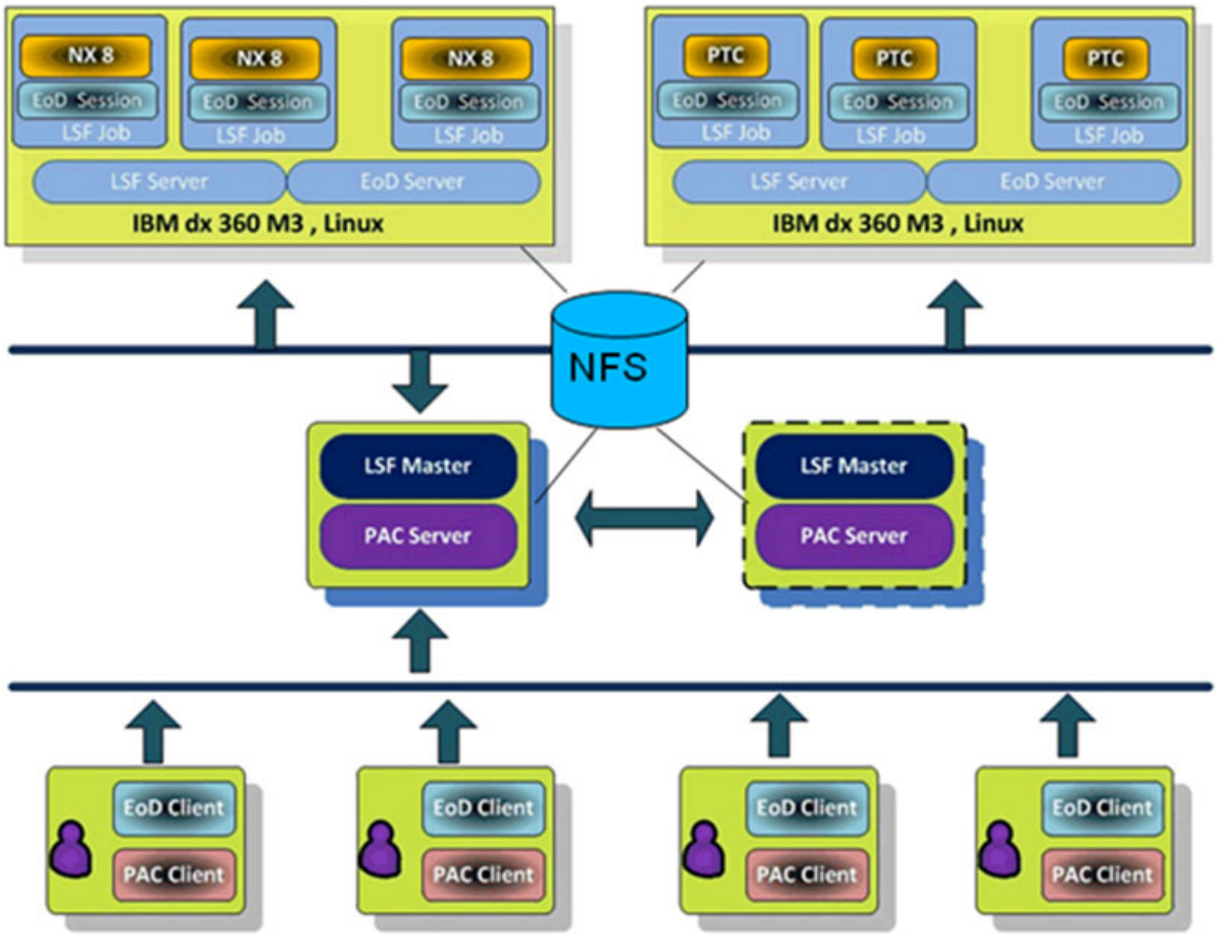
The following diagram illustrates a typical installation in which the head host(in which LSF master host, Platform Application web server, and EoD server is installed) is a physical 64-bit Linux server, and application hosts (LSF compute hosts) are Linux 64-bit hosts. High Availability is supported in this configuration.



Mode 2

A typical installation includes the following:

- An LSF cluster with an LSF master host and LSF compute hosts.
- EoD server is installed on each LSF compute host. The LSF compute hosts act as application hosts.
- Platform Application Center is installed on the same host as the EoD master.
- LSF/Platform Application Center high availability is supported when both Platform Application Center and LSF are installed with failover and the LSF master and Platform Application Center web server are on the same host.
- Users access Platform Application Center through a web browser on their workstations and are able to remotely visualize 2D/3D interactive applications.



Requirements (Mode 1)

Platform LSF compute hosts (application hosts)

Item	Description
Operating system	• 64-bit Linux, glibc 2.3
Machine type	• Virtual machine, physical machine, or logical partition
Software	• Platform LSF 9.1. • Siemens NX 8 is properly installed • Your application is properly installed and configured. For example, CATIA V4 (AIX®) if using the CATIA application.
User accounts	• User accounts are configured in a central user database • The same OS user account name is configured on both LSF master and compute host
Shared disk	• Configure a shared disk accessible to all LSF compute hosts (application hosts). • File permissions on the shared disk are managed through OS user account permissions.

Platform LSF master and Platform Application Center web server

Item	Description
Operating system	• RHEL 6 or SUSE 10.x, 11 64-bit
Machine type	• Physical machine recommended
Software	• Platform Application Center 9.1.0.0 • Platform LSF 9.1.0.0 • Exceed onDemand server 13.8 SP1
Additional requirements	• The LSF master host, Exceed onDemand Server, and Platform Application Center web server must all be installed on the same host. • Refer to the Platform LSF and Platform Application Center installation documentation for more details.

3D rendering hosts

The 3D Rendering Host is the LSF master host.

User workstations (EoD client)

Item	Description
Operating system	• Windows English Edition
Machine type	• Virtual machine or physical machine
Software	• Exceed onDemand Client 13.8 SP1 is installed • Make sure the port 5500 is available to install Exceed onDemand clients.
Web browser	• Internet Explorer 8, 9 • Mozilla Firefox 9, 10 • JRE1.6.0_10 or higher plugin in web browser.

Limitations

- Jobs cannot be rerun or resubmitted.
- Web Services cannot be used to submit EoD jobs.
- The Suspend job operation suspends the X application, but does not suspend the EoD job's session.
- EoD jobs must be submitted only from Platform Application Center so that Platform Application Center manages the jobs.

Requirements (Mode 2)

Platform LSF compute hosts (application hosts)

Item	Description
Operating system	• 64-bit Linux, glibc 2.3
Machine type	• Virtual machine, physical machine, or logical partition

Item	Description
Software	• Platform LSF 9.1. • Siemens NX 8 is properly installed • CATIA V4 (AIX) is properly installed and configured (if using the CATIA application)
User accounts	• User accounts are configured in a central user database • The same OS user account name is configured on both LSF master and compute host
Shared disk	• Configure a shared disk accessible to all LSF compute hosts (application hosts). • File permissions on the shared disk are managed through OS user account permissions.

Platform LSF master and Platform Application Center web server

Item	Description
Operating system	• RHEL 6 or SUSE 10.x, 11 64-bit
Machine type	• Physical machine recommended
Software	• Platform Application Center 9.1.0.0 • Platform LSF 9.1.0.0
Additional requirements	• Install Exceed onDemand Server on all LSF compute hosts. Exceed onDemand Server must be installed in the same location on all compute hosts. • Refer to the Platform LSF and Platform Application Center installation documentation for more details.

3D rendering hosts

The 3D Rendering host is the LSF master host.

User workstations (EoD client)

Item	Description
Operating system	• Windows English Edition
Machine type	• Virtual machine or physical machine
Software	• EoD Client 13.8 SP1 is installed • Make sure the port 5500 is available to install Exceed onDemand clients.
Web browser	• Internet Explorer 8, 9 • Mozilla Firefox 9, 10 • JRE1.6.0_10 or higher plugin in web browser.

Limitations

- Jobs cannot be rerun or resubmitted.
- Web Services cannot be used to submit EoD jobs.

- The Suspend job operation suspends the X application, but does not suspend the EoD job's session.
- EoD jobs must be submitted only from Platform Application Center so that Platform Application Center manages the jobs.

1. Configure Platform LSF for EoD

Procedure

1. Log on to the LSF master host as the LSF administrator.
2. Set your LSF environment.

   For csh or tcsh:

   ```
   % source LSF_TOP/conf/cshrc.lsf
   ```

   For sh, ksh, or bash:

   ```
   $ . LSF_TOP/conf/profile.lsf
   ```
3. Configure LSF so that each LSF compute host (application host) only has a maximum number of job slots.

   This is required so that only a specific number of concurrent EoD client sessions can run at the same time on the host.

   Edit the file `$LSF_ENVDIR/lsb.hosts` and set `MXJ` to the maximum number of concurrent client sessions for each LSF compute host.
4. Run **`lsadmin reconfig`** and **`badmin reconfig`** to make your changes take effect.

2. Configure the LSF compute hosts (application hosts)

Repeat the following steps on each LSF compute host (application host).

Procedure

1. Enable the SSH service to allow users to log on and execute commands through SSH.
2. Check that gnome exists.

   ```
   #ls -l /usr/bin/gnome-session
   -rwxr-xr-x 1 root root 149488 Jan 31  2011 /usr/bin/gnome-session
   ```

   Note: If Gnome does not exist, edit the EXCEED_desktop template submission script in Platform Application Center and replace the line `EOD_XAPP_CMD="nohup /usr/bin/gnome-session` with another MDM Linux desktop manager.

3. Configure the Platform Application Center web server

Procedure

1. Create a shared directory with the name `eod_client_conn` on the Platform Application Center web server and set permissions so that all of the Platform LSF users have read and write permissions on that directory.

   ```
   # mkdir -p ${CATALINA_HOME}/webapps/platform/pac/exceed/eod_client_conn
   # chmod –R 777 ${CATALINA_HOME}/webapps/platform/pac/exceed/eod_client_conn
   ```
2. Modify the file `${GUI_CONFDIR}/application/exceed/ CONFIG_GLOBAL_ENV` and set the EOD_SERVER_DIR.

   For example:

   ```
   #EOD_SERVER_DIR     /opt/Exceed_Connection_Server_13.8_64
   ```

4. Configure the EoD server

Procedure

Set permissions so that all of the Platform LSF users can read and write to the `Xstart` and `Xconfig` directories under EoD server.
For example:

```
# chmod -R 777 /opt/Exceed_Connection_Server_13.8_64/conf/users
```

5. Configure EoD client hosts

For each end user host, you will need to complete the following configuration before users can access EoD through Platform Application Center.

Procedure

1. To make sure clients can access the IP address of the EoD Server host, edit the `C:\WINDOWS\system32\drivers\etc\hosts` file and add the mappings of IP addresses to the EoD Server host name.
2. Select **Control Panel** > **System Properties** > **Environment Variables**, edit the system **Path** variable and add `C:\Program Files\Hummingbird\Connectivity\Exceed onDemand Client 8 x64`.
3. Run `regedit`, go to `HKCU\Software\Hummingbird\Connectivity\Exceed onDemand Client 8\`, create a key/value pair, and set to: `SessionAutoResume=1` .
4. Run `regedit`, go to `HKCU\Software\Hummingbird\Connectivity\Exceed onDemand Client 8\`, create a key/value pair, and set to: `DoubleClickEoDFileAction=2` .
5. Modify `$JRE_HOME/lib/security/java.policy` and add the following line to the file:

   ```
   grant { ...
   permission java.io.FilePermission"<<ALL FILES>>","execute";
   ...};
   ```

6. Configure application templates

Default configuration

Submission form

Mode Type		Header
Mode 1	EXCEED_Xterm application	• **Job Name:** Job session name, user can change it to track different jobs. • **Submit to this Queue:** Select a queue for the job.
Mode 1	EXCEED_Xterm_Enterprise application	• **Job Name:** Job session name, user can change it to track different jobs. • **Submit to this Queue:** Select a queue for the job. • **Reusable:** If an EoD session is running and is empty, session is reused. Default is reuse. • **CPU:**the numb of CPUs required to run the job. • **Memory:** the memory requirement(MB). • **Idle time-out :**the maximum waiting time in minutes, for the Exceed client submit the job.

Mode Type		Header
Mode 2	EXCEED_Desktop application	• **Job Name:** Job session name, user can change it to track different jobs. • **Display Resolution:** Display Resolution of your desktop ,default value is 1024x768 • **Submit to this Queue:** Select a queue for the job.
Mode 2	EXCEED_Desktop_Enterprise application	• **Job Name:** Job session name, user can change it to track different jobs. • **Display Resolution:**Display Resolution of your desktop,default value is 1024x768 • **Submit to this Queue:** Select a queue for the job. • **Reusable:** If an EoD session is running and is empty, session is reused. Default is reuse. • **CPU:**the numb of CPUs required to run the job. • **Memory:**he memory requirement(MB) • **Idle time-out :**the maximum waiting time in minutes, for the Exceed client submit the job.

Submission script

Set Global environment:

- **EOD_SERVER_DIR**: EoD Server directory, read from `${GUI_CONFDIR}/application/exceed/CONFIG_GLOBAL_ENV`, used for connection to the EoD server.
- **EOD_HOST**: EoD server host name, default is hostname

Set Exceed **bsub** command environment:

- **EOD_CFG**: The name of the `Xconfig` file, without the file type.
- **EOD_XSTART**: The name of the `Xstart` file, without the file type.
- **EOD_XAPP_CMD**: X-application command, such as `xterm`, `nohup /usr/bin/gnome-session`.
- **QUEUE:** The queue for your job.

Create a custom application for EoD(Mode 1)

Procedure

1. Log on to Platform Application Center the LSF administrator, select **Resources** > **Submission Templates** > **Application Templates** and select **EXCEED_Xterm**.
2. Click the **Save As** button and save the application with a new name.
3. In **Resources** > **Submission Templates** > **Application Templates** select the new template and click the **Modify** button.
4. In the **Submission Form** subtab, set default properties such as JOB_NAME.
5. In the **Submission Script** subtab, replace xterm with the new application execution binary.

```
######Set Application environments ########
#Set Global environments
export MODE="mode1"
export EOD_HOST=`hostname`export EOD_SERVER_DIR=`cat ${GUI_CONFDIR}/application/exceed/CONFIG_GLOBAL_ENV |
grep EOD_SERVER_DIR |awk {'print $2'}`
#Set Constant environments
export EOD_CLIENT_CONN_DIR="${CATALINA_HOME}/webapps/platform/pac/exceed/eod_client_conn"
export EOD_CONF_DIR="${EOD_SERVER_DIR}/conf"
```

```
export EOD_CONF_TEMPLATE="${EOD_CONF_DIR}/admin/Templates/User"
export EOD_USER_DIR="${EOD_CONF_DIR}/users/${EXECUTIONUSER}"

#Set Exceed bsub command environments
export EOD_CFG="EXCEED_multi_window"
export EOD_XSTART="EXCEED_generic"export EOD_XAPP_CMD="xterm"
```

6. Save and publish your new application template so that it is accessible to users.

Publish the EOD application template

Procedure

1. Log on to Platform Application Center the LSF administrator, select **Resources** > **Submission Templates** > **Application Templates** and select the Exceed template that applies to your environment. All Exceed onDemand templates start with EXCEED.
2. Save the application template and click the **Publish** button so that the submission form is accessible to users.

7. Visualize

Procedure

1. Log on to Platform Application Center the LSF administrator, select **Jobs** > **Submission Forms** and select **EXCEED_Xterm**.
2. Enter the **Job Name** and **Password** and click the **Submit** button.

   Wait until the Xterm EoD job is created and started.

   For mode 2, when you click the Submit button, you can start the application by clicking the **Visualize** icon.
3. Select **Jobs** > **Jobs** and select your job from the job list.

   There will be no **Visualize** icon.
4. Suspend the EoD job from the EoD client.

   In Job Details, there will now be a **Visualize** icon.
5. Click the **Visualize** icon, download *.eod8 file to Resume the EoD job, and Run with ExceedOnDemand Client.

   The EoD job will be resumed.
6. If EoD job has resumed, when you click the **Visualize** icon again, it will pop up a warning window to alert user.

8. Test

Procedure

1. Log on to Platform Application Center, select **Jobs** > **Submission Forms** and select your application name. For example, EXCEED_Desktop.

   The job submission form is displayed.
2. Complete any required parameters, select any required input files, and click the **Submit** button.

   The job is submitted and job details are displayed.
3. Remote Desktop will display whenever there is an available LSF compute host.

Chapter 6. Configure and Manage

Enable license usage monitoring with the IBM Platform License Scheduler integration

About this task

Platform Application Center can display license usage when IBM 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:

- IBM Platform License Scheduler is installed in your IBM Platform LSF cluster.
- You have IBM Platform Application Center Standard Edition.
- Platform Application Center is installed and can access $LSF_ENVDIR.

After installation, you need to enable license usage monitoring through Platform Application Center.

Procedure

1. In `$GUI_CONFDIR/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.

IBM Platform Analytics reports in Platform Application Center

About IBM Platform Analytics in Platform Application Center

Platform Application Center embeds IBM Platform Analytics (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 same location as Platform Analytics.

Access to reports:

- When the add-on has been installed, the reports in Platform 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	Platform Application Center automatically loads Tableau Licensed users with the new role Report administrator.A user with the Report administrator role can: • 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	A user with the Normal user role can: • View past reports • Subscribe to a report, or unsubscribe from a report to receive emails when reports are updated
Cluster administrator	A user with the Cluster administrator role can: • 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

About this task

Requirements:

- Platform Analytics is installed in your IBM Platform LSF(LSF) cluster.
- You have an IBM Platform Application Center Standard Edition.
- Platform Application Center is installed and can access $LSF_ENVDIR.
- You have copied the `tableau.conf` file created by the Platform Analytics installation to the Platform Application Center web server. The `tableau.conf` file is created in the *PA_SERVER_TOP*`\reports\conf` directory on the Platform Analytics Reporting Server. For more details on this file, refer to the Platform Analytics documentation *Integrating Platform Analytics into Platform Application Center*.
- The Vertica client driver is installed on your Tableau server. You can download the driver package from 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 same location as Platform Analytics.

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

Procedure

1. As the LSF administrator, log on to your Platform Application Center host.
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. Run the command **tableauconfig.sh enable -f tableau.conf**.
4. Restart the web server.

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

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

   You should be able to see the Platform Analytics reports.

Create an Oracle database schema

About this task

Follow these instructions to create a database schema in Oracle for Platform Application Center data.

Procedure

1. Untar the schema package.

   ```
   tar -xvf pcc-appcenter-9.1-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 Platform Application Center database schema.

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

6. If you are migrating your Platform Application Center MySQL database to Oracle, follow the data migration instructions at:

   `http://www.oracle.com/technetwork/developer-tools/sql-developer/omwb-getstarted-093461.html`

Configure the database connection

Follow these instructions to write the database connection string in the Platform Application Center configuration file `$PERF_TOP/conf/datasource.xml` with encrypted passwords.

Before you begin

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

Procedure

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.

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 IBM 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 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 IBM 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

About this task

Note:

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

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

Procedure

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 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: `$EGO_TOP/eservice/esd/conf/esddefault.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.">rUlWkhrNFCsXkOwZBu/xVA==</ESD_EGO_KEY>
  <!-- CORP DNS domain TSIG key created by dnssec-keygen -->
  <ESD_CORP_KEY name="ego."> e1rBv20yZFhOsUMyL76wqQ==</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

About this task

Note:

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

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

Procedure

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:
     a. From the Windows **Start** menu, click **Control Panel**, and select **Network Connections**.
     b. Right-click **Local Area Connection**, and then select **Properties**.
     c. Select Internet Protocol (TCP/IP) from the list of protocols, and then click **Properties**.
     d. On the General page, click **Advanced**.
     e. Click the DNS tab.
     f. 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.

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 IBM Platform Process Manager, IBM Platform Analytics, and IBM 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	Logs all debug-level messages.At this log level, all DEBUG, INFO, WARN, ERROR, and FATAL messages are logged.
INFO	Logs all informational messages about events that occurred. For example, Platform Application Center successfully started.At this log level, all INFO, WARN, ERROR, and FATAL messages are logged.
WARN	Logs only those messages that are potentially harmful. The system can tolerate this error. Features and performance are not affected.At this log level, all WARN, ERROR, and FATAL messages are logged.
ERROR	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.For example, cannot create a process to execute a job, failed to create directories because of disk quota, etc.At this log level, all ERROR and FATAL messages are logged.
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.

Chapter 7. Tuning

When is tuning required?

The default Platform Application Center configuration works for small clusters. If you have a medium or large cluster, you will need to tune Platform Application Center for optimal performance.

The cluster sizes outlined below describe the maximum number of jobs, hosts, users for a specific cluster size category.

Item	Small cluster (Up to)	Medium cluster (Up to)	Large cluster (Up to)
Active jobs	5000	50,000	500,000
Job throughput	10,000 jobs/day	100,000 jobs/day	1 million jobs/day
Hosts	100	1000	6000
Active users	50	200	1000
Concurrent users	10	40	200
Time to keep job information and data (default)	14 days	14 days	14 days
Minimum recommended hardware for the Platform Application Center web server	1 CPU, 4 cores each Memory: 8 GB Disk type: Normal	2 CPUs, 4 cores each Memory: 16 GB Disk type: Faster	2 CPUs, 8 cores each Memory: 24 GB Disk type: Faster
Tuning required?	No	Yes	Yes

Database

Recommended database MySQL 5.6

MySQL

At installation

- If the file `etc/my.cnf` exists on the Platform Application web server, nothing is changed in that file.

  Refer to the file `PAC_TOP/gui/3.0/etc/my.cnf.mysql_tuned` and make the appropriate changes to your `etc/my.cnf` file. The file `PAC_TOP/gui/3.0/etc/my.cnf.mysql_tuned` is tuned for medium clusters.
- If the file `etc/my.cnf` does not exist on the Platform Application web server, the new file `/etc/my.cnf` is added. This file is tuned for medium clusters.

Data loaders

The amount of data logged and loaded into the database from LSF affects how quickly a job is displayed in the **Jobs** page after it is submitted and job information querying.

The following settings reduce the number of events that need to be loaded, and as such increase the speed at which data is displayed in the web interface.

Reduce the number of LSF events logged

If you are not planning on using Standard reports, follow these steps. Standard reports will not work after this configuration is done.

If you are using Platform Analytics for reporting, check the Platform Analytics documentation for additional event types that you may need to add to the ALLOW_EVENT_TYPE parameter.

1. Log on to the LSF master host as the LSF administrator.
2. Set the LSF environment.
3. Edit `lsb.params`.
4. For the parameter ALLOW_EVENT_TYPE, only define the following events. LSF will only log these events.

```
ALLOW_EVENT_TYPE=JOB_NEW JOB_STATUS JOB_FINISH2 JOB_START JOB_EXECUTE JOB_EXT_MSG JOB_SIGNAL JOB_REQUEUE JOB_MODIFY2
JOB_SWITCH METRIC_LOG
```

5. Run **badmin reconfig** to reconfigure mbatchd.

Disable loaders for both Standard reports and Scheduler Dashboard

If you are not planning on using Standard reports or the Scheduler Dashboard, you can disable the LSF event loader.

Neither Standard reports nor the Scheduler Dashboard will work after this configuration is done.

1. Log on to the Platform Application Center web server host as root.
2. Set the Platform Application Center environment.
3. Open `$PERF_CONFDIR/plc/plc_lsf.xml`
4. Set the `lsfeventloader` to `Enable="false"`:

```
<DataLoader Name="lsfeventsloader" Interval="60" Enable="false" LoadXML="dataloader/lsbevents.xml"/>
```

5. Restart the plc service.

   If EGO is not enabled:

   a. In the command console, stop the plc service.

      ```
      perfadmin stop plc
      ```

   b. Start the plc service.

      ```
      perfadmin start plc
      ```

   If failover with EGO is enabled:

   a. In the command console, stop the plc service.

      ```
      egosh service stop plc
      ```

   b. Start the plc service.

      ```
      egosh service start plc
      ```

Disable loaders only for Standard reports

If you want to use the Scheduler Dashboard but do not need Standard reports, complete these steps.

Standard reports will not work after this configuration is done.

1. Log on to the Platform Application Center web server host as root.
2. Set the Platform Application Center environment.
3. Open `$PERF_CONFDIR/dataloader/lsbevents.properties`.
4. For ALLOW_EVENT_TYPE, remove everything and only enable the metric log:
   `ALLOW_EVENT_TYPE=METRIC_LOG`
5. Restart the plc service.
   If EGO is not enabled:
   a. In the command console, stop the plc service.
      ```
      perfadmin stop plc
      ```
   b. Start the plc service.
      ```
      perfadmin start plc
      ```
   If failover with EGO is enabled:
   a. In the command console, stop the plc service.
      ```
      egosh service stop plc
      ```
   b. Start the plc service.
      ```
      egosh service start plc
      ```

Increase data loader memory so that the loader works properly

1. Log on to the Platform Application Center web server host as root.
2. Set the Platform Application Center environment.
3. Open `$PERF_CONFDIR/wsm/wsm_plc.conf`.
4. Set the memory in the parameter JAVA_OPTS to a number larger than the default.
   Change the `-Xms` and `-Xmx` numbers to a higher number. These numbers are in MB and set the minimum and maximum memory that the loader can use. If you reach the maximum number, Java throws an exception.
   - For medium clusters, 50,000 active jobs and a throughput of 10 jobs/second, a maximum memory of 6 GB is required.
   - For large clusters, 500,00 active jobs, and a throughput of 30 jobs/second, a maximum memory of 14 GB is required.

   For example:
   ```
   JAVA_OPTS=-Xms1024m -Xmx14336m -Dcom.sun.management.jmxremote=true
   ```
5. Restart the plc service.
   If EGO is not enabled:
   a. In the command console, stop the plc service.
      ```
      perfadmin stop plc
      ```
   b. Start the plc service.
      ```
      perfadmin start plc
      ```
   If failover with EGO is enabled:
   a. In the command console, stop the plc service.
      ```
      egosh service stop plc
      ```
   b. Start the plc service.
      ```
      egosh service start plc
      ```
6. Go into Platform Application Center and navigate to the **Jobs** page.
   See how quickly data loads into the page.
   If you need data to load faster, increase the memory again in JAVA_OPTS until you reach the desired data loading speed.

Web server

Increase the memory on the web server

1. Open `$GUI_CONFDIR/wsm_webgui.conf` and set MEM_HIGH_MARK and JAVA_OPTS.
   a. Set MEM_HIGH_MARK to a number higher than 3000 if you have a large number of jobs.

      This parameter indicates the maximum memory the web server can reach in MB, before the process is terminated.
      - For medium clusters, the default 3 GB is adequate.
      - For large clusters, set this parameter to 8 GB.

   b. If you change MEM_HIGH_MARK to a higher number, you must set a higher number for JAVA_OPTS.

   For example, if you change HIGH_MEM_MARK to 8 GB, you need to change the JAVA_OPTS maximum memory -Xmx to 8 GB.

   ```
   MEM_HIGH_MARK=8000
   JAVA_OPTS="-server -Xms2048m -Xmx8000m -XX:NewSize=128m -XX:MaxNewSize=256m -XX:PermSize=128m -XX
   -Xgcpolicy:gencon"
   ```
2. 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.

Set the number of concurrent users

The default configuration of 500 threads, set with the parameter CATALINA_MAX_THREADS in `$GUI_CONFDIR/wsm_webgui.conf`, is adequate for 200 concurrent users.

If you have more users that are going to log on to Platform Application Center, you need to increase the value of this parameter so that new connections do not fail.

As a rule of thumb, each user uses 2 to 3 threads, so set the parameter CATALINA_MAX_THREADS to serve the number of concurrent users at your site. For example, if you have 300 concurrent users, you would set CATALINA_MAX_THREADS=600.

1. Open `$GUI_CONFDIR/wsm_webgui.conf`.
2. Set the number of concurrent users with CATALINA_MAX_THREADS.

   For example:

   ```
   CATALINA_MAX_THREADS=600
   ```
3. 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.

Purge data more often

How much data is in the database affects job querying performance.

The parameter that controls job information and data purging for done and exited jobs is FINISH_JOB_TIME_TO_LIVE in `$GUI_CONFDIR/pmc.conf`.

The default for FINISH_JOB_TIME_TO_LIVE is 14 days. Make it smaller to reduce the database table size and speed up job querying.

Note that the value of FINISH_JOB_TIME_TO_LIVE must be larger than 1 day and larger than the parameter MOVE_FINISHED_JOBS_AFTER in `$PERF_CONFDIR/dataloader/commonjobloader.properties`.

The value you set for FINISH_JOB_TIME_TO_LIVE depends on your users' expectations and for how long users need to keep information and data for done and exited jobs.

If in doubt, set this parameter to a higher number because after the FINISH_JOB_TIME_TO_LIVE time period is reached, job information and data is permanently removed and cannot be recovered.

1. Open `$GUI_CONFDIR/pmc.conf`.
2. Reduce the number of days from 14 to any desired number.

   For example:

   ```
   FINISH_JOB_TIME_TO_LIVE=10
   ```
3. 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.

Disable job status change notifications

By default, job status change notifications are disabled. If you enable job status change notifications, the system periodically checks the job status, making the database and the web server very busy. It is best to disable job status change notifications if you do not use this feature.

1. Open `$GUI_CONFDIR/pmc.conf`.
2. Set the parameter ENABLE_JOB_NOTIFICATION=N.
3. 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.

Disable loading of LSF user groups

By default, Platform Application Center automatically loads user groups defined in LSF. This is controlled with the parameter ENABLE_USERGROUP in `$GUI_CONFDIR/pmc.conf`.

LSF user groups are loaded when Platform Application Center is started up.

If LSF has complex user groups, this information can take a long time to load into Platform Application Center.

If you are not using LSF user groups to control job permissions in Platform Application Center, you can disable automatic loading of LSF user groups.

Note that after you disable this feature, you will no longer be able to manage access to Platform Application Center pages and features by using user groups in Platform Application Center.

1. Open `$GUI_CONFDIR/pmc.conf`.
2. Set ENABLE_USERGROUP=N.
3. 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.

Increase the number of file descriptors for the web server

To meet the performance requirements of a large cluster, increase the maximum number of open files for the web server.

1. Open `PAC_TOP/gui/3.0/bin/pmcadmin`.
2. Add ulimit -n 65535 before CATALINA_OPTS:

```
# ------------------------------------------------------------------------------
#set the init environment
#-------------------------------------
#show report without x11
ulimit -n 65535
CATALINA_OPTS="-Djava.awt.headless=true"
```

Chapter 8. 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.

Reporting with Standard Reports

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 Platform Application Center.

Standard and custom reports

A set of built-in 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

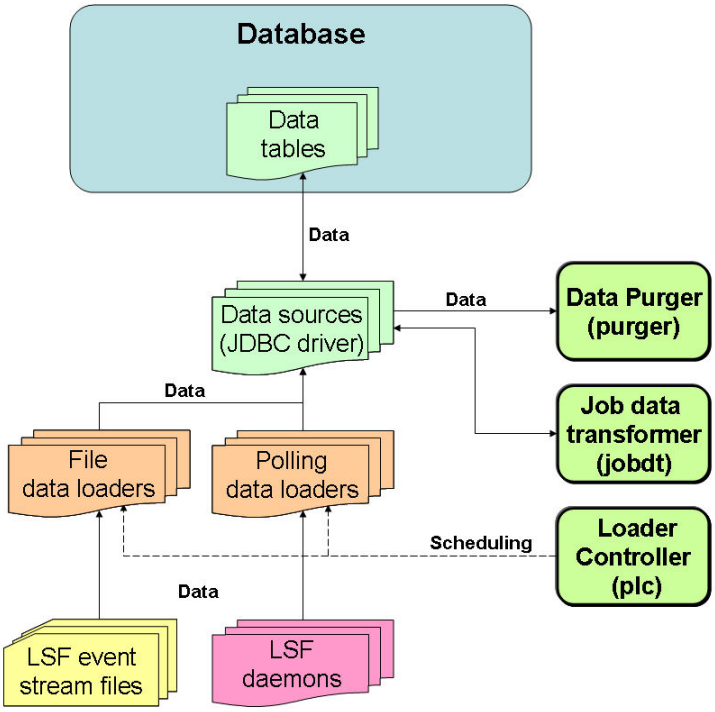
The reporting feature includes the MySQL database, a small-footprint, open source database. If you want to use the reporting feature to produce regular reports for a production cluster, you can also use a supported commercial database. The reporting feature supports Oracle databases.

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:

Data loader name	Data type	Data gathering interval	Data loads to	Loader type
License Scheduler (**bldloader**)	license usage	5 minutes	BLD_LICUSAGE	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 (**lsfbhostsloader**)	host utilization and state-related	5 minutes	LSF_BHOSTS	polling
LSF events (**lsfeventsloader**)	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 1. 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
Dynamic metric (**egodynamicresloader**)	host-related dynamic metric	5 minutes	RESOURCE_METRICS RESOURCES_RESOURCE _METRICS	polling
EGO allocation events (**egoeventsloader**)	resource allocation	5 minutes	ALLOCATION_EVENT	file
Static attribute (**egostaticresloader**)	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).

Standard reports

For your convenience, several standard reports are provided 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 2. 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

Table 2. Standard reports (continued)

Name	Description	Category
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

Procedure

1. In the Console, navigate to **Reports**, then **Job Reports** or **Resource Reports**.
2. Click the name of your report to display the description.

Produce a standard report

About this task

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.

Procedure

1. In the Console, navigate to **Reports**, then **Job Reports** or **Resource Reports**.
2. Click the name of your report to select it.
3. Click **Produce Report**.

   The Produce Report dialog is displayed.
4. Set the report parameters as desired and click **Produce Report**.

   You report is generated.

Results

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

Export report data

About this task

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.

Procedure

1. Produce and view your report.
2. In the **Produce Report** window, click **Export Data**.
3. In the browser dialog, select **Open with** or **Save File**.

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, 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 when you produce a report.

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

About this task

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.

Procedure

1. Select the report that you want to copy.
2. Click **Produce Report**.

   The Produce Report dialog is displayed.
3. In the **Produce Report** window, click **Copy to New Custom Report**.
4. 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 **Data Schema Reference** in the online help.
   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.
5. To finish, click **Create**.

Results

To access your new custom report, select it from the report list. Custom reports have the type **Custom**.

Modify a custom report

Procedure

1. Navigate to **Reports** then **Job Reports** or **Resource Reports**.
2. Click the name of your report.
3. Click the **Modify** button.

The **Modify** window is displayed.

4. Make the desired changes to the report.
   a. Edit the report properties and SQL query string.

      For further information on the data schema, refer to **Data Schema Reference** in the online help.
   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.
5. To confirm your changes, click **OK**.

Produce a custom report

Procedure

1. Navigate to **Reports**, then **Job Reports** or **Resource Reports**.
2. Click the name of your report to select it.
3. Click **Produce Report**.

   The Produce Report dialog is displayed.
4. Set the report parameters as desired and click **Produce Report**.

   You report is generated.

Results

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

Export report data

About this task

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.

Procedure

1. Produce and view your report.
2. In the **Produce Report** window, click **Export Data**.
3. In the browser dialog, select **Open with** or **Save File**.

Delete a custom report

Procedure

1. Navigate to **Reports**, then **Job Reports** or **Resource Reports**..
2. Locate your report in the list.
3. Click the **Delete** button.

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
`$PERF_TOP`	Reports framework directory	*PAC_TOP*/perf
`$PERF_CONFDIR`	Configuration files	*PAC_TOP*/perf/conf/
`$PERF_LOGDIR`	Log files	*PAC_TOP*/perf/log
`$PERF_WORKDIR`	Working directory	*PAC_TOP*/perf/work
`$PERF_DATADIR`	Data directory	*PAC_TOP*/perf/data

Reporting services

The reporting feature uses the following services.

- Job data transformer (**jobdt**)
- Loader controller (**plc**)
- 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 **plc** 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 `jobdt.xml` configuration file, but you need to restart the **jobdt** service and EGO on the LSF master host after editing the file. The `jobdt.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:

- **bldloader** (License Scheduler data loader)
- **desktopjobdataloader** (Desktop job data loader)
- **desktopclientdataloader** (Desktop client data loader)
- **desktopeventloader** (Desktop active event data loader)
- **egoconsumerresloader** (consumer resource data loader)
- **egodynamicresloader** (dynamic metric data loader)
- **egoeventsloader** (EGO allocation events data loader)
- **egostaticresloader** (static attribute data loader)
- **lsfbhostsloader** (bhosts data loader)
- **lsfeventsloader** (LSF events data loader)
- **lsfslaloader** (SLA data loader)
- **lsfresproploader** (LSF resource properties data loader)
- **sharedresusageloader** (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 **plc** 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 **plc** configuration files, you need to restart the **plc** service.

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

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 or 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_service.xml` service configuration file, but you must restart the **purger** service and EGO on the LSF master host after editing the file. The `purger_service.xml` file is located in the EGO service directory: *LSF_CONFDIR*/ego/*cluster_name*/eservice/esc/conf/services.

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

About this task

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.

Procedure

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)

Before you begin

Your cluster must have PERF controlled by EGO.

About this task

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.

Procedure

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)

Before you begin

Your cluster must have PERF not controlled by EGO.

About this task

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.

Procedure

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

About this task

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

Procedure

Launch the loader controller client tool with the **-s** option.

- In UNIX, run `$PERF_TOP/`*version_number*`/bin/plcclient.sh -s`.
  For example:
  `$PERF_TOP/1.2/bin/plcclient.sh -s`
- In Windows, run `%PERF_TOP%\`*version_number*`\bin\plcclient -s`.

For example:

```
%PERF_TOP%\1.2\bin\plcclient -s
```

Dynamically change the log level of your loader controller log file

About this task

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.

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

Procedure

Launch the loader controller client tool with the **-l** option.

- In UNIX, run `$PERF_TOP/`*`version_number`*`/bin/plcclient.sh -l` *`log_level`*.

  For example:

  ```
  $PERF_TOP/1.2/bin/plcclient.sh -l WARN
  ```

- In Windows, run `%PERF_TOP%\`*`version_number`*`\bin\plcclient -l` *`log_level`*.

  For example:

  ```
  %PERF_TOP%\1.2\bin\plcclient -l WARN
  ```

Dynamically change the log level of your data loader log files

About this task

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.

Procedure

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

About this task

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

Procedure

1. Edit the `log4j.properties` file, located in the reports configuration directory (*PERF_CONFDIR*).
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

About this task

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.

Procedure

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.

   `badmin reconfig`

Change the location of the LSF event data files

About this task

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

Procedure

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.

   ```
   badmin 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

Before you begin

Your cluster must be EGO-enabled.

About this task

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.

Procedure

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

Before you begin

Your cluster must be EGO-enabled.

About this task

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

Procedure

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

Before you begin

Your cluster must be EGO-enabled.

About this task

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.

Procedure

1. Edit `jobdt.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> ...jobdt.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 **jobdt** 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> ...jobdt... -t 22:20
     ```

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

     ```
     <ego:Command> ...jobdt... -t *:25
     ```

   - To convert job data every fifteen minutes from when the **jobdt** service first starts up:

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

   - To convert job data every two hours from when the **jobdt** service first starts up:

     ```
     <ego:Command> ...jobdt... -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 **jobdt** service.

Change the default record expiry time

About this task

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.

Procedure

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. 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

About this task

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.

Procedure

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

About this task

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

Procedure

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="egodynamicresloader" 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="egodynamicresloader" Interval="200" ... />
   ```
4. Restart the **plc** service.

Disable data collection for individual data loaders

About this task

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

Procedure

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 disable.

   For example:

   ```
   <DataLoader Name="egodynamicresloader" ... Enable="true" .../>
   ```
3. Edit the `Enable` attribute to "false".

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

   ```
   <DataLoader Name="egodynamicresloader" ... Enable="false" ... />
   ```
4. Restart the **plc** service.

Test the reporting feature

Verify that components of the reporting feature are functioning properly.

Procedure

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/plc.log.`*`host_name`*
      - Windows: `%PERF_LOGDIR%/plc.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:
      - `bldloader`
      - `desktopjobdataloader`
      - `desktopclientdataloader`
      - `desktopeventloader`
      - `lsfbhostsloader`
      - `lsfeventsloader`
      - `lsfslaloader`
      - `lsfrespropoloader`
      - `sharedresusageloader`
      - EGO data loaders (for EGO-enabled clusters only):
        - `egoconsumerresloader`
        - `egodynamicresloader`
        - `egoeventsloader`
        - `egostaticresloader`
   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`*`/dataloader/bldloader.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/dataloader/desktopjobdataloader.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/dataloader/desktopclientdataloader.`*`host_name`*`. log`
      - *`PERF_LOGDIR`*`/dataloader/desktopeventloader.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/jobdt.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/dataloader/lsfbhostsloader.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/dataloader/lsfeventsloader.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/dataloader/lsfslaloader.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/purger.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/dataloader/lsfrespropoloader.`*`host_name`*`.log`
      - *`PERF_LOGDIR`*`/dataloader/sharedresusageloader.`*`host_name`*`.log`
      - EGO data loader log files (EGO-enabled clusters only):
        - *`PERF_LOGDIR`*`/dataloader/egoconsumerresloader.`*`host_name`*`.log`
        - *`PERF_LOGDIR`*`/dataloader/egodynamicresloader.`*`host_name`*`.log`
        - *`PERF_LOGDIR`*`/dataloader/egoeventsloader.`*`host_name`*`.log`
        - *`PERF_LOGDIR`*`/dataloader/egostaticresloader.`*`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.

What to do next

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

Chapter 9. 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 can be Linux or Windows, 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.

Requirements

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_webgui.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** 2.0.6 (`http://code.google.com/p/httplib2/downloads/list`)

Get 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 logon**. The connection information and security information is be saved in `$HOME/.pacpass`.
2. To perform a task such as listing jobs or submitting a job, run the corresponding command, such as **pacclient.py job** or **pacclient.py submit**.

3. To log out, run **pacclient.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**

Description

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

-h

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

Subcommand synopsis

pacclient.py ping

pacclient.py ping -l *URL*

pacclient.py logon

pacclient.py logon -l *URL* **-u** *user_name* **-p** *password*

pacclient.py logout

pacclient.py logout

pacclient.py app

pacclient.py app -l | **-p** *app_name*[**:***template_name*]

pacclient.py submit

pacclient.py submit -a *app_name*[**:***template_name*] [**-c** *file_path*] [**-p** *field_ID*=value**;**[*field_ID*=*value***;**]]

pacclient.py job

pacclient.py job [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*] *job_ID* | "*job_ID*[*index_list*]"

pacclient.py job -s Pending|**Running**|**Done**|**Exit**|**Suspended** [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*]

pacclient.py job [**-l**] *job_ID* | "*job_ID*[*array_index*]"

pacclient.py job -n *job_name* [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*]

pacclient.py job -g *job_group_name* [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*]

pacclient.py job -h

pacclient.py jobaction

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

pacclient.py jobdata

pacclient.py jobdata -l *job_ID*

pacclient.py download

pacclient.py download [**-d** *dir_name*] [**-f** *file_name*] *job_ID*

pacclient.py upload

pacclient.py upload [**-d** *dir_name*] **-f** *file_name*[,*file_name*..] *job_ID*

pacclient.py upload -d *host_name*:*dir_name* **-f** *file_name*[,*file_name*..]

pacclient.py upload -h

pacclient.py usercmd

pacclient.py usercmd -c *user_command*

pacclient.py ping

Synopsis

pacclient.py ping -l *URL*

Description

Detects whether or not the web service is running at the specified URL. If no parameters are specified, prompts for the URL interactively.

Options

-l *URL*

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

`http://host_name:port_number/`

Examples

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

pacclient.py logon

Synopsis

pacclient.py logon -l *URL* **-u** *user_name* **-p** *password*

Description

Logs you in to Platform Application Center.

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.

Options

-l *URL*

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

`http://pac_host:port_number/`

-u *user_name*

Specify your user name.

-p *password*

Specify your password.

Examples

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

pacclient.py logout

Synopsis

pacclient.py logout

Description

Logs you out.

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.

Examples

```
pacclient.py logout
```

pacclient.py app

Synopsis

pacclient.py app -l | **-p** *app_name*[**:***template_name*]

Description

List applications or parameters of an application.

Options

-l

List all applications and application status.

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

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

Examples

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

pacclient.py submit

Synopsis

pacclient.py submit -a *app_name*[**:***template_name*] [**-c** *file_path*] [**-p** *field_ID*=value**;**[*field_ID*=*value*;]]

Description

Submits a job.

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).

Options

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

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

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

[-c *file_path* **]**

Set 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 in Platform Application Center 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
[Inputfile]
FLUENT_JOURNAL=C:\demo\fluent\fluent-test.jou,upload
CAS_INPUT_FILE=C:\demo\fluent\fluent-test.cas.gz,upload
```

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

Define the job submission parameters on the command line.

Specify field ID and value pairs as described for the **-c** option.

Examples

```
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
```

pacclient.py job

Synopsis

pacclient.py job [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*] *job_ID* | "*job_ID*[*index_list*]"

pacclient.py job -s Pending|**Running**|**Done**|**Exit**|**Suspended** [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*]

pacclient.py job [**-l**] *job_ID* | "*job_ID*[*array_index*]"

pacclient.py job -n *job_name* [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*]

pacclient.py job -g *job_group_name* [**-l**] [**-u** *user_name* | **-u all**] [**-p** *hours*]

pacclient.py job -h

Description

Displays information for one or more jobs.

By default, displays information about pending, running, and suspended jobs submitted by the user running this command. By default, also displays information about done and exited jobs submitted by the user running this command that have ended within the past hour.

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 by clicking the **Options** button.

Options

-l

Long format. Displays detailed information for each job, job array, or job group in a multiline format.

-u *user_name* | **-u all**

Only displays information about jobs that have been submitted by the specified user. The keyword all specifies all users.

-p *hours*

In addition to pending, running, and suspended jobs, displays information about all done and exited jobs that have ended within the specified number of hours. If the number of hours is not specified, displays information about done and exited jobs submitted by the user running this command that have ended within the past hour.

-s Pending|Running|Done|Exit|Suspended

Displays information only about jobs that have the specified state.

-n *job_name*

Displays information about jobs or job arrays with the specified job name. The wildcard character (*) can be used within a job name, but cannot appear within array indices. For example job* returns jobA and jobarray[1], *AAA*[1] returns the first element in all job arrays with names containing AAA, however job1[*] will not return anything since the wildcard is within the array index.

p *job_ID*

Display information for the job with the specified job ID.

-g *job_group_name*

Display information for jobs in the specified job group.

-h

Display subcommand usage.

Examples

Get information about jobs submitted by the user who is logged on

```
pacclient.py job
```

Get information about jobs submitted by user user1

```
pacclient.py job -u user1
```

Get detailed information about job 11919

```
pacclient.py job -l 11919
```

Get detailed information about jobs with the job name TEST

```
pacclient.py job -l -n TEST
```

Get detailed information about jobs with the status Running

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

Get information about the job array element 1 in job array 11921

```
pacclient.py job "11921[1]"
```

Get information about jobs in job group /dev

```
pacclient.py job -g "/dev"
```

Get information about jobs with the status Exit that have ended within the last 5 hours

```
pacclient.py job -s Exit -p 5
```

Get information about jobs submitted by user user2 with the status Done that have ended within the last hour

```
pacclient.py job -u user2 -s Done -p 1
```

pacclient.py jobaction

Synopsis

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

Description

Control a job.

Options

-a kill|suspend|requeue|resume

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

job_ID

Specify the job ID.

Examples

```
pacclient.py jobaction -a resume 54321
```

pacclient.py jobdata

pacclient.py jobdata -l *job_ID*

Description

List input and output files for a job.

Options

-l *job_ID*

Specify the ID of the job for which to list files.

Examples

List files for job with ID 1234

```
pacclient.py jobdata -l 1234
```

pacclient.py upload

Synopsis

pacclient.py upload [**-d** *dir_name*] **-f** *file_name*[,*file_name*..] *job_ID*

pacclient.py upload -d *host_name*:*dir_name* **-f** *file_name*[,*file_name*..]

pacclient.py upload -h

Description

Uploads data for a job to the job directory on the web server or the remote directory on a specific host.

By default, if a job ID is specified but no directory is specified, files are uploaded to the job directory on the web server.

Options

-d *dir_name*

Copies files to the specified directory on the web server. The directory can be an absolute path on the web server or a relative path to the job directory on the web server.

-d *host_name*:*dir_name*

Uploads files to the specified job directory on the specified host. The directory must be an absolute path to the remote job directory on the host. The host can be any LSF compute host.

-f *file_name*[,*file_name*...]

Uploads the specified file. Specify the path. Separate multiple files with a comma. If no path is specified for the file, the job directory on the web server is used as the location of the file.

job_ID

Only used to upload files to the job directory on the web server. The ID of the job for which to upload files. The job must have the state Running. The ID must be a positive integer.

-h

Display subcommand usage.

Examples

Upload files to the job directory on the web server

```
pacclient.py upload -f /dir1/file1,/dir2/file2 54321
```

Upload file1 into the result subdirectory under job 101's job directory

```
pacclient.py upload -d result -f file1 101
```

Upload file1 into /tmp/result on the web server

```
pacclient.py upload -d /tmp/result -f file1
```

Upload file1 into /tmp/result on the remote host host1

```
pacclient.py upload -d  host1:/tmp/result -f file1
```

pacclient.py download

Synopsis

pacclient.py download [**-d** *dir_name*] [**-f** *file_name*] *job_ID*

Description

Download job data for a job.

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

Options

[-d *directory***]**

Copy files to the specified directory.

[-f *file_name***]**

Downloads the specified file only.

job_ID

Specify the job ID.

Examples

For example:

```
pacclient.py download 54321
```

pacclient.py usercmd

Synopsis

pacclient.py usercmd -c *user_command*

Description

Runs the specified command as the user who is logged on to the web server host.

You list commands in the configuration file `$GUI_CONFDIR/webservice.usercmd`.

By default, the following LSF commands are listed in this file:

- **bbot**
- **bchkpnt**
- **bkill**
- **bpost**
- **brequeue**
- **brestart**
- **bresume**
- **brun**
- **bstop**
- **btop**
- **bswitch**
- **bmig**

You can also add any other commands to the file that you want to run such as for example, **ls** or **hostname**.

Options

-c *user_command*

Specify the executable command with its full path and options. The command must be listed in the configuration file `$GUI_CONFDIR/webservice.usercmd`.

-h

Display subcommand usage.

Examples

Kill job 101 by sending a SIGTERM signal

```
pacclient.py usercmd -c bkill -s SIGTERM 101
```

Force checkpoint of job 201 every 5 minutes

```
pacclient.py usercmd -c bchkpnt -f -p 5 201
```

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.

The token is an encrypted string for single sign-on and includes user name, password, and expiry data. You cannot modify the contents of the token or parse them.

The following is an example of a token that is returned:

```
platform_token=lsfadmin"2010-04-03T09:31:02Z"UihSStsCCyv9yXJ+1hyiaJXrHr6wtc4ZT+V0lVWKEe99L3Ht4RJBXr7h
fzd0nNaJWVmWP5I2kyuPZZAcFDIalVwAlSOmhqKLcwLcAfadHpgB1LYf3/g1V8pDUIngGbCe"054LJVyt
YAa7QagStQiqdw==
```

When using the token, you need to encrypt double quotes("). For example:

```
lsfadmin#quote#2010-04-03T09:31:02Z#quote#UihSStsCCyv9yXJ+1hyiaJXrHr6wtc4ZT+V0lVWKEe99L3Ht4RJBXr7h
fzd0nNaJWVmWP5I2kyuPZZAcFDIalVwAlSOmhqKLcwLcAfadHpgB1LYf3/g1V8pDUIngGbCe#quote#054LJVyt
YAa7QagStQiqdw==
```

After all web service requests are done, the client must call the logout service to release the token. Otherwise, the token will be occupied for a period of time (by default 60 minutes), then Platform Application Center will log the user out and release the token 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=`platform/webservice/pacclient/`*function*

For example:

```
http://host1.my.company.com:8080/platform/webservice/pacclient/logon
```

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

- CATALINA_START_PORT=8080
- CATALINA_HTTPS_START_PORT=8443

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>/platform/webservice/pacclient/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://xxxx: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>/platform/webservice/pacclient/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	

HTTP Response	Field format/value	Notes
Success Message	Web Services are ready on URL: http://xxxx: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>/platform/webservice/pacclient/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>/platform/webservice/pacclient/logonPost	

HTTP Request Field	Field format/value	Notes
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>/platform/webservice/pacclient/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	
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>/platform/webservice/pacclient/appStatus	
Content-Type	text/plain	Also supports application/json
Accept	text/xml	Also supports application/json
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>`
	`  <appName>%s</appName>`
	`  <status>%s</status>`
	`  </AppInfo>`
	`  <AppInfo>`
	`  <appName>%s</appName>`
	`  <status>%s</status>`
	`  </AppInfo>`
	`</AppInfos>`	

Failure Message	`<AppInfos>`
	` <AppInfo>`
	`  <errMsg>%s</errMsg>`
	` </AppInfo>`
	`</AppInfos>`
		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>/platform/webservice/pacclient/appParams	
Content-Type	text/plain	Also supports application/json
Accept	text/xml	Also supports application/json
Cookie	platform_token=<token>	<token>: returned from logon

HTTP Request Field	Field format/value	Notes
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>`
	`  <AppInfo>...</AppInfo>`
	`</AppParams>`	

Failure Message	`<AppParams>`
	`  <AppParam>`
	`  <errMsg>%s</errMsg>`
	`  </AppParam>`
	`</AppParams>`
		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>/platform/webservice/pacclient/appParamsPost	
Content-Type	text/plain	
Accept	text/xml	
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	

HTTP Response	Field format/value	Notes

Success Message	`<AppParams>`
	`  <AppParam>`
	`  <id>%s</id>`
	`  <label>%s</label>`
	`  <mandatory>%s</mandatory>`
	`  <defaultValue>%s</defaultValue>`
	`  </AppParam>`
	`  <AppParam>...</AppParam>`
	`</AppParams>`	

Failure Message	`<AppParams>`
	`  <AppParam>`
	`  <errMsg>%s</errMsg>`
	`  </AppParam>`
	`</AppParams>`
		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.
- Just like through the web graphical interface, when you submit a job, a temporary directory is created for the job and the file you attached is copied to the directory. The owner for the directory is the user that is logged on. The directory inherits the logged on user's `umask` permission on the Platform Application Center host.

Request

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

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

```
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	

Field Name	Field format/value	Notes
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.jou>
"\r\n"
PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA+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 Jobs

Purpose

The get jobs service returns short or detailed information for jobs and job arrays with the specified job ID, job name, job status. It also returns jobs and job arrays that belong to a specified job group, were submitted by a specified user, or that have finished within a specified time period. By default, when no parameters are specified, returns short information for all jobs and job arrays submitted by the user who is currently logged on.

The information returned is similar to the information that can be retrieved with the LSF commands **bjobs** and **bhist**.

Request

HTTP Request Field	Field format/value	Description
HTTP Method	GET	

HTTP Request Field	Field format/value	Description

URL	http://<hostname>:<port>/platform/webservice/pacclient/jobs
	
	http://<hostname>:<port>/platform/webservice/pacclient/jobs?id=<id>
	
	http://<hostname>:<port>/platform/webservice/pacclient/jobs?status=<status>
	
	http://<hostname>:<port>/platform/webservice/pacclient/jobs?name=<name>
	
	http://<hostname>:<port>/platform/webservice/pacclient/jobs?group=/<group>
	
	http://<hostname>:<port>/platform/webservice/pacclient/jobs?user=<user>
	
	http://<hostname>:<port>/platform/webservice/pacclient/jobs?details=<details>
	
	http://<hostname>:<port>/platform/webservice/pacclient/jobs?status=<status>&past=<past>
		Parameters can be combined together with &.
		
		Before passing a parameter, use UTF-8 to encode the value. For example:
		
		`id=encode(201[1])`
		
		`group=encode(/test)`
		
		Parameter descriptions:
		• <id>: Job ID or job array ID. Integer.
		• <status>: Job status. String. Possible values:
		– Pending
		– Suspended
		– Running
		– Exit
		– Done
		• <name>: Name of job. String.
		• <group>: Name of job group to which the job belongs. String.
		• <user>: Name of user for which to display job information. The logged in user must have permission to view the jobs. Specify the keyword `all` to retrieve information for all jobs of all users.
		• <details>: Return detailed information about a job. Valid values:
		– yes
		– no
		• <past>: Number of hours. Integer. Can only be used with the job status Done or Exit. Retrieves information for jobs that have finished within the specified number of hours.

Content-Type	application/xml	Also supports application/json
Accept	text/xml	Also supports application/json
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	Short format:
	`<Jobs>`
	`  <Job>`
	`  <name>%s</name>`
	`  <id>%s</id>`
	`  <status>%s</status>`
	`  <cmd>%s</cmd>`
	`  </Job>`
	`  <Job>`
	`  ...`
	`  </Job>`
	`</Jobs>`
	
	Detailed format:
	`<Jobs>`
	`  <Job>`
	`  <name>%s</name>`
	`  <id>%s</id>`
	`  <status>%s</status>`
	`  <cmd>%s</cmd>`
	`  <cwd>/home/lsfadmin/generic_1342080220286ST0Yx</cwd>`
	`  <description>-</description>`
	`  <exHosts>test</exHosts>`
	`  <execCwd>/home/lsfadmin/generic_1342080220286ST0Yx</execCwd>`
	`...`
	`  <Job>`
	`  ...`
	`  </Job>`
	`</Jobs>`
		Returns multiple jobs in XML format.

Failure Message	When no data is found, the following is returned:
	`<Jobs>`
	`  <note>%s</note>`
	`</Jobs>`
	
	When there is an error message, the returned XML is:
	`<Jobs>`
	`  <errMsg>%s</errMsg>`
	`</Jobs>`
		%s=return note message or error message

Note Message	`<Jobs>`
	`  <note>%s</note>`
	`</Jobs>`
		%s=return note message

The following is the information returned when the parameter `details=yes`. All returned information is of type String.

Job information	Description
Application Profile	The application profile the job was submitted to.
Application Type	Name of application template used to submit the job.
Command	The job command specified during job submission.
Current Working Directory	The current working directory on the submission host.
Description	The job description assigned to the job during job submission.
End Time	For done or exited jobs: when the job finished. For running jobs, when the job is estimated to finish. Calculated as Current time + Time Remaining.
Estimated Run Time	Estimated running time for the job specified by the user during job submission, with the LSF command **bsub -We**.
Execution Cwd	Current working directory where job is running.

Job information	Description
Execution Hosts	The name of one or more hosts on which the job is running.
Exit Code	Job exit code.
External Status	External status information for a job, specified with the LSF command `bpost -d`.
Graphic Job	Indicates whether the job was submitted with remote console support enabled.
Input Files	Input files specified during job submission.
Job ID	The job ID that LSF assigned to the job.
Job Name	Name of job specified during job submission.
Job Priority	Priority specified for the job during job submission.
Job Type	Type of workload. Possible values: are job, job array, or flow.
Mem	Total resident memory usage in MB, of all processes in a job. For host-based resource usage, the total resident memory usage of all processes in a job running on a host.
Number of threads	Number of currently active threads of a job.
Output Files	Output files created by the job.
Pending Reason	The reason the job is in the Pending state.
Process Group ID	Currently active process group ID in a job.
Process ID	Currently active processes in a job.
Project	The project the job was submitted from.
Queue	The name of the job queue to which the job was submitted.
Requested Hosts	All the resource requirement strings you specified during job submission.
Required Processors	Number of processors specified as required to run the job during job submission.
Run Time	How long the job has been running.
Start Time	When the job started running.
Status	Current status of job.
Submission Host	Host from which the job was submitted.
Submission Time	When the job was sent to the system.
Swap	Total virtual memory usage in MB, of all processes in a job. For host-based resource usage, the total virtual memory usage of all processes in a job running on a host.
Time Remaining	Time remaining to complete the job. Calculated as Estimated Runtime – Runtime. Available only for jobs submitted with an estimated runtime with the LSF command **bsub -We**.
User	The user who submitted the job.

Examples

Get information about all jobs submitted by the user who is currently logged in

```
/platform/webservice/pacclient/jobs
```

Get short information for job with ID 123

```
/platform/webservice/pacclient/jobs?id=123
```

Get detailed information for job with ID 123

/platform/webservice/pacclient/jobs?id=123&details=yes

Get job information for job array 1 in job 101

/platform/webservice/pacclient/jobs?id=101[1]&details=yes

Get short information for all jobs submitted by user abc

/platform/webservice/pacclient/jobs?user=abc

Get detailed information for all Done jobs submitted by user abc

/platform/webservice/pacclient/jobs?status=done&user=abc&details=yes

Get job information for all Done jobs submitted by the user that is currently logged on that have ended within the past 24 hours

/platform/webservice/pacclient/jobs?status=Done&past=24

Get detailed job information for all jobs submitted by the user that is currently logged on in the job group mygroup

/platform/webservice/pacclient/jobs?group=/mygroup&details=yes

Get detailed job information for all jobs in the job group /mygroup submitted by user abc

/platform/webservice/pacclient/jobs?group=/mygroup&user=abc&details=yes

Get detailed job information for all jobs submitted by user abc in xml format

/platform/webservice/pacclient/jobs?user=abc&details=yes&_type=xml

Get Job(Deprecated)

Purpose

Deprecated since version 9.1.0.0. Use “Get Jobs” on page 123 instead.

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>/platform/webservice/pacclient/jobs/<id>	<id>: integer
Content-Type	application/xml	Also supports application/json
Accept	text/xml	Also supports application/json
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	`<Job>`
	`  <name>%s</name>`
	`  <id>%s</id>`
	`  <status>%s</status>`
	`  <cmd>%s</cmd>`
	`</Job>`	

Failure Message	When no data is found, the following is returned:
	`<Job>`
	`  <note>%s</note>`
	`</Job>`
	
	When there is an error message, the returned XML is:
	`<Job>`
	`  <errMsg>%s</errMsg>`
	`</Job>`
		%s=return note message or error message

Get Jobs for Status(Deprecated)

Purpose

Deprecated since version 9.1.0.0. Use "Get Jobs" on page 123 instead.

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>/platform/webservice/pacclient/jobsforstatus/<status>	<status>: one of: Running, Pending, Exit, Done, Suspended
Content-Type	application/xml	Also supports application/json
Accept	text/xml	Also supports application/json
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>`	

HTTP Response	Field format/value	Notes

Failure Message	When no data is found, the following is returned:
	`<Jobs>`
	`  <note>%s</note>`
	`</Jobs>`
	
	When there is an error message, the returned XML is:
	`<Jobs>`
	`  <errMsg>%s</errMsg>`
	`</Jobs>`
		%s=return note message or error message

Get Jobs for Name(Deprecated)

Purpose

Deprecated since version 9.1.0.0. Use “Get Jobs” on page 123 instead.

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>/platform/webservice/pacclient/jobsforname/<jobName>	<jobName>: alphanumeric string. Support "*" to match multiple jobs.
Content-Type	application/xml	Also supports application/json
Accept	text/xml	Also supports application/json
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>`	

HTTP Response	Field format/value	Notes

Failure Message	When no data is found, the following is returned:
	`<Jobs>`
	`  <note>%s</note>`
	`</Jobs>`
	
	When there is an error message, the returned XML is:
	`<Jobs>`
	`  <errMsg>%s</errMsg>`
	`</Jobs>`
		%s=return note message or error message

Job Operations

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	http://<hostname>:<port>/platform/webservice/pacclient/jobOperation/<action>/<ID>	<action>: one of the following: • kill • suspend • requeue • resume
Content-Type	application/xml	Also supports application/json
Accept	text/xml	Also supports application/json
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>` `  <actionMsg>%s</actionMsg></Job>`	%s=returned action output message

Failure Message	`<Job>`
	`  <errMsg>%s</errMsg>`
	`</Job>`
		%s=return error message

Custom Actions

Purpose

The custom actions service allows you to run any command. You can extend the job control actions that are currently supported and also run any other commands that are not related to jobs such as for example, **ls** or **hostname**.

You list commands in the configuration file `$GUI_CONFDIR/webservice.usercmd`.

By default, the following LSF commands are listed in this file:

- **bbot**
- **bchkpnt**
- **bkill**
- **bpost**
- **brequeue**
- **brestart**
- **bresume**
- **brun**
- **bstop**
- **btop**
- **bswitch**
- **bmig**

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	POST	
URL	http://<hostname>:<port>/platform/webservice/pacclient/userCmd	userCmd: Uses the configuration file `$GUI_CONFDIR/webservice.usercmd` to execute any command listed.
Content-Type	application/xml	Also supports application/json
Accept	text/xml	Also supports application/json
Cookie	platform_token=<token>	<token>: returned from logon
Body	`<userCmd>%s</userCmd>`	%s = string started with supported commands configured in `$GUI_CONFDIR/webservice.usercmd`.

Response

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

Success Message	`<Job>`
	`  <cmdMsg>%s</cmdMsg>`
	`</Job>`
		%s=returned command output message

Failure Message	`<Job>`
	`  <errMsg>%s</errMsg>`
	`</Job>`
		%s=returned 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>/platform/webservice/pacclient/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.

Upload Files

Purpose

Uploads data for a job to the job directory on the web server or the remote directory on a specific host. By default, if a job ID is specified but no directory is specified, files are uploaded to the job directory on the web server.

The job can be in any status.

Request

HTTP Request Field	Field format/value	Notes
HTTP Method	POST	

HTTP Request Field	Field format/value	Notes
URL	http://<hostname>:<port>/platform/webservice/pacclient/upfile/<job_ID>	• <dir_name>: Optional. Uploads files to the specified directory on the web server. The directory can be an absolute path on the web server or a relative path to the job directory on the web server. If not specified, the job directory on the web server is used. • <job_ID>: Only used to upload files to the job directory on the web server. The ID of the job for which to upload files. The ID must be a positive integer and must match an LSF job ID. • <file_name>: Files to upload. To specify multiple files, separate with a space. If no path is specified for the file, the job directory on the web server is used as the location of the file. If no file is specified, all files in the directory are uploaded. • <host_name><dir_name>: Uploads files to the specified job directory on the specified host. The directory must be an absolute path to the remote job directory on the host. The host can be any LSF compute host.
Content-Type	multipart/mixed;boundary=XXX	
Accept	text/plain	
Cookie	platform_token=<token>	<token>: returned from logon
Body	[--boundary <Directory Name>] --boundary <File Attachment> --boundary <File Attachment> --boundary ...	Boundary can be any random unique string. For details, see additional tables.

<Directory Name>

Field Name	Field format/value	Notes
Content-Disposition	Form-data;name="DirName"	
Content-ID	DirName	
Content	<Dir_Name>	<Dir_Name>: The directory on the web server. If the whole area is missing, the job directory on the web server.

```
Example
-----------------------------------
Content-Disposition: form-data; name=DirName
Content-ID: <DirName>
"\r\n"
/shareDisk/jobRepo/lsfadmin/workspace_dept_00001B
-----------------------------------
```

<File Attachment>

Field Name	Field format/value	Notes
Content-Disposition	Form-data;name="fl"	
Content-Type	application/octet-stream	

Field Name	Field format/value	Notes
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: <examplefile.jou>
"\r\n"
PGh0bWw+CiAgPGhlYWQ+CiAgPC9oZWFkPgogIDxib2R5PgogICAgPHA+VGhpcyBpcyB0aGUg
-----------------------------------
```

Response

HTTP Response	Field format/value	Notes
Response code	"200" - connection ok; otherwise connection failed	
Success Message	Files successfully uploaded.	
Failure Message	Failed to upload files.	Error details are in the log file.

Example: upload file DATA_SAMPLE_1.txt to the job directory on the web server for job 1508

URL: `http://pac91:8080/platform/webservice/pacclient/upfile/1508`

```
HTML HEADER:

Cookie:platform_token=user1#quote#2013-01-18T04:51:39Z#quote#J5exDToJhwqFMDgRr+ngVBXPyma37zproA9VdEKNeYBUGbeQ1f/
oIoaQCTRC+LDR7L5/p0JFldlhi/RJ9A/wr+tPAGK0G3pORtYQi0blq5DD2g78MBJeFb7VeAiXSIQN#quote#ZUQCAyMsUlJF5i87vk9gBg==
Content-Length:492
Content-Type:multipart/mixed; boundary=4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Accept:text/plain;

HTML BODY:

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DirName"
Content-ID: <DirName>

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DATA_SAMPLE_1.txt"; filename="DATA_SAMPLE_1.txt"
Content-Type: application/octet-stream
Content-Transfer-Encoding: UTF-8
Content-ID: <DATA_SAMPLE_1.txt>

SAMPLE FILE 1
THIS IS THE DATA SAMPLE FOR TESTING UPLOAD
IBM PLATFORM COMPUTING
Jan 31, 2013

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m-
```

Example: upload file DATA_SAMPLE_1.txt and DATA_SAMPLE_2.txt to the job directory on the web server for job 1508

URL: `http://pac91:8080/platform/webservice/pacclient/upfile/1508`

```
HTML HEADER:

Cookie:platform_token=user1#quote#2013-01-18T04:51:39Z#quote#J5exDToJhwqFMDgRr+ngVBXPyma37zproA9VdEKNeYBUGbeQ1f/
oIoaQCTRC+LDR7L5/p0JFldlhi/RJ9A/wr+tPAGK0G3pORtYQi0blq5DD2g78MBJeFb7VeAiXSIQN#quote#ZUQCAyMsUlJF5i87vk9gBg==
Content-Length:825
Content-Type:multipart/mixed; boundary=4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Accept:text/plain;

HTML BODY:

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DirName"
Content-ID: <DirName>

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DATA_SAMPLE_1.txt"; filename="DATA_SAMPLE_1.txt"
Content-Type: application/octet-stream
Content-Transfer-Encoding: UTF-8
Content-ID: <DATA_SAMPLE_1.txt>

SAMPLE FILE 1
THIS IS THE DATA SAMPLE FOR TESTING UPLOAD
IBM PLATFORM COMPUTING
Jan 31, 2013

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m-
Content-Disposition: form-data; name="DATA_SAMPLE_2.txt"; filename="DATA_SAMPLE_2.txt"
Content-Type: application/octet-stream
Content-Transfer-Encoding: UTF-8
Content-ID: <DATA_SAMPLE_2.txt>

SAMPLE FILE 2
THIS IS THE DATA SAMPLE FOR TESTING UPLOAD
IBM PLATFORM COMPUTING
Jan 31, 2013

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m--
```

Example: upload file DATA_SAMPLE_1.txt to the /tmp directory on host ib05b01

URL: `http://pac91:8080/platform/webservice/pacclient/upfile/0`

```
HTML HEADER:

Cookie:platform_token=user1#quote#2013-01-18T04:51:39Z#quote#J5exDToJhwqFMDgRr+ngVBXPyma37zproA9VdEKNeYBUGbeQ1f/
oIoaQCTRC+LDR7L5/p0JFldlhi/RJ9A/wr+tPAGK0G3pORtYQi0blq5DD2g78MBJeFb7VeAiXSIQN#quote#ZUQCAyMsUlJF5i87vk9gBg==
Content-Length:504
Content-Type:multipart/mixed; boundary=4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Accept:text/plain;

HTML BODY:

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DirName"
Content-ID: <DirName>

ib05b01:/tmp
--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DATA_SAMPLE_1.txt"; filename="DATA_SAMPLE_1.txt"
Content-Type: application/octet-stream
Content-Transfer-Encoding: UTF-8
Content-ID: <DATA_SAMPLE_1.txt>

SAMPLE FILE 1
THIS IS THE DATA SAMPLE FOR TESTING UPLOAD
```

```
IBM PLATFORM COMPUTING
Jan 31, 2013

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m-
```

Example: upload file DATA_SAMPLE_1.txt and DATA_SAMPLE_2.txt to the /tmp directory on host ib05b01

URL: `hhttp://pac91:8080/platform/webservice/pacclient/upfile/0`

```
HTML HEADER:

Cookie:platform_token=user1#quote#2013-01-18T04:51:39Z#quote#J5exDToJhwqFMDgRr+ngVBXPyma37zproA9VdEKNeYBUGbeQ1f/
oIoaQCTRC+LDR7L5/p0JFldlhi/RJ9A/wr+tPAGK0G3pORtYQi0blq5DD2g78MBJeFb7VeAiXSIQN#quote#ZUQCAyMsUlJF5i87vk9gBg==
Content-Length:837
Content-Type:multipart/mixed; boundary=4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Accept:text/plain;

HTML BODY:

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DirName"
Content-ID: <DirName>

ib05b01:/tmp
--4k89ogja023oh1-gkdfk903jf9wngmujfs95m
Content-Disposition: form-data; name="DATA_SAMPLE_1.txt"; filename="DATA_SAMPLE_1.txt"
Content-Type: application/octet-stream
Content-Transfer-Encoding: UTF-8
Content-ID: <DATA_SAMPLE_1.txt>

SAMPLE FILE 1
THIS IS THE DATA SAMPLE FOR TESTING UPLOAD
IBM PLATFORM COMPUTING
Jan 31, 2013

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m-
Content-Disposition: form-data; name="DATA_SAMPLE_2.txt"; filename="DATA_SAMPLE_2.txt"
Content-Type: application/octet-stream
Content-Transfer-Encoding: UTF-8
Content-ID: <DATA_SAMPLE_2.txt>

SAMPLE FILE 2
THIS IS THE DATA SAMPLE FOR TESTING UPLOAD
IBM PLATFORM COMPUTING
Jan 31, 2013

--4k89ogja023oh1-gkdfk903jf9wngmujfs95m--
```

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>/platform/webservice/pacclient/file/<ID>	<ID>: a positive integer
Content-Type	text/plain	

HTTP Request Field	Field format/value	Notes
Accept	text/plain	
Cookie	platform_token=<token>	<token>: returned from logon
Body	/shareDisk/jobRepo/lsfadmin/myJob_22111/a.tar.z ORa.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>/platform/webservice/pacclient/filePost/<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 ORa.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
- httplib2
- 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 security token in the following location: `$HOME/.pacpass`.

logout also removes the `.pacpass` file.

Input

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

Output

On success

- Release the security token 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 = getJobListInfo (parameter)

Description

Returns short or detailed information for jobs and job arrays with the specified job ID, job name, job status. It also returns jobs and job arrays that belong to a specified job group, were submitted by a specified user, or that have finished within a specified time period. By default, when no parameters are specified, returns short information for all jobs and job arrays submitted by the user who is currently logged on.

The information returned is similar to the information that can be retrieved with the LSF commands **bjobs** and **bhist**.

Input

Parameters can be combined together with &.

id Job ID or job array ID. Integer.

status
Job status. String. Possible values:

- Pending
- Suspended
- Running
- Exit
- Done

name
Name of job. String.

group
Name of job group to which the job belongs. String.

user
Name of user for which to display job information. The logged in user must have permission to view the jobs. Specify the keyword `all` to retrieve information for all jobs of all users.

details
Return detailed information about a job. Valid values:

- yes
- no

past
Number of hours. Integer. Can only be used with the job status Done or Exit. Retrieves information for jobs that have finished within the specified number of hours.

Output

On success

Short format:

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

Detailed format:

```
(status="ok")
message = <Jobs>
  <Job>
     <name>%s</name>
     <id>%s</id>
     <status>%s</status>
     <cmd>%s</cmd>
     <cwd>/home/lsfadmin/generic_1342080220286ST0Yx</cwd>
     <description>-</description>
     <exHosts>test</exHosts>
     <execCwd>/home/lsfadmin/generic_1342080220286ST0Yx</execCwd>
...
  <Job>
   ...
  </Job>
</Jobs>
```

The following is the information returned when the parameter `details=yes`. All returned information is of type String.

Job information	Description
Application Profile	The application profile the job was submitted to.
Application Type	Name of application template used to submit the job.
Command	The job command specified during job submission.
Current Working Directory	The current working directory on the submission host.
Description	The job description assigned to the job during job submission.
End Time	For done or exited jobs: when the job finished. For running jobs, when the job is estimated to finish. Calculated as Current time + Time Remaining.
Estimated Run Time	Estimated running time for the job specified by the user during job submission, with the LSF command **bsub -We**.
Execution Cwd	Current working directory where job is running.
Execution Hosts	The name of one or more hosts on which the job is running.
Exit Code	Job exit code.
External Status	External status information for a job, specified with the LSF command `bpost -d`.
Graphic Job	Indicates whether the job was submitted with remote console support enabled.
Input Files	Input files specified during job submission.

Job information	Description
Job ID	The job ID that LSF assigned to the job.
Job Name	Name of job specified during job submission.
Job Priority	Priority specified for the job during job submission.
Job Type	Type of workload. Possible values: are job, job array, or flow.
Mem	Total resident memory usage in MB, of all processes in a job. For host-based resource usage, the total resident memory usage of all processes in a job running on a host.
Number of threads	Number of currently active threads of a job.
Output Files	Output files created by the job.
Pending Reason	The reason the job is in the Pending state.
Process Group ID	Currently active process group ID in a job.
Process ID	Currently active processes in a job.
Project	The project the job was submitted from.
Queue	The name of the job queue to which the job was submitted.
Requested Hosts	All the resource requirement strings you specified during job submission.
Required Processors	Number of processors specified as required to run the job during job submission.
Run Time	How long the job has been running.
Start Time	When the job started running.
Status	Current status of job.
Submission Host	Host from which the job was submitted.
Submission Time	When the job was sent to the system.
Swap	Total virtual memory usage in MB, of all processes in a job. For host-based resource usage, the total virtual memory usage of all processes in a job running on a host.
Time Remaining	Time remaining to complete the job. Calculated as Estimated Runtime – Runtime. Available only for jobs submitted with an estimated runtime with the LSF command **bsub -We**.
User	The user who submitted the job.

On failure

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

Example

```
status, message = getJobListInfo('status='+jobStatus+'&user='+user+'&details='+details+'&past='+pa
```

status,message = getJobInfo (jobId)(Deprecated)

Description

Deprecated since version 9.1.0.0. Use “status,message = getJobListInfo (parameter)” on page 141 instead.

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 **External Status** column. You will need to select the column for display with the **Options** button.

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)(Deprecated)

Description

Deprecated since version 9.1.0.0. Use “status,message = getJobListInfo (parameter)” on page 141 instead.

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 **External Status** column. You will need to select the column for display with the **Options** button.

Input

`jobStatus`

One of the following:

- Running
- Pending
- Suspended
- Exit
- 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)(Deprecated)

Description

Deprecated since version 9.1.0.0. Use “status,message = getJobListInfo (parameter)” on page 141 instead.

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 **External Status** column. You will need to select the column for display with the **Options** button.

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

status,message = doUserCmd(userCmd)

Description

Send the command to the Platform Application Web Services for execution. Used to extend actions you can take on jobs or to execute custom commands.

Input

userCmd

A command listed in the configuration file `$GUI_CONFDIR/webservice.usercmd`.

By default, the following LSF commands are listed in this file:

- **bbot**
- **bchkpnt**
- **bkill**
- **bpost**
- **brequeue**
- **brestart**
- **bresume**
- **brun**
- **bstop**
- **btop**
- **bswitch**
- **bmig**

You can also add any other commands to the file that you want to run such as for example, **ls** or **hostname**.

Output

On success

(status="ok") command 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

uploadJobFiles(jobId, dstPath, dFile)

Description

Uploads data for a job to the job directory on the web server or the remote directory on a specific host. By default, if a job ID is specified but no directory is specified, files are uploaded to the job directory on the web server.

The job can be in any status.

Input

jobId

Only used to upload files to the job directory on the web server.

The ID of the job for which to upload files.

The ID must be a positive integer and must match an LSF job ID.

dstPath

Format: `[host_name:]dstPath`

Optional. Uploads files to the specified directory on the web server. The directory can be an absolute path on the web server or a relative path to the job directory on the web server. If not specified, the job directory on the web server is used.

When a host name is specified, uploads files to the specified job directory on the specified host. The directory must be an absolute path to the remote job directory on the host. The host can be any LSF compute host.

dFile

Files to upload. To specify multiple files, separate with a comma(,). If no path is specified for the file, the job directory on the web server is used as the location of the file. If no file is specified, all files in the directory are uploaded.

If it is an empty string, uploads all files.

Output

On success

Uploads the job files and copies them to <dstPath>.

On failure

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

Chapter 10. 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 Plaform 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.

Try out the example pages from this URL:

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

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 demomstrates basuc job submission and LSF job submission ouput 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 (serviceId= 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?serviceId=JOBGROUP_CUSTOM_Sample_Service`

AJAX call and parameter passing (serviceId=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 serviceId=ajaxcall.

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

Cache data and results with AJAX (serviceId=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 host name 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 serviceId=ajaxspooler.

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

Call a service and pass parameters (serviceId=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 (serviceId=receive).

The redirected page receives the parameters from the service callservice 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='$JOBFLOW_DETAIL_URL?insId=${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 <pac:service> defined in `services.xml`. For example: `<pac:service id="menu">`	True

Access a custom page from an external portal

In some cases, you may want to access a custom Platform Application Center page that you created from an external portal, and do not want the Platform Application Center navigation tree to be displayed.

You can access any custom page by adding the parameter `&embed=yes` at the end of the URL.

For example, to call the example custom page `submitjob` from an external portal:

```
http://pac_host_name:8080/platform/service/pageFactoryAction.action?serviceId=submitjob&embed=yes
```

The Platform Application Center logon screen is displayed only the first time a user accesses the URL. Logging in will not be required for accessing subsequent custom pages in Application Center.

The Platform Application Center logon screen is also displayed when the session has timed out.

Note:

According to the user's browser settings, it is possible that when the user attempts to view the URL, the action will be blocked. Users will need to select **Allow blocked content**. This will only be required once per user.

In addition, for Internet Explorer, you will need to do the following so Internet Explorer does not block the action:

1. In Internet Explorer, select**Tools** > **Internet Options**.
2. Select the **Advanced** tab.
3. Scroll down to the **Security** section and make sure the following option has a checkmark in it: **Allow active content to run in files on My Computer**.
4. Click **OK** and restart Internet Explorer.

   The change will take effect the next time you open Internet Explorer.

Page customization XML reference

<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

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

Sub-tags

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	Specifies the ID of the action to execute. The action must have been previously defined.	True

Triggers

<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 I
mage</button>
      <pac:action id="put_my_gif_in_div" hidden="true">
           <![CDATA[
           echo "\$('msgarea').innerHTML='<img src=\"$PAC_DOWNLOAD_URL?spooler=$PAC_SPOO
LER_URI&amp;file=corp_logo.jpg\" 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>
<Configuration>
 <Repository>
 <User>all</User>
<Path>/home</Path>
</Repository>
 </Configuration>
</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

Examples:

- Create a dynamic list option:

```
<pac:option id="QUEUE" label="@{selectTag}" type="list">
   <pac:action 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:action>
</pac:option>
```

- Forward to a Java action:

```
<pac:action id="submit" name="LicenseOverviewAction" label="submit">
   <pac:result type="text/html"/>
</pac:action>
```

<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/schemaBinds 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: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=\"${FLOWID}\"
             redirect='$JOBFLOW_DETAIL_URL?insId=${FLOWID}'/>"
           ]]>
      </pac:action>
</pac:service>
```

<pac:flow-submit>

This tag has been deprecated since version 8.0.2. Use `<pac:flow-details>` instead.

<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-spooler> has the following attributes:

Name	Description	Required
id	Specifies the job directory page to redirect to	True

Example:

```
<pac:spooler 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: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: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: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 $JOB_DETAIL_URL?insld=${JOBID} for the URL.	True
jobName	Descriptive name for the job.	Optional for jobs. Required for job arrays.
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?jobld=${JOBID}' bsubH="false"  jobName=\"${JOBNAME}\"/>"
           ]]>
      </pac:action>
</pac:service>
```

<pac:job-submit>

This tag has been deprecated since version 8.0.2. Use `<pac:job-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 following variable for the URL: $JOBGROUP_DETAIL_URL?groupName=${groupName}	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 JOBID
              if [ "${JOBID}"="" ]; then
                          JOBID="$ID"
              else
                          JOBID="$JOBID,$ID"
              fi
    done

 echo "<pac:jobgroup-details id=\"${JOBID}\" groupName=\"${GROUPNAME}\" redirect='$JOBGROUP_DETAIL_URL
?groupName=${GROUPNAME}' bsubH='false'/>"
]]>
    </pac:action>
</pac:service>
```

<pac:jobgroup-submit>

This tag has been deprecated since version 8.0.2. Use `<pac:jobgroup-details>` instead.

<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: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: lablel,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
value	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
html_event	Specifies an HTML event to trigger when the control is used.Supported events: • 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
<pac:option>	Specifies an optional tag if the parent <pac:option> requires a list of choices, like <pac:list>, <pac:radio> and <pac:checkbox>	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

A label option

```
<pac:option id="lable1" 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 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[
 $GUI_CONFDIR/application/options/queue.sh normal
          ]]>
    </pac:action>
</pac:option>
```

<pac:reset-spooler>

This tag resets the time-to-live of the spooler identified by the uri attribute.

<pac:reset-spooler> 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: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:result> 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: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

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

<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:spooler-info> 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:spooler-info>`, 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

Built-in JavaScript functions

The following JavaScript functions are built-in functions. You can use them directly in services.xml.

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().**

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 jsonStr="{'name':'" + n + "', 'age':'" + a + "'}";
             var jsonObj = pac_toJSONObject(jsonStr);
             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>
```

CDATA section

CDATA is text data that should not be parsed by the XML parser. For example, caracters like "<" and "&" are not valid in XML elements. CDATA can include JavaScript or a Linux shell script is a 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 $INP_INPUT_FILE ];then
           echo "You must upload a local file."
        else
           echo "You uploaded this file:<br/>"
           echo "$PAC_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.jpg\" alt=\"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>
```

Chapter 11. Reference

perfadmin

Administer the reporting 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 reporting services, or shows status.

Run the command on the PERF host to control the following reporting services: loader controller (plc), job data transformer (jobdt), and data purger (purger).

If reporting services are controlled by EGO, let the EGO service controller start and stop the reporting 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).

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).

pmcadmin

Administer Platform Application Center.

Synopsis

pmcadmin start | stop | list | https enable | https disable | -h | -V

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.

-V

Displays product version.

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
- **pmcremoverc** command: disable automatic startup of Platform Application Center on a host

pmcremoverc.sh

Prevents automatic startup of the Platform Application Center on a UNIX host.

Synopsis

pmcremoverc.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 **`pmcsetrc.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.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Intellectual Property Law
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application

programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

LSF, Platform, and Platform Computing are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Printed in USA

SC22-5396-01