The Process Controller for the LHCb On-Line Farm

Public Note

Issue: 1
Revision: 0

Reference: LHCb-2007-095
Created: May 30, 2007
Last modified: July 3, 2007

Prepared by: Federico Bonifazi[a], Daniela Bortolotti[b], Angelo Carbone[a], Domenico Galli[c], Daniele Gregori[a], Umberto Marconi[b], Gianluca Peco[b], Vincenzo M. Vagnoni[b]

[a]INFN-CNAF, Bologna, I-40127, Italy.
[b]INFN Sezione di Bologna, I-40126, Italy.
[c]Alma Mater Studiorum – Università di Bologna and INFN Sezione di Bologna, Bologna, I-40126, Italy.

Abstract

The *Process Controller* is a tool of the LHCb **FMC** (Farm Monitoring and Control System) in charge of **keeping a list of applications up and running on the farm nodes**. It tipically runs on a few *control PCs* each one watching ∼200 farm nodes and performs its task by maintaining the list of scheduled applications for each controlled farm node and by interacting with the Task Manager Servers [1] running on the farm nodes to start processes, to obtain the notification of process termination, to re-spawn the terminated processes (if requested) and to stop processes. Processes can be added to or removed from the scheduled application list for one or more nodes by means of DIM [2] commands, while DIM services provide the list of scheduled applications for each controlled farm node together with their properties, the number of re-spawns and the re-spawn times.

Document Status Sheet

1. Document Title: The Process Controller for the LHCb On-Line Farm			
2. Document Reference Number: LHCb-2007-095			
3. Issue	**4. Revision**	**5. Date**	**6. Reason for change**
Draft	0	May 30, 2007	First version. Introduction only, conclusion is missing
Draft	1	May 30, 2007	Added conclusion.
1	0	July 3, 2007	Refers to FMC version 3.6.10.

Contents

List of Figures

List of Tables

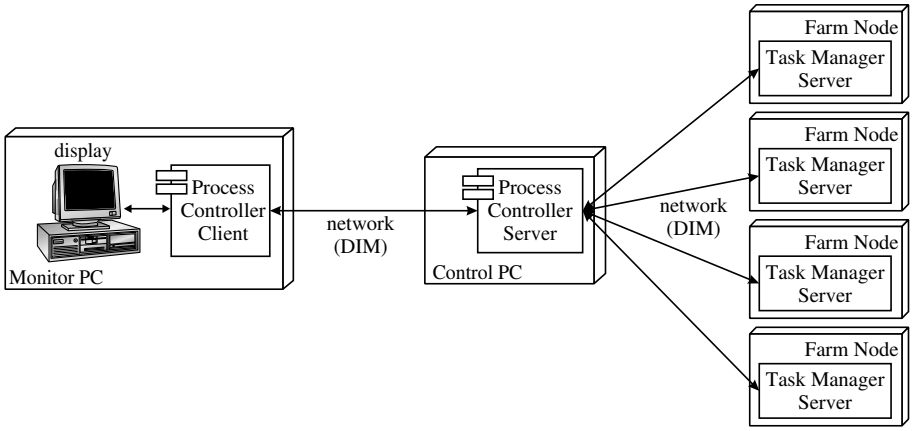

Figure 1 Diagram of the Process Controller deployment.

1 Requirements

In the normal operation of the LHCb Event Filter Farm, a certain number of processes must be **kept running** on each node (trigger processes, monitor processes, etc.) in order for the node to be operational. If one of these processes terminates unintentionally it must be restarted **as soon as possible** in order to minimize the node down-time.

The standard Linux process control (`init` [3]) provides a mechanism to keep a list of process up and running, accessible by means of the `respawn` option of the `inittab` [4] [3]. However, to add or remove a process from the list of the scheduled processes, the `/etc/inittab` file has to be edited and the command "`/sbin/telinit q`" has to be issued on the node itself.

The LHCb Event Filter Farm needs a more flexible mechanism, which allows to **add or remove** processes **on-the-fly** from the **list of the scheduled processes**, for **one or more nodes**, and to record the **number of restarts** and the **restart times**.

2 The Process Controller Implementation

The FMC Process Controller is a tool to keep a given list of processes up and running on each farm node, allowing to on-line *add* and *remove* processes from the list for one or more nodes by means of commands issued from a central console.

As shown in Fig. 1, the complete Process Controller System involves:

- a **Process Controller Client** running on the Monitor PC;
- a few **Process Controller Servers** running on the Control PCs (each one watching $\sim$200 farm nodes);
- the **Task Manager Servers** running on each farm node.

The communication between the Process Controller Client and the Process Controller Servers as well as that between the Process Controller Servers and the Task Manager Servers is performed by means of the DIM [2] network communication layer.

The **Process Controller Server** maintains a **list of scheduled processes** for each controlled node and keeps the processes up and running by interacting with the **Task Manager Server** (to be notified about unintentionally terminated processes and to re-start them).

A process can be **added** or **removed** from the list for one or more nodes by means of commands sent to the Process Controller Server by the **Process Controller Client**.

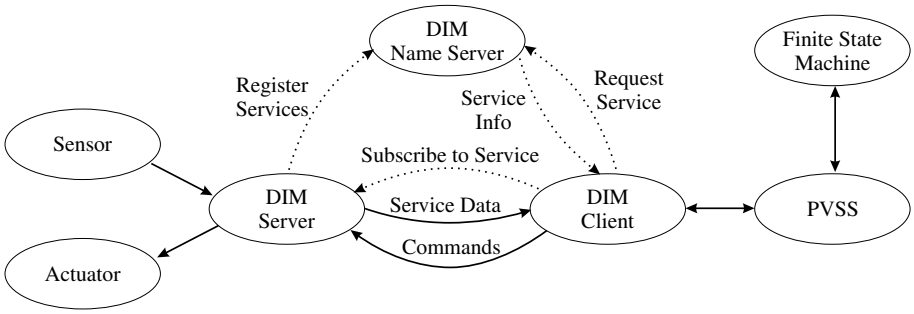

Figure 2 Diagram of the DIM network communication mechanism.

If a new process is **added** to the list of the i-th node, the process is also immediately *started* on the i-th node (then it is *re-started* whenever it unintentionally terminates on the i-th node). If a process is **removed** from the list of the i-th node it is also *terminated* on the i-th node.

The **Process Controller Client** can also get by the Process Controller Server the *list the scheduled processes* for a given node together with the *process start parameters*, the *number of restarts* and the *time-stamp of each restart*.

2.1 The DIM Network Communication Layer

DIM (Distributed Information Management System) [2] is a network communication layer built on top of the TCP protocol and based on the client/server paradigm, making use of a name server (Fig. 2).

The *server* provides *data services* and *command services* to clients by "registering" them with the *name server* (normally once, at startup). The *client* "subscribes" to services by asking the name server which server provides the service and then contacting the server directly.

The update of the data on the client side can happen ONCE_ONLY (the client will receive the data once and then will disconnect from the server), TIMED (the client will receive the data at regular time intervals) or MONITORED (the client will receive the data only when the server updates them).

The client can also send a command to a command service to require the execution of a procedure on the server side.

2.2 The Interaction Among the Components

The **Process Controller Server** acts both as a DIM client and as a DIM server.

- As a **DIM client**, it subscribes on each controlled farm node to the Task Manager **`list`** service (to be notified about the processes running on the node) and contacts the Task Manager **`start`** command (to start a new process or re-start an unintentionally terminated process) and **`stop`** command (to terminate a process).
- As a **DIM server**, it publishes the commands **`add`** (to add a process to the list of scheduled processes for one or more nodes and to start it on those nodes) and **`rm`** (to remove a process from the list of scheduled processes for one or more nodes and to stop it on those nodes) and the services **`ls`** (to list the scheduled processes), **`ll`** (to list the scheduled processes and their start parameters, i.e. path, UTGID, environment variables, scheduler, priority, re-spawn parameters, etc.) and **`lss`** (to list the scheduled processes and, for each process, the number of restarts and the time of each restart).

The services **`ls`**, **`ll`** and **`lss`** and the commands **`add`** and **`rm`**, published by the Process Controller Server, are then contacted by the Process Controller Client to issue commands and to get the process status information.

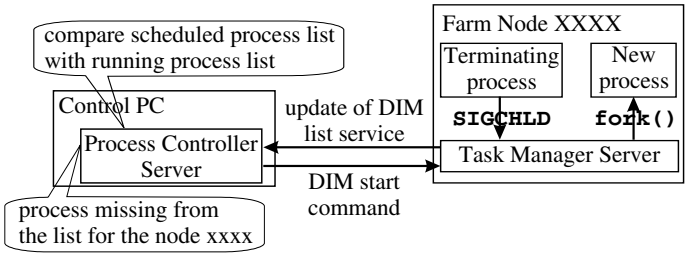

Figure 3 Diagram of the Process Controller Fast Re-spawn Mechanism.

2.3 The Fast Process Re-spawn

The Process Controller Server, in collaboration with the Task Managers provides a mechanism for the fast re-spawn of unintentionally terminated processes (Fig. 3). The mechanism works as follow:

- If a process on a node terminates, the OS kernel sends a CHLD signal to the Task Manager Server (parent of the terminating process), which is waiting for signals (`sigtimedwait(2)` [5]).
- Upon the reception of the CHLD signal, the Task Manager Server immediately stops waiting and updates the *list* service.
- The Process Controller Server—which has subscribed to the *list* service— upon receiving the *list* service update, compares the list of running processes with the list of scheduled processes and finds a missing process.
- Thus the Process Controller Server sends the *start* command to the Task Manager Server which, in turn, restarts the process.

2.4 The Re-spawn Control

The Process Controller Server is provided by a mechanism to **avoid the continuous re-spawn** of a process which terminates immediately, due to some circumstances to be corrected.

The mechanism conforms to the following rules:

- if a process is restarted more than $N_{\max}$ times in T_C seconds, then the process re-spawn is disabled for T_D seconds and then enabled again;
- default values (different values can be set from the Process Controller *add* command) are: $N_{\max} = 10$; $T_C = 120$ s; $T_D = 300$ s;
- if $N_{\max}$ is set to -1, then the re-spawn control is excluded, i.e. the process can be restarted indefinitely;
- if T_D is set to -1, then the process restart, once disabled, is never re-enabled.

2.5 Process Controller Configuration

The Process Controller Server configuration, i.e. the list of the nodes to be controlled, can be loaded in 2 different ways:

- through a file on a file system mounted on the machine running `pcSrv`, by specifying the `-c conf_file` command line option;
- through the DIM service `/⟨CTRL_PC_NAME⟩/config_manager/pc/get` provided by the FMC Configuration Manager `cmSrv` (if the `-c conf_file` option is not specified on the `pcSrv` command line).

In the former case the file `conf_file` has to be a UNIX ASCII file (lines terminated by the only '\n' character) containing one hostname for each line. Blank lines as well as lines starting with the '#' characters (i.e. comment lines) are skipped.

In the latter case, if the Process Controller is started before the Configuration Manager, it waits for the Configuration Manager to be running and configured.

3 The Process Controller Server

The Process Controller Server, whose executable image name is **`pcSrv`**, runs on the Control Nodes of the farm. Each server typically controls ∼200 farm nodes. The list of the nodes controlled by the current control node should be available either through the FMC Configuration Manager or through the file **`conf_file`**.

3.1 Synopsis

`pcSrv[-l logger_unit][-c conf_file][-s init_file]`

`pcSrv[-h]`

3.2 Description

Starts the Process Controller Server **`pcSrv`** on the current control node and sends its diagnostic messages to a logger unit.

3.3 Command Line Options

`-h` — Print a help message and exit.

`-l logger_unit` — Send diagnostic messages to the logger units defined in the logger mask, which is the bitwise OR of the following values:

`0x1` — **`L_DIM`**, the default FMC Event Logger (**`/tmp/logSrv.fifo`**);

`0x2` — **`L_STD`**, the standard error stream;

`0x4` — **`L_SYS`**, the Linux **`syslog`** facility.

`-c conf_file` — Read the Process Controller configuration (i.e. the list of the farm nodes controlled by the current Process Controller, running on the current node) from the file **`conf_file`**. If this option is not specified, the configuration is read through the DIM Configuration Manager **`cmSrv`**, which must be running on this host.

`-s init_file` — Read from the file **`init_file`** a list of commands to add scheduled processes to the list, as soon as the Process Controller is ready. If this option is not specified no processes are started at **`pcSrv`** start-up. Anyhow processes can be added later by using the **`add`** DIM CMD.

3.3.1 `conf_file` *sample*

```
# conf_file sample
farm0101
farm0102
# farm0103
farm0104
```

3.3.2 `init_file` *sample*

```
# interrupt monitor
-m * -u irqSrv_u -D DIM_DNS_NODE=ctrl01.daq.lhcb
  -D LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib /opt/FMC/sbin/irqSrv -l 1
# network interface monitor
-m * -u nifSrv_u -n root -D DIM_DNS_NODE=ctrl01.daq.lhcb
  -D LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib /opt/FMC/sbin/nifSrv
  -l 1 -u 10
```

3.4 Environment

`deBug` — The debug level (default: 0). Can be set to 0…3. Higher debug level corresponds to more verbose output to the logger.

`DIM_DNS_NODE` — **`hostname.domain`** of the DIM dns node.

`LD_LIBRARY_PATH` — Variable, in `PATH` format, which must contain the path to the shared libraries **`libdim.so`** and **`libFMCutils.so`**.

3.5 Published DIM Commands and Services

3.5.1 CMD: **`/⟨CTRL_PC_NAME⟩/process_controller/add`**

- *Command String Synopsis*:

  `[-m node_pattern...] [-c] [-D NAME=value...] [-s Scheduler]`
  `[-p nice_level] [-r rt_priority] [-a cpu_num...] [-d] [-n user_name]`
  `[-g group_name] [-w wd] [-o] [-e] [-O outFIFO] [-E errFIFO] [-A]`
  `[-N maxStartN] [-K checkPeriod] [-X disPeriod] -u utgid path [arg...]`

- *Description*:
  Add a new process to the list for the controlled nodes whose hostname matches at least one of the wildcard **`node_pattern`**, using the executable file located in **`path`** and the arguments specified in **`arg`** and **start the process**. In the environment of the started process the string variable UTGID (User assigned unique Thread Group Identifier) is set to **`utgid`**. By default, before starting the process, all file descriptors are closed and standard file descriptors (`STDIN_FILENO`, `STDOUT_FILENO` and `STDERR_FILENO`) are reopened on `/dev/null`.

- *Command String Options*:

  `-m node_pattern (repeatable)` — Add process to the controlled nodes whose hostname matches the wildcard pattern **`node_pattern`**. *Default*: add process to all the controlled nodes.

  `-c` — Clear the environment inherited by the Task Manager. *Default*: inherit the Task Manager environment.

  `-D NAME=Value (repeatable)` — Set the environment variable **`NAME`** to the value **`Value`**.

  `-s Scheduler` — Set the process scheduler for the process to **`Scheduler`**. Allowed values for **`Scheduler`** are:

  - **`0 = SCHED_OTHER`**, the default Linux time-sharing scheduler, with a dynamic priority based on the nice level;
  - **`1 = SCHED_FIFO`**, the static-priority Linux real-time FIFO scheduler, without time slicing. A `SCHED_FIFO` process runs until either it is blocked by an I/O request, it is preempted by a higher priority process, or it calls `sched_yield(2)`.

- **2 = SCHED_RR**, The static-priority Linux real-time Round-Robin scheduler. It differs from SCHED_FIFO because each process is only allowed to run for a maximum time quantum. If a SCHED_RR process has been running for a time period equal to or longer than the time quantum, it will be put at the end of the list for its priority.

See manual pages for `sched_setscheduler(2)` [6] for more details. *Default*: 0.

-p nice_level — If `Scheduler = 0`, set the nice level of the process to **nice_level**. This value is used by the SCHED_OTHER time-sharing Linux scheduler to compute the dynamic priority. Allowed values for `nice_level` are in the range 20...19 (20 corresponds to the most favorable scheduling; 19 corresponds to the least favorable scheduling). Nice level may be lowered also for processes which run as a user different from `root` (in fact the `nice_level` is set before the `uid`). See manual pages for `setpriority(2)` [7] for details.

-r rt_priority — If `scheduler = 1,2`, set the real time priority to **rt_priority**. Only value 0 for `rt_priority` is allowed for scheduler SCHED_OTHER (the default time-sharing Linux scheduler). For SCHED_FIFO and SCHED_RR real-time schedulers, allowed values are in the range 1...99 (1 is the lowest priority, 99 is the highest priority). See manual pages for `sched_setscheduler(2)` [6] for more details.

-a cpu_num (repeatable) — Add the CPU **cpu_num** to the process-to-CPU affinity mask. More than one of these options can be present in the same command to add more than one CPU to the affinity mask. Started process is allowed to run only on the CPUs specified in the affinity mask. Omitting this option, process is allowed to run on any CPU of the node. Allowed `cpu_num` depend on the PC architecture: e.g., in a single-processor PC with Hyper-Threading activated or in a dual processor PC without Hyper-Threading `cpu_num` can be 0 or 1; in a dual processor PC with Hyper-Threading activated `cpu_num` can be 0, 1, 2 or 3.

-d — Start process as *daemon*: the process umask (user file-creation mask) is changed into 0 and the program is run in a new session as process group leader (`setsid(2)`).

-n user_name — Set the effective `UID`, the real `UID` and the saved `UID` of the process to the `UID` of the user **user_name**. If `tmSrv` was started with `-p 0` command line option only `default_user` can be specified as `user_name`. If `tmSrv` was started with `-p 1` command line option, any existing user name different from `root` can be specified as `user_name`. If `tmSrv` was started with `-p 2` command line option, any existing user name including `root` can be specified as `user_name`.

-g group_name — Set the effective `GID`, the real `GID` and the saved `GID` of the process to the `GID` of the group **group_name**. If `tmSrv` was started with `-p 0` command line option only `default_group` can be specified as `group_name`. If `tmSrv` was started with `-p 1` command line option, any existing group name different from `root` can be specified as `group_name`. If `tmSrv` was started with `-p 2` command line option, any existing group name including `root` can be specified as `group_name`.

-w wd — Set the process working directory to **wd**. File open by the process without path specification are sought by the process in this directory. Process relative file path start from this directory.

-e — Redirect process `stderr` to the default DIM logger. Omitting `-e` or `-E errFIFO` options, standard error is thrown in `/dev/null`.

-o — Redirect process `stdout` to the default DIM logger. Omitting `-o` or `-O outFIFO` options, standard output is thrown in `/dev/null`.

-E errFIFO — Redirect process `stderr` to the DIM logger which uses the FIFO (named pipe) **errFIFO**. Omitting `-e` or `-E errFIFO` options, standard error is thrown in `/dev/null`.

-O outFIFO — Redirect process `stdout` to the DIM logger which uses the FIFO (named pipe) **outFIFO**. Omitting `-o` or `-O outFIFO` options, standard output is thrown in `/dev/null`.

-A — In conjunction with `-e`, `-E`, `-o` and `-O` options uses the no-drop policy for the FIFO. *Default*: uses the congestion-proof policy for the FIFO.

-N maxStartN -K checkPeriod -X disPeriod — Set the respawn control parameters. If a process is (re)started more than **maxStartN** times in **checkPeriod** seconds, then the

process respawn is disabled for **disPeriod** seconds. The *default* respawn parameters are: maxStartN = 10, checkPeriod = 120 s and disPeriod = 300 s. If maxStartN = −1, then respawn control is excluded, i.e. process can be restarted indefinitely. If disPeriod = −1, then the process- restart for the process, once disabled, is never re-enabled.

-u utgid — Set the process UTGID to **utgid**.

3.5.2 *CMD:* /⟨CTRL_PC_NAME⟩/process_controller/rm

- *Command String Synopsis*:

  [-m node_pattern...] [-s Signal] [-d kill9_delay] utgid_pattern

- *Description*:
  Remove the processes whose UTGID matches the wildcard pattern **utgid_pattern** from the list for the controlled nodes whose hostname matches at least one of the wildcard patterns **node_pattern** and **terminates** these processes by sending them the signal **Signal** and — if they block or ignore the signal — by sending them the **KILL** signal after **kill9_delay** seconds.

- *Command String Options*:

  -m node_pattern (repeatable) — Remove processes from the controlled nodes whose hostname matches the wildcard pattern **node_pattern**. *Default*: remove processes from all the controlled nodes.

  -S Signal — Send the signal **Signal**. If not specified, signal 15 (SIGTERM) is sent.

  -d kill9_delay — Use **kill9_delay** as the delay between the first signal (SIGTERM or signal specified with the -s option) and the second signal (SIGKILL). If not specified a one second delay is assumed.

3.5.3 *SVC:* /⟨CTRL_PC_NAME⟩/process_controller/ls

- Description:
  Return an array of strings (separated by the NUL byte) containing, for each controlled node, the number of the scheduled processes and the list of their UTGIDs.

- Format:

  $$\langle \mathtt{hostname}_1 \rangle : \mathtt{n}_1 : \mathtt{utgid}_{1,1}, \ldots, \mathtt{utgid}_{1,\mathtt{n}_1}.\backslash 0 \langle \mathtt{hostname}_2 \rangle : \mathtt{n}_2 : \mathtt{utgid}_{2,1}, \ldots, \mathtt{utgid}_{2,\mathtt{n}_2}.\backslash 0 \ldots$$

3.5.4 *SVC:* /⟨CTRL_PC_NAME⟩/process_controller/ll

- Description:
  Return an array of strings (separated by the NUL byte) containing, for each controlled node, the number of the scheduled processes, the list of their UTGIDs and, for each UTGID, the process start parameters:
  - the working directory wd if different from "/",
  - the program path path,
  - the string clearEnv if the process was started with the -c option,
  - the list of the added environment variables VAR1=value1, VAR2=value2,...,
  - the string daemon if the process was started with the -d option,
  - the scheduler Scheduler,
  - the nice level nice_level,
  - the real time priority rt_priority,
  - the affinity mask affMask,

- the user `user`, if different from the default user,
- the group `group`, if different from the default group,
- the string `rdrStderr` if the standard error was redirected to the default DIM logger,
- the string `rdrStdout` if the standard output was redirected to the default DIM logger,
- if the standard error was redirected to a specific DIM logger, the name `errFIFO` of the specific FIFO used by the logger,
- if the standard output was redirected to a specific DIM logger, the name `outFIFO` of the specific FIFO used by the logger,
- the string `noDrop` if the process was started with the `-A` option,
- re-spawn parameters `maxStartN`, `disPeriod` and `checkPeriod`.

- Format:

$$\langle \texttt{hostname}_1 \rangle : \texttt{n}_1 : \texttt{utgid}_{1,1}(\texttt{wd} = \text{"}\ldots\text{"}, \texttt{path} = \text{"}\ldots\text{"}, \ldots), \ldots, \texttt{utgid}_{1,\texttt{n}_1}(\ldots).\backslash\texttt{0}$$
$$\langle \texttt{hostname}_2 \rangle : \texttt{n}_2 : \texttt{utgid}_{2,1}(\ldots), \ldots, \texttt{utgid}_{2,\texttt{n}_2}(\ldots).\backslash\texttt{0}\ldots$$

3.5.5 SVC: `/⟨CTRL_PC_NAME⟩/process_controller/lss`

- Description:
  Return an array of strings (separated by the `NUL` byte) containing, for each controlled node, the number of the scheduled processes, the list of their UTGIDs and, for each UTGID, the number of the starts and the date and the time of each start:

- Format:

$$\langle \texttt{hostname}_1 \rangle : \texttt{n}_1 : \texttt{utgid}_{1,1}(\texttt{start} = \texttt{m}_{1,1} : \texttt{MMMdd} - \texttt{hhmmss}_{1,1,1}, \ldots, \texttt{MMMdd} - \texttt{hhmmss}_{1,1,\texttt{m}_{1,1}}),$$
$$\ldots, \texttt{utgid}_{1,\texttt{n}_1}(\texttt{start} = \texttt{m}_{1,\texttt{n}_1} : \texttt{MMMdd} - \texttt{hhmmss}_{1,\texttt{n}_1,1}, \ldots, \texttt{MMMdd} - \texttt{hhmmss}_{1,\texttt{n}_1,\texttt{m}_{1,\texttt{n}_1}}).\backslash\texttt{0}$$
$$\langle \texttt{hostname}_2 \rangle : \texttt{n}_2 : \texttt{utgid}_{2,1}(\ldots), \ldots, \texttt{utgid}_{2,\texttt{n}_2}(\ldots).\backslash\texttt{0}\ldots$$

3.5.6 SVC: `/⟨CTRL_PC_NAME⟩/process_controller/server_version`

- Description:
  Returns the RCS Identification string of the Process Controller Server main program source file, containing version and last modification time.

3.5.7 SVC: `/⟨CTRL_PC_NAME⟩/process_controller/success`

- Description:
  This dummy service always returns 1. It is used by PVSS-DIM [8] to check if the `pcSrv` process is running.

4 The Process Controller Command-Line Clients

4.1 The pcAdd Command-Line Client

4.1.1 Synopsis

```
pcAdd [-v] [-C ctrl_pc_pattern...] [-m node_pattern...] [-c]
      [-D NAME=value...] [-s Scheduler] [-p nice_level]
      [-r rt_priority] [-a cpu_num...] [-d] [-n user_name]
      [-g group_name] -u utgid [-w wd] [-o] [-e] [-O outFIFO]
      [-E errFIFO] [-A] [-N maxStartN] [-K checkPeriod]
      [-X disPeriod] path [arg...]
pcAdd [-h]
```

4.1.2 Description

Add a new process to the list for the nodes whose hostname matches at least one of the wildcard **`node_pattern`**, controlled by the control PCs whose hostname matches at least one of the wildcard **`ctrl_pc_pattern`**, using the executable file located in **`path`** and the arguments specified in **`arg`** and **start the process**. In the environment of the started process the string variable UTGID (User assigned unique Thread Group Identifier) is set to **`utgid`**. By default, before starting the process, all file descriptors are closed and standard file descriptors (**`STDIN_FILENO`**, **`STDOUT_FILENO`** and **`STDERR_FILENO`**) are reopened on **`/dev/null`**.

4.1.3 Command-Line Options

`-h` — Print a help message and exit.

`-v` — Print verbose output.

`-C ctrl_pc_pattern (repeatable)` — Contact the Process Controller Servers (**`pcSrv`**) of the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`** (*default*: contact all the control PCs).

`-m node_pattern (repeatable)` — Start process on nodes (controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**) whose hostname matches the wildcard pattern **`node_pattern`** (*default*: start process on all the nodes controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**).

All other option are described in section 3.5.1 at page 8.

4.1.4 Environment

The program **`pcAdd`** needs the two environment variables:

`DIM_DNS_NODE` — **`hostname.domain`** of DIM dns node.

`LD_LIBRARY_PATH` — Variable, in `PATH` format, which must contain the path to the shared libraries **`libdim.so`** and **`libFMCutils.so`**.

4.1.5 Warning

The *wildcards* [9] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

4.1.6 Examples

```
pcAdd -u counter_0 /opt/FMC/tests/counter
pcAdd -u counter_0 -o -e /opt/FMC/tests/counter
pcAdd -C ctrl02 -m farm0141 -u counter_0 /opt/FMC/tests/counter
pcAdd -C "ctrl*" -u counter_0 /opt/FMC/tests/counter
pcAdd -C "ctrl*" -m "farm01*" -u counter_0 /opt/FMC/tests/counter
pcAdd -C "ctrl0*" -C "ctrl1*" -m "farm01*" -m "farm12*"
      -u counter_0 /opt/FMC/tests/counter
pcAdd -C ctrl\* -u counter_0 /opt/FMC/tests/counter
pcAdd -C ctrl\* -m farm01\* -u counter_0 /opt/FMC/tests/counter
pcAdd -C "ctrl0?" -u counter_0 /opt/FMC/tests/counter
pcAdd -C ctrl0\? -u counter_0 /opt/FMC/tests/counter
pcAdd -C "ctrl[3-7]0" -u counter_0 /opt/FMC/tests/counter
pcAdd -C "ctrl[3-7]?" -u counter_0 /opt/FMC/tests/counter
```

```
pcAdd -C ctrl03 -d -u counter_0 -w /opt/FMC/tests ./counter
pcAdd -C ctrl03 -c -u myps -w /bin ps -e -f
pcAdd -d -u counter_0 /opt/FMC/tests/counter
pcAdd -d -s 1 -r 1 -u counter_0 /opt/FMC/tests/counter
pcAdd -d -p -10 -n galli -u counter_0 /opt/FMC/tests/counter
pcAdd -C crtl03 -m farm0301 -c -d
      -D LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
      -D DIM_DNS_NODE=ctrl00.daq.lhcb -s 1 -r 1 -n galli
      -u myCounter -w /opt/FMC/tests -e -o ./counter 123
pcAdd -C crtl03 -m farm0301 -c -d
      -D LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
      -D DIM_DNS_NODE=ctrl00.daq.lhcb -s 1 -r 1 -n galli -g users
      -u myCounter -w /opt/FMC/tests -e -o ./counter 123
pcAdd -C crtl03 -m farm0301 -c -d
      -D LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
      -D DIM_DNS_NODE=ctrl00.daq.lhcb -s 1 -r 1 -n galli
      -u myCounter -w /opt/FMC/tests -E /tmp/counterErr.fifo
      -O /tmp/counterOut.fifo -A ./counter 123
pcAdd -C ctrl02 -m farm0141 -c -d
      -D LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
      -D DIM_DNS_NODE=ctrl00.daq.lhcb -s 1 -r 1 -a 0 -a 2 -n galli
      -u myCounter -N 20 -K 180 -X 60 -w /opt/FMC/tests -e -o
      ./counter 123
```

4.2 The `pcRm` Command-Line Client

4.2.1 Synopsis

`pcRm [-v] [-C ctrl_pc_pattern...] [-m node_pattern...] [-s Signal]`
`     [-d kill9_delay] utgid_pattern`
`pcRm [-h]`

4.2.2 Description

Remove the processes whose UTGID matches the wildcard pattern **`utgid_pattern`** from the list for the controlled nodes whose hostname matches at least one of the wildcard patterns **`node_pattern`**, which are controlled by the control PCs whose hostname matches at least one of the wildcard patterns **`ctrl_pc_pattern`** and **terminates** these processes by sending them the signal **`Signal`** and — if they block or ignore the signal — by sending them the **`KILL`** signal after **`kill9_delay`** seconds.

4.2.3 Command-Line Options

`-h` — Print a help message and exit.

`-v` — Print verbose output.

`-C ctrl_pc_pattern (repeatable)` — Contact the Process Controller Servers (**`pcSrv`**) of the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`** (*default*: contact all the control PCs).

`-m node_pattern (repeatable)` — Remove the processe from the nodes (controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**) whose hostname matches the wildcard pattern **`node_pattern`** (*default*: remove processes from all the nodes controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**).

All other option are described in section 3.5.2 at page 10.

4.2.4 Environment

The program **pcRm** needs the two environment variables:

DIM_DNS_NODE — **hostname.domain** of DIM dns node.

LD_LIBRARY_PATH — Variable, in PATH format, which must contain the path to the shared libraries **libdim.so** and **libFMCutils.so**.

4.2.5 Warning

The *wildcards* [9] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

4.2.6 Examples

```
pcRm counter_0
pcRm "count*"
pcRm count\*
pcRm "count*[2-5]"
pcRm -C ctrl01 -m farm0101 counter_0
pcRm -C ctrl01 -C ctrl02 -m farm0101 -m farm0201 counter_0
pcRm -C "ctrl0?" -C "ctrl1?" -m "farm01?" -m "farm02?" "counter_?"
pcRm -C ctrl0\? -C ctrl1\? -m farm01\? -m farm02\? counter_\?
pcRm -C "ctrl0*" "count*"
pcRm -C ctrl0\* count\*
pcRm -C "ctrl[3-7]?" "count*[2-5]"
pcRm -s 2 counter_0
pcRm -s 2 "count*[2-5]"
pcRm -C ctrl01 -s 2 counter_0
pcRm -C "ctrl0*" -s 2 "count*"
pcRm "*"
pcRm \*
pcRm -s 2 -d 4 "*"
pcRm -s 2 -d 4 \*
pcRm -C "ctrl0[1357]" -m farm0101 -s 2 -d 4 counter_0
```

4.3 The pcLs Command-Line Client

4.3.1 Synopsis

pcLs [-v] [-C ctrl_pc_pattern...] [-m node_pattern...]

pcLs [-h]

4.3.2 Description

List the processes in the list for the nodes whose hostname matches at least one of the wildcard patterns **node_pattern**, which are controlled by the control PCs whose hostname matches at least one of the wildcard patterns **ctrl_pc_pattern**.

4.3.3 Command-Line Options

`-h` — Print a help message and exit.

`-v` — Print verbose output.

`-C ctrl_pc_pattern (repeatable)` — Contact the Process Controller Servers (**`pcSrv`**) of the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`** (*default*: contact all the control PCs).

`-m node_pattern (repeatable)` — List the processes in the list for the nodes (controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**) whose hostname matches the wildcard pattern **`node_pattern`** (*default*: list the processes in the list for all the nodes controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**).

4.3.4 Environment

The program **`pcLs`** needs the two environment variables:

`DIM_DNS_NODE` — **`hostname.domain`** of DIM dns node.

`LD_LIBRARY_PATH` — Variable, in PATH format, which must contain the path to the shared libraries **`libdim.so`** and **`libFMCutils.so`**.

4.3.5 Warning

The *wildcards* [9] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

4.3.6 Examples

```
pcLs
pcLs -m farm0101
pcLs -m farm010\*
pcLs -m "farm010*"
pcLs -m "farm0[2-5]???" -m "far[a-z,A-Z]0101"
pcLs -C ctrl01
pcLs -C ctrl0\*
pcLs -C "ctrl0*"
pcLs -C ctrl01 -m farm0101
pcLs -C ctrl0\* -m farm010\*
pcLs -C "ctrl0*" -m "farm010*"
```

4.3.7 Output sample

```
[online@lhcbmonitor ~]$ pcLs
Ctrl Pc patterns: "*".
Node patterns:    "*".
  Ctrl PC pattern: "*"
    Processes scheduled by Process Controller  0: "ctrl01":
    0     farm0101:5:tcpipSrv_u,psSrv_u,nifSrv_u,memSrv_u,irqSrv_u.
    1     farm0102:5:tcpipSrv_u,psSrv_u,nifSrv_u,memSrv_u,irqSrv_u.
```

4.4 The `pcLl` Command-Line Client

4.4.1 Synopsis

`pcLl [-v] [-c] [-C ctrl_pc_pattern...] [-m node_pattern...]`

`pcLl [-h]`

4.4.2 Description

List the processes — together with their **start parameters** described in 3.5.4 at page 10 — in the list for the nodes whose hostname matches at least one of the wildcard patterns **`node_pattern`**, which are controlled by the control PCs whose hostname matches at least one of the wildcard patterns **`ctrl_pc_pattern`**.

4.4.3 Command-Line Options

`-h` — Print a help message and exit.

`-v` — Print verbose output.

`-c` — Print compact output.

`-C ctrl_pc_pattern (repeatable)` — Contact the Process Controller Servers (**`pcSrv`**) of the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`** (*default*: contact all the control PCs).

`-m node_pattern (repeatable)` — List the processes in the list for the nodes (controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**) whose hostname matches the wildcard pattern **`node_pattern`** (*default*: list the processes in the list for all the nodes controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**).

4.4.4 Environment

The program **`pcLl`** needs the two environment variables:

`DIM_DNS_NODE` — **`hostname.domain`** of DIM dns node.

`LD_LIBRARY_PATH` — Variable, in `PATH` format, which must contain the path to the shared libraries **`libdim.so`** and **`libFMCutils.so`**.

4.4.5 Warning

The *wildcards* [9] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

4.4.6 Examples

```
pcLl
pcLl -c
pcLl -m farm0101
pcLl -m farm010\*
pcLl -m "farm010*"
pcLl -m "farm0[2-5]???" -m "far[a-z,A-Z]0101"
pcLl -C ctrl01
```

```
pcLl -C ctrl0\*
pcLl -C "ctrl0*"
pcLl -C ctrl01 -m farm0101
pcLl -C ctrl0\* -m farm010\*
pcLl -c -C "ctrl0*" -m "farm010*"
```

4.4.7 Output sample

```
[online@lhcbmonitor ~]$ pcLl
Ctrl Pc patterns: "*".
Node patterns:    "*".
  Ctrl PC pattern: "*"
    Processes scheduled by Process Controller  0: "ctrl01":
    farm0101: 2 scheduled process(es)
      UTGID: tcpipSrv_u
        path="/opt/FMC/sbin/tcpipSrv"
        DIM_DNS_NODE=ctrl01.daq.lhcb
        LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
        SCHED_OTHER
        nice=0
        rtprio=0
        affMask=0xffffffff
        maxStartN=10
        disPeriod=300s
        checkPeriod=120s
      UTGID: nifSrv_u
        path="/opt/FMC/sbin/nifSrv"
        DIM_DNS_NODE=ctrl01.daq.lhcb
        LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
        SCHED_OTHER
        nice=0
        rtprio=0
        affMask=0xffffffff
        user="root"
        maxStartN=10
        disPeriod=300s
        checkPeriod=120s
    farm0102: 2 scheduled process(es)
      UTGID: tcpipSrv_u
        path="/opt/FMC/sbin/tcpipSrv"
        DIM_DNS_NODE=ctrl01.daq.lhcb
        LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
        SCHED_OTHER
        nice=0
        rtprio=0
        affMask=0xffffffff
        maxStartN=10
        disPeriod=300s
        checkPeriod=120s
      UTGID: nifSrv_u
        path="/opt/FMC/sbin/nifSrv"
        DIM_DNS_NODE=ctrl01.daq.lhcb
        LD_LIBRARY_PATH=/opt/dim/linux:/opt/FMC/lib
        SCHED_OTHER
        nice=0
        rtprio=0
        affMask=0xffffffff
```

```
user="root"
maxStartN=10
disPeriod=300s
checkPeriod=120s
```

4.5 The `pcLss` Command-Line Client

4.5.1 Synopsis

`pcLss [-v] [-c] [-C ctrl_pc_pattern...] [-m node_pattern...]`

`pcLss [-h]`

4.5.2 Description

List the processes — together with the **number of the starts** and the **date and the time of each start**, as described in 3.5.5 at page 11 — in the list for the nodes whose hostname matches at least one of the wildcard patterns **`node_pattern`**, which are controlled by the control PCs whose hostname matches at least one of the wildcard patterns **`ctrl_pc_pattern`**.

4.5.3 Command-Line Options

`-h` — Print a help message and exit.

`-v` — Print verbose output.

`-c` — Print compact output.

`-C ctrl_pc_pattern (repeatable)` — Contact the Process Controller Servers (**`pcSrv`**) of the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`** (*default*: contact all the control PCs).

`-m node_pattern (repeatable)` — List the processes in the list for the nodes (controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**) whose hostname matches the wildcard pattern **`node_pattern`** (*default*: list the processes in the list for all the nodes controlled by the control PCs whose hostname matches the wildcard pattern **`ctrl_pc_pattern`**).

4.5.4 Environment

The program **`pcLss`** needs the two environment variables:

`DIM_DNS_NODE` — **`hostname.domain`** of DIM dns node.

`LD_LIBRARY_PATH` — Variable, in `PATH` format, which must contain the path to the shared libraries **`libdim.so`** and **`libFMCutils.so`**.

4.5.5 Warning

The *wildcards* [9] in the command line must be *escaped* (by means of a *back-slash* or by means of a couple of *double quotation marks*) in order to avoid their shell expansion.

4.5.6 Examples

```
pcLss
pcLss -c
pcLss -m farm0101
pcLss -m farm010\*
pcLss -m "farm010*"
pcLss -m "farm0[2-5]???" -m "far[a-z,A-Z]0101"
pcLss -C ctrl01
pcLss -C ctrl0\*
pcLss -C "ctrl0*"
pcLss -C ctrl01 -m farm0101
pcLss -C ctrl0\* -m farm010\*
pcLss -c -C "ctrl0*" -m "farm010*"
```

4.5.7 Output sample

```
[online@lhcbmonitor ~]$ pcLss
Ctrl Pc patterns: "*".
Node patterns:    "*".
  Ctrl PC pattern: "*"
    Processes scheduled by Process Controller  0: "ctrl01":
    farm0101: 3 scheduled process(es)
      UTGID: fsSrv_u, start=3:
        Jun28-165258
        Jul03-134054
        Jul03-134055
      UTGID: tcpipSrv_u, start=1:
        Jun28-165128
      UTGID: nifSrv_u, start=1:
        Jun28-165158
    farm0102: 3 scheduled process(es)
      UTGID: fsSrv_u, start=1:
        Jun28-165258
      UTGID: tcpipSrv_u, start=1:
        Jun28-165128
      UTGID: nifSrv_u, start=1:
        Jun28-165158
```

5 Component versions

The current documentation refers to the Process Controller software contained in **FMC** version **3.6.10**. In particular, the versions of the Process Controller components, which can be retrieved with the Linux **`ident`** command applied to the executables, are shown in Table 1.

Table 1 Versions of the Process Controller components in FMC v. 3.6.10.

Executable component	Source(s)	Version
`pcSrv`	**`pcSrv.c`**	1.27
`pcAdd`	**`pcAdd.C`**	1.10
`pcRm`	**`pcRm.C`**	1.5
`pcLs`	**`pcLs.C`**	1.4
`pcLl`	**`pcLl.C`**	1.6
`pcLss`	**`pcLss.C`**	1.7

6 Conclusion

We have realized a software tool — the Process Controller — in charge of keeping a list of applications up and running on the farm nodes (by quickly restarting unintentionally terminated processes), in collaboration with the Task Manager [1]. It is composed of a server process, which tipically runs on a few *control PCs* each one watching ∼200 farm nodes, and a client process to be used for sending commands to the server and to retrieve the status information (the list of scheduled applications for each controlled farm node together with their properties, the number of re-spawns and the re-spawn times). Processes can be added or removed on-the fly from the list for one or more nodes by means of DIM commands sent by the client to the server.

7 References

[1] F. Bonifazi, D. Bortolotti, A. Carbone, D. Galli, D. Gregori, U. Marconi, G. Peco, and V. M. Vagnoni, "The Task Manager for the LHCb on-line farm," CERN, LHCb note 2004-099 DAQ, Nov. 24, 2004. [Online]. Available: http://doc.cern.ch/archive/electronic/cern/others/LHB/public/lhcb-2004-099.pdf

[2] C. Gaspar and M. Dönszelmann, "DIM – A Distributed Information Management System for the DELPHI experiment at CERN," in *Proceedings of the 8th Conference on Computer Applications in Nuclear, Particle and Plasma Physics*. Vancouver, BC, Canada: IEEE, Jun 1993, pp. 156–158. [Online]. Available: http://dim.web.cern.ch/dim/papers/DIM.PS

[3] init(8) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man8/init.8.html

[4] inittab(5) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man5/inittab.5.html

[5] sigtimedwait(2) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man2/sigtimedwait.2.html

[6] sched_setscheduler(2) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man2/sched_setscheduler.2.html

[7] setpriority(2) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man2/setpriority.2.html

[8] PVSS-DIM Integration. [Online]. Available: http://lhcb-online.web.cern.ch/lhcb-online/ecs/fw/FwDim.html

[9] glob(7) – Linux man page. [Online]. Available: http://www.die.net/doc/linux/man/man7/glob.7.html