OVSA SPI for Service Providers

Equipment Connections Pools Administrator Reference

Release v.5.1

Legal Notices

Warranty.

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices.

Trademark Notices.

Java™ is a U.S. trademark of Sun Microsystems, Inc.

Linux is a U.S. registered trademark of Linus Torvalds

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of the Open Group.

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Table of Contents

Support

Support for the HP Open View Service Activator SPI product is available on the following mailing list:

ovsa.spain.support@hp.com

In This Guide

This guide will explain the SO and File System resources needed by an ECP instance, such as IPs, ports, configuration files etc... The non programmatic tools offered for monitoring and operating on the ECP are also explained.

Audience

The audience for this guide is the Systems Administrator (SA). The SA has a combination of some or all of the following capabilities:

Understands and has a solid working knowledge of:

- UNIX® commands
- Windows® system administration

Understands networking concepts and language

Understands Java™ and XML

Understands security issues

Conventions

The following typographical conventions are used in this guide.

Font	What the Font Represents	Example
Italic	Book or manual titles, and man page names	Refer to the *HP Service Activator — Workflows and the Workflow Manager* and the *Javadocs* man page for more information.
	Provides emphasis	You *must* follow these steps.
	Specifies a variable that you must supply when entering a command	Run the command: `java -classpath` *<classpath>*
	Parameters to a method	The *assigned_criteria* parameter returns an ACSE response.
Bold	New terms	The **distinguishing attribute** of this class...
`Computer`	Text and items on the computer screen	The system replies: `Press Enter`
	Command names	Use the `java` command ...
	Method names	The `get_all_replies()` method does the following...
	File and directory names	Edit the file `$ACTIVATOR_ETC/config/mwfm.xml`
	Process names	Check to see if `mwfm` is running.
	Properties files keys names	Set the property `LOG_DIR` to establish the log files path.
	Window/dialog box names	In the `Test and Track` dialog...
	XML tag references	Use the `<DBTable>` tag to...
`Computer Bold`	Text that you must type	At the prompt, type: **`ls -l`**
Keycap	Keyboard keys	Press **Return**.
[Button]	Buttons on the user interface	Click `[Delete]`. Click the `[Apply]` button.
Menu Items	A menu name followed by a colon (:) means that you select the menu, then the item. When the item is followed by an arrow	Select `Locate:Objects->by Comment`.

Font	What the Font Represents	Example
	(->), a cascading menu follows	

Install Location Descriptors

The following names are used throughout this guide to define install locations.

Descriptor	What the Descriptor Represents
$ACTIVATOR_OPT	The base install location of Service Activator. The UNIX location is `/opt/OV/ServiceActivator` The Windows location is `<drive>:\HP\OpenView\ServiceActivator\`
$ACTIVATOR_ETC	The install location of specific Service Activator configuration files. The UNIX location is `/etc/opt/OV/ServiceActivator` The Windows location is `<drive>:\HP\OpenView\ServiceActivator\etc\`
$ACTIVATOR_VAR	The install location of specific Service Activator logging files. The UNIX location is `/var/opt/OV/ServiceActivator` The Windows location is `<drive>:\HP\OpenView\ServiceActivator\var\`
$ACTIVATOR_BIN	The install location of specific Service Activator binary files. The UNIX location is `/opt/OV/ServiceActivator/bin` The Windows location is `<drive>:\HP\OpenView\ServiceActivator\bin\`
$ACTIVATOR_THIRD_PARTY	The location for new Java components such as workflow nodes and modules. Third-party libraries can also be placed in this directory. The UNIX location is `/opt/OV/ServiceActivator/3rd-party` The Windows location is `<drive>:\HP\OpenView\ServiceActivator\3rd-party\` Customized inventory files are stored in the following locations: UNIX: `$ACTIVATOR_THIRD_PARTY/inventory` Windows: `$ACTIVATOR_THIRD_PARTY\inventory`
$JBOSS_HOME	HOME The install location for JBoss. The UNIX location is `/opt/HP/jboss` The Windows location is `<drive>:\HP\jboss`
$JBOSS_DEPLOY	The install location of the Service Activator J2EE components. The UNIX location is `/opt/HP/jboss/server/default/deploy`

	The Windows location is `<drive>:\HP\jboss\server\default\deploy`
`$ACTIVATOR_DB_USER`	The database user name you define. Suggestion: `ovactivator`
`$ACTIVATOR_SSH_USER`	The Secure Shell user name you define. Suggestion: `ovactusr`
`$SOSA_HOME`	The base install location of SOSA. The UNIX location is `/opt/OV/Sosa` The Windows location is <drive>:\HP\OpenView\Sosa\
`$SOSA_BIN`	The install location of specific SOSA binary files. The UNIX location is `/opt/OV/Sosa/bin` The Windows location is `<drive>:\HP\OpenView\Sosa\bin\`
`$SOSA_ETC`	The install location of specific SOSA configuration files. The UNIX location is `/opt/OV/Sosa/config` The Windows location is <drive>:\HP\OpenView\Sosa\config\
`$ECP_HOME`	The base install location of Equipment Connections Pool. The UNIX location is `/opt/OV/ECP` The Windows location is <drive>:\HP\OpenView\ECP\
`$ECP_BIN`	The install location of specific Equipment Connections Pool binary files. The UNIX location is `/opt/OV/ECP/bin` The Windows location is <drive>:\HP\OpenView\ECP\bin\
`$ECP_ETC`	The install location of specific Equipment Connections Pool configuration files. The UNIX location is `/opt/OV/ECP/conf` The Windows location is <drive>:\HP\OpenView\ECP\conf\
`$ECP_LIB`	The install location of specific Equipment Connections Pool jar files. The UNIX location is `/opt/OV/ECP/lib` The Windows location is <drive>:\HP\OpenView\ECP\lib\
`$ECP_LOG`	The install location of specific Equipment Connections Pool log files. The default UNIX location is `/opt/OV/ECP/log` The default Windows location is <drive>:\HP\OpenView\ECP\log\

1. Introduction

1.1. Purpose

This document is a manual for all ECP Module administrators. It describes the tools offered by the ECP Module and the SO resources it will need.

1.2. General Description

The function of the ECP Module, as part of the SPI, is automating user interactive textual sessions, via TCP/IP connections to networked devices, such as routers, switches, proxies, etc…

The ECP Module receives a textual representation of the session, which states the commands to issue, their output and their meanings, and the control flow logic (such as the conditions under which a command must be issued or how many times must be issued).

The ECP Module is the module in SPI which in the last instance directly connects to the SPI managed devices, centralizing the SPI management connections. This situation inside the SPI framework is ideal to perform task such as load balancing, high availability and resources use optimization when referring to management connections.

The ECP Module is divided in two elements, the ECP Client and the ECP Service (an RMI service). The ECP Service receives the representations of the sessions and actually executes them, and contains the Pool Manager. The ECP Client acts mainly as a proxy, easing access to the ECP Service. It also allows the user to totally bypass the ECP RMI Service if needed, being the process transparent to the user. Bypassing the ECP Service is known as “Direct Connection” as opposed to “Pooled Connections” when using the ECP RMI Service. The use of either method is transparent to the user.

Such division allows easier scalability of the SPI, while maintaining the ECP Module objectives of load balancing, high availability and resources use optimization.

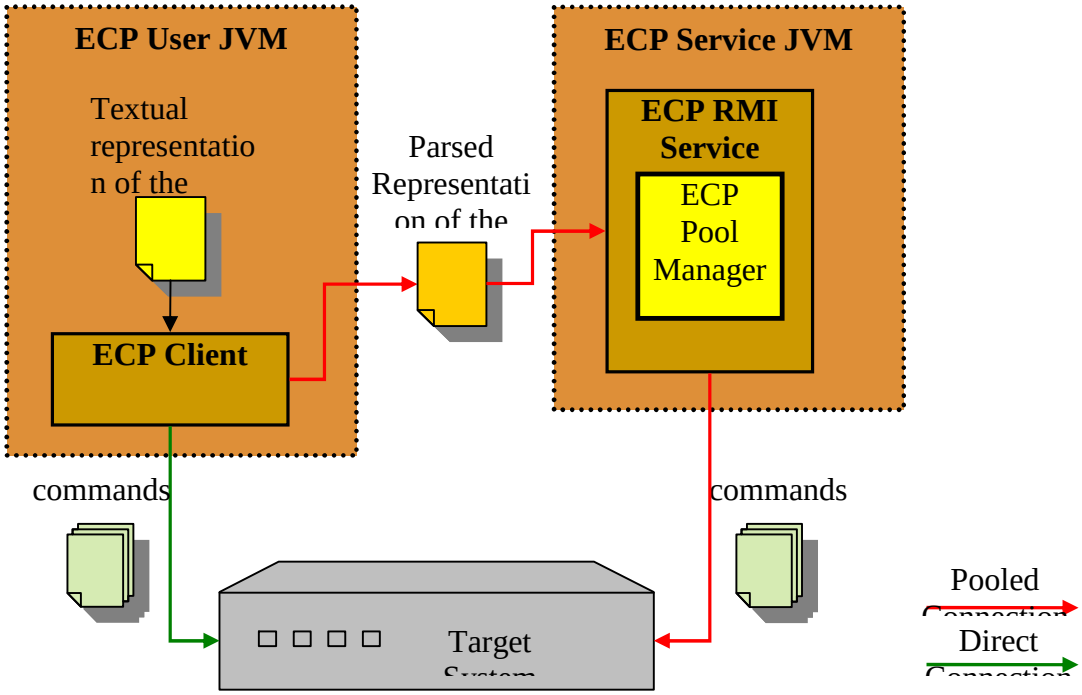


Figure 1: ECP Simplified General diagram.

1.3. ECP Module Entities and Concepts

1.3.1. Target System

In the context of the ECP, a "Target System" is the collection of resources accessible through a single direct TCP/IP Connection. Usually, a "Target System" will be a single router, switch or other similar device. However, more complex scenarios are possible if other devices are accessed from the connection end-point.

1.3.2. Operation

By "Operation" we refer to the collection of commands and logic needed to perform a certain process on the Target System. The purpose of the process may be a data inquiry, a configuration change or any other action needed on a Target System. An "Operation" should be atomic, that is, it should completely occur, or have no effects on the Target System. As a consequence, "Operations" should include the commands and logic needed to rollback the changes on the Target System if any. However, this policy is not enforced. Its use is left to the user's discretion.

1.3.3. Commands Template/Operation Template

A "Command Template" is a string which complies with a certain syntax through which an Operation is expressed, for the ECP Module to interpret and process it, usually with the purpose of automating a human interactive session on the Target System. The "Command Template" states the commands needed

to perform the process (and usually to roll it back too), with specific information on every command, such as possible command outputs and their meaning (error, success) and the control flow which determines their execution order, among other things.

1.3.4. Operation Execution

An "Operation Execution" is the process through which Command Template is processed, resulting in commands inputted into the Target System.

1.3.5. Resource

In the context of the ECP, "Resource" is synonym of connection instance.

1.3.6. Pool

In the context of the ECP, a "Pool" is a set of established and authenticated connections (resources) to a single Target System that are kept ready to use. Each connection instance belongs to a single "Pool". Connection instances life time is managed by the "Pool". Pools are identified by name.

1.3.7. SubPool

A "SubPool" is a subset of the connections belonging to a Pool which are established with the Target System through the same interface, what generally implies though the same IP and Port (and user). The existence of the "SubPool" is only needed in the context of the ECP Configuration and Administration. In other contexts its use is transparent to the user. Each SubPool belongs to a single Pool. Every connection belongs to a single SubPool.

1.3.8. Equipment Driver

An "Equipment Driver" is a class whose instance encapsulates a single TCP/IP connection as a Pool Resource and is in charge of establishing, authenticating, verifying, and closing the underlying connection, when required by the Pool and as needed by the Target System. As some of this processes (especially authenticating, verifying and closing the connection) are dependent on the Target System type, usually a different "Equipment Driver" is needed for each Target System type, hence its name. It allows the developer and designer to easily add functionality to the ECP on per connection, per equipment, per equipment connection or even on connection event basis. Equipment Drivers must be provided by the ECP User.

The "Equipment Driver" is also in charge of executing every individual Commands Template command, that is: composing the Target System command, sending it to the Target System, reading the Target system answer, and interpreting it. Nevertheless, this functionality is provided by the ECP through inheritance.

For some tasks (such as establishing and closing the connection, or sending and reading data from it), the Equipment Driver will usually rely on a Protocol Driver to perform them as very often those task are not dependant on the Target System type, but on the network protocol to communicate with it. Entrusting this task on the Protocol Driver allows the programmer to reuse network protocol dependant functionality.

Typically, a different Equipment Driver is needed for each model of switch or router.

In the context of the ECP, the terms "connection", "Resource", and "Equipment driver", are interchangeable.

1.3.9. Default Equipment Driver

This driver is able to connect to any type of equipments using some variables or templates. The most important fearture of this driver is the capability to connect any equipment and not require any java development.

This equipment driver is configured using the class com.hp.spain.connection.TemplateDriver. We can configure this driver adding into the DriverSpecificParameters the extra variables on properties format or referring to the Common Configuration.

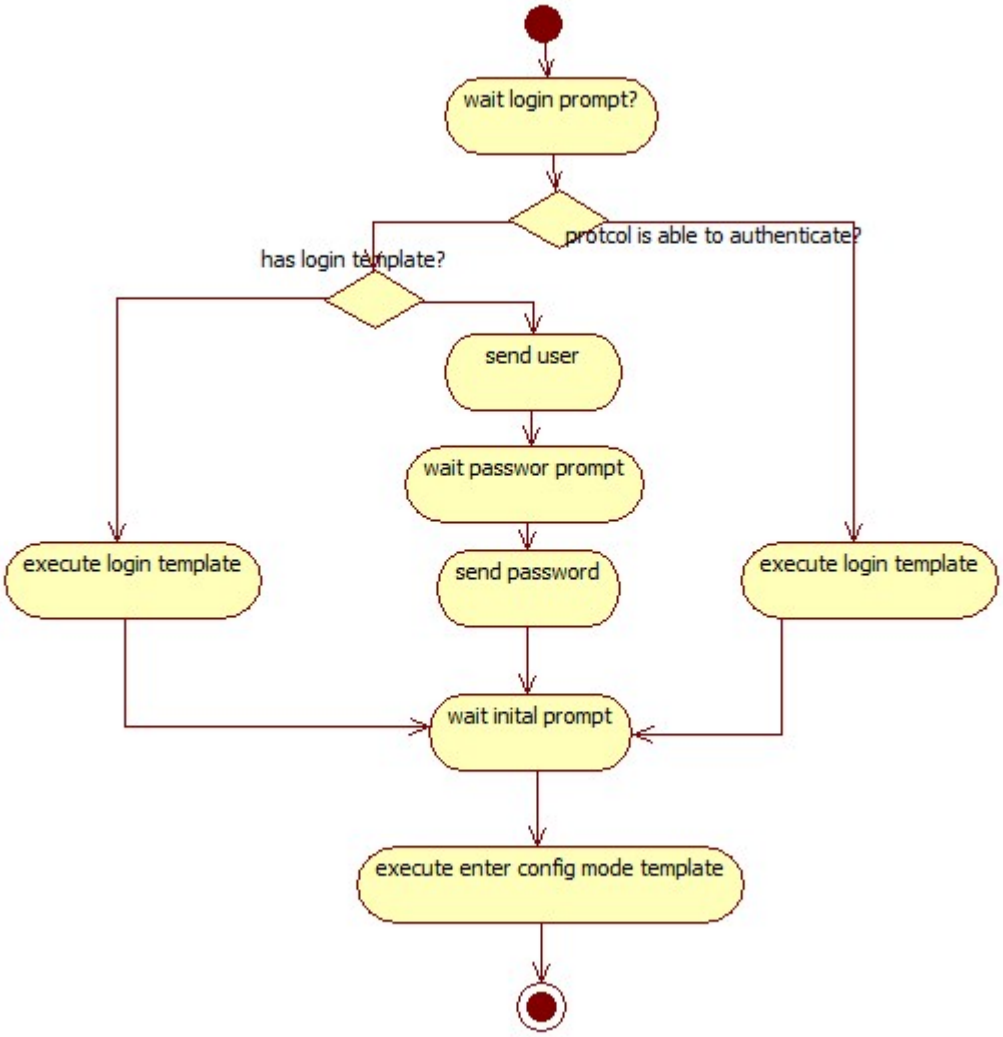
To connect, when the driver starts the connection the first step is to check if the LOGIN_USER_PROMPT is configured. In that case, synchronize this prompt. After that, the LOGIN_TEMPLATE is executed if it's configured. If not and the protocol driver doesn't support authentication, send the user, synchronize the password prompt (LOGIN_PWD_PROMPT) and send the password.

In this moment, the driver is authenticated and in case the INITIAL_PROMPT is configured the driver synchronizes the initial prompt.

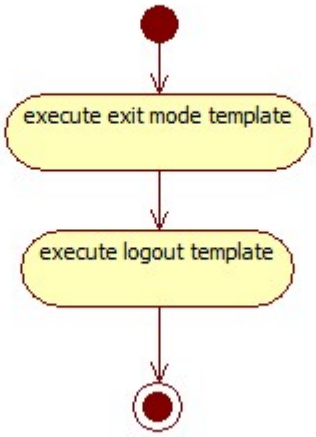
Usually, when the protocol supports the authentication (for example, ssh) it's only necessary to configure the INITIAL_PROMPT and not the LOGIN_TEMPLATE and neither LOGIN_USER_PROMPT.

After synchronize the INITIAL_PROMPT the driver execute the ENTER_CONFIG_MODE_TEMPLATE and finally executes the VERIFY_TEMPLATE.

In this moment, the driver is connected and ready to be used.



To disconnect, first, the driver execute the template EXIT_CONFIG_MODE_TEMPLATE and after that the LOGOUT_TEMPLATE.



1.3.10. Protocol Driver

A "Protocol Driver" is a class whose instance encapsulates a single TCP/IP connection, and is in charge of establishing and closing the connection, sending and reading data from it, and encoding and decoding those data as needed by the Target System interface. Generally speaking, a Protocol Driver provides partial or total independence from the Application Layer of the OSI model. Entrusting this task on the protocol driver allows the programmer to reuse network protocol dependant functionality.

The ECP provides Protocol Drivers for Telnet, SSH, and raw TCP network protocols.

1.3.11. ECPJmsModule

The ECPJmsModule is a mwfm manager module that listen jms message generated by ECP. Using this module and the node ECPCall is possible to save the executions into the database. The full output of the terminal , dates, commands sent, … will be saved.

1.3.12. ECPCall node

The ECPCall node is the easiest way to call the an activation from a HPSA workflow. See QuickStartGuide for further information.

2. Functionality and Architecture

2.1. Connection and Pool Management

A single instance of the Pool Manager exists in the ECP Service. The Pool Manager contains a single Pool for each Target System (in a typical configuration).

Each Pool contains all the connections to a Target System, and is responsible of their life time and management. Additionally, it is responsible for:

a) Connections reuse. The connections are kept alive, opened and authenticated, reusing the connections while possible.

b) Identifying redundant interfaces on the Target System, and their connections, providing high availability.

c) Queuing and prioritizing the Operation Engines' requests for connection to the Target System, providing load balance.

d) Target System independence.

e) Protocol independence.

See Figure 2: Pool Manager Architecture

2.1.1. Connection Reuse

Opening and maintaining a connection for each user is costly and wastes resources. On the contrary, pooling the connections enhances the performance of executing commands on a Target System. After a connection is created, it is placed in the Pool and reused over again while possible so that another connection does not have to be established and authenticated. The Pool creates (`initialize`) and destroys (`finalize`) new connections as needed, not exceeding the configured limits and politics. Connections are verified for consistency before being assigned to a client (`verify`). Additionally pooling the connections allows abstracting the client of the details of the connections management. Pooling the connections achieves reliable connections reuse. See Figure 2: Pool Manager Architecture

2.1.2. High Availability

Every Pool may have one or more SubPools. Each SubPool represents a connection factory and container. Every SubPool comply the following rules:

a) Each SubPool "owns" a different Target System interface. This means that all ECP connections to that Target System through that interface should be created and contained by the same SubPool instance.

b) Connections from different SubPools should be equivalent, that is, executing an Operation through one or another SubPool should have the same effects on the Target System (provided the same initial Target System State).

Complying with this rules, allows the ECP to temporarily ignore a SubPool (interface) if it fails and becomes unusable (and another SubPool exists in the Pool), using the other SubPools (interfaces) instead. SubPooling the connections achieves high availability. See Figure 2: Pool Manager Architecture.

2.1.3. Target System Independence

The ECP needs to be able to connect, login, verify and disconnect the connections to the Target Systems as part of the Pooled connections management. As these processes are Target System specific, the ECP is unable to do so by itself. As a consequence, the ECP User must provide an Equipment Driver which performs those operations on behalf of the ECP. The Equipment Driver will wrap a connection, abstracting the ECP from the real tasks needed for those operations. Roughly speaking, the Equipment Driver scope is at a "per command" level. See Equipment Driver and Figure 2: Pool Manager Architecture.

2.1.4. Protocol Independence

Although Equipment Drivers perform Target System specific tasks, the underlying network protocol is usually standardized, and is not Target System dependant. For example, is very common for Target Systems to use SSH or Telnet protocols. To ease Equipment Driver development and allow protocol interchangeability, a Protocol Layer abstraction layer is implemented, called "Protocol Driver". That layer will be responsible for establishing and closing the connection, sending and reading data from it, and encoding and decoding those data as needed by the Target System interface.

The ECP provides Protocol Drivers for Telnet, SSH, and raw TCP network protocols. See Protocol Driver and Figure 2: Pool Manager Architecture.

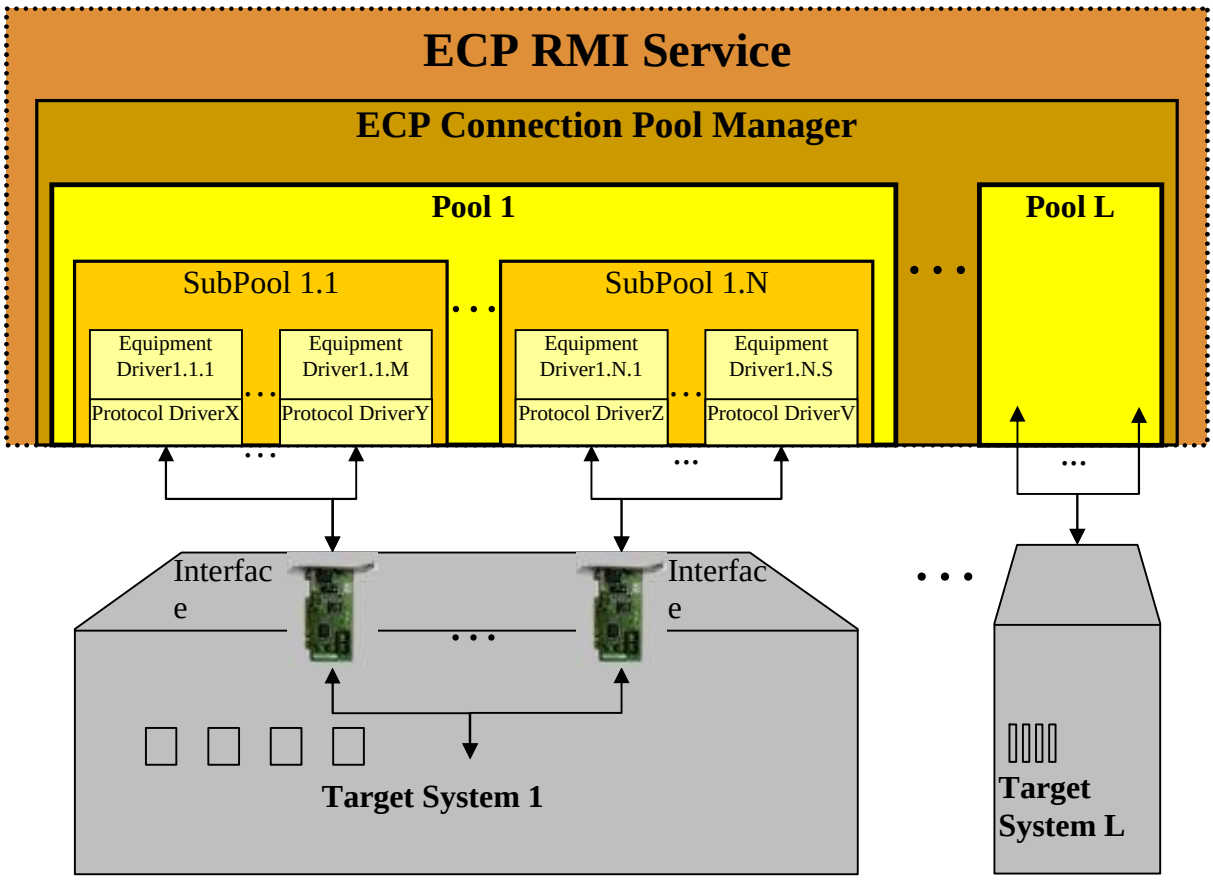


Figure 2: Pool Manager Architecture.

2.1.5. Load Balance

Every Pool, has a configurable number of "Request Queues", where clients in need of a connection are kept waiting for their turn to acquire a connection. Queues can be prioritized, allowing critical Operations to remain as short as possible waiting for an available connection, and avoiding clients from becoming starved because of a high not critical Operations load. The priority of each request is established programmatically.

The frequency at which requests are dispatched and the number of available connections on each SubPool can be configured, allowing management of the load over the Target System and the ECP host. Request queuing and Pool size achieve load balance.

2.2. Pool and Connection types

Two types of Connection Pools are available, depending on how the pools are created.

2.2.1. Static vs Temporary Pools

ECP Module provides two different types of Pools: Static and Temporary.

Functionally, Temporary Pools are exactly the same as Static Pools, the only difference being that Temporary Pools will expire if unused for a configured amount of time, while static Pools will never expire.

Temporary Pools are useful when a Target System is going to be used for a short period of time and remain unused for long periods. Temporary Pools allow saving host resources in such situation.

When Pools are used, the Operation Execution is delegated on the ECP Service. See **Error! Reference source not found.iError! No se encuentra el origen de la referencia.**.

2.2.2. Direct Connections (Not Pooled Connections)

When using Direct Connections, a connection is created for each executed Operation, being the connection private to the ECP Operation Engine instance used to issue the Operation. The Connection exists in the context of the ECP Operation Engine instance JVM. The Operation is executed in the JVM of the client. No ECP RMI Service is needed for this kind of Operation, although the Equipment Driver and Protocol Driver and their libraries will be needed. See **Error! Reference source not found.iError! No se encuentra el origen de la referencia.**.

2.2.3. Dynamic Pools

The ECP Module allows the user to programmatically create Pools. Programmatically created Pools are referred "Dynamic Pools". Dynamic Pools are usually temporary, although they can be static. As a consequence, "Dynamic Pools" aren't created independently, but as part of the Operation Executions which uses them. This is due to the fact that a client can't know whether the Dynamic Temporary Pool will still exist when the Operation Execution call is processed by the RMI ECP Service. For these reason, Operation Executions which use Dynamic Pools always carry the Dynamic Pool definition. On arrival to the ECP Service, the Dynamic Pool will be created if it does not exist. If it exists, the running Dynamic Pool instance will be used.

2.3. Commands Template

As "Commands Template" we understand a specially crafted String where, using a syntax specified by the ECP, the commands to Do, Undo, Commit and Rollback the Operation are established.

A Velocity Engine version 1.4 is provided with ECP, to ease the implementation of dynamic Commands Templates for the user. Through the method `TemplateParser#composeTemplate()`, a Velocity Commands Template can be easily merged with the data. See http://velocity.apache.org/ for more details. See **Error! Reference source not found.iError! No se encuentra el origen de la referencia.**.

What follows is an example of a possible Commands Template:

```
[TEMPLATE:Do]


[TEMPLATE:Section 0]

show eth0 connections
  [TEMPLATE:EndStrPattern "admin#"]
  [TEMPLATE:Pattern "detination IP: (.*)"]
  [TEMPLATE:Array "destinationIPs"]

show eth1 connections
  [TEMPLATE:EndStrPattern "admin#"]
  [TEMPLATE:Pattern "detination IP: (.*)"]
  [TEMPLATE:Array "destinationIPs"]


[TEMPLATE:ForEach "var" In " destinationIPs"]
      ping  %var% -n 1
            [TEMPLATE:EndStrPattern "admin#"]
[TEMPLATE:EndFor]

[TEMPLATE:Undo]
[TEMPLATE:Section 0]
```

The previous template queries connections through `eth0`, storing the destination IP in the array variable `destinationIPs`. The same process is repeated on eth1. After that, a ping is executed to all the obtained IPs. All commands are over when the prompt `admin#` is encountered. As the Template does not modify the Target System state, no Undo commands are needed.

2.4. Realtime Monitoring

From SPI version 2.3 onwards the ECP is able to provide real time information of its execution through JMS. Currently, ECP includes Active MQ 4.1.1 which fully implements JMS 1.1. If JMS monitoring is enabled, ECP may start its own embedded JMS service (by default) or connect to a remote one (see ecp.properties). Active MQ includes many features, like persintent, transactional and XA messaging; message groups, virtual destinations, wildcards and composite destinations; pluggable transport protocols as TCP, SSL, UDP, in-VM (embedded); clustering; bridging to other JMS providers; JMX administration, etc…

2.4.1. ECPJmsModule

The ECP gives a workflow manager module that listen jms message generated by ECP. Next is a example of configuration:

```
    <Module>
        <Name>JmsModule</Name>
        <Class-Name>com.hp.spain.engine.module.jms.JmsModuleImpl</Class-
Name>
        <Param name="connectionfactoryjndiname"
value="TopicConnectionFactory"/>
        <Param name="destinationjndiname"
value="dynamicTopics/dynamicTopics/ECP.MainTopic"/>
        <Param name="acknowledgemode"                    value="1"/>
        <Param name="transactedsession"                  value="false"/>
        <Param name="receivelocalmessages"               value="true"/>
        <Param name="messajereceptiontimeout"            value="10000"/>
        <Param name="java.naming.factory.initial"
value="org.apache.activemq.jndi.ActiveMQInitialContextFactory"/>
        <Param name="java.naming.provider.url"
value="tcp://127.0.0.1:4001"/>
        <Param name="db"    value="db"/>
    </Module>
```

The feature of this module is used by ECPCall node. See QuickStart Reference for further information.

2.5. BackgroundCall module

The ECP gives a workflow manager module to call the ecp in background mode. The ECPCall node will connect to this module and add a activation. The node will wait on askfor until the backgroundCall module sends the result. It's possible to configure the number of maximum connections to ECP, if value is 0 there's no limit:

```
    <Module>
              <Name>EcpBackgroundModule</Name>
              <Class-
Name>com.hp.spain.connection.modules.BackgroundCallModule</Class-Name>
              <Param name="max_threads" value="20"/>
      </Module>
```

The feature of this module is used by ECPCall node. See QuickStart Reference for further information.

3. Main Tasks

3.1. ECP Service Process

3.1.1. Starting ECP Service

To start the ECP Service, use the following:

Windows:

```
$ECP_BIN\StartServer.bat
```

On Unix

```
$ECP_BIN\StartServer.sh
```

3.1.2. Stopping ECP Service

To stop the ECP Service, use the following:

Windows:

```
$ECP_BIN\StopServer.bat
```

On Unix

```
$ECP_BIN\StopServer.sh
```

3.1.3. Restarting ECP Service

Just stop and start the ECP Service.

3.1.4. Checking ECP Service

To check the ECP Service, use the following:

Windows:

```
$ECP_BIN\showStatus.bat
```

On Unix:

```
$ECP_BIN\showStatus.sh
```

If the ECP Module Server is started and responsive, the result of the command will be similar to:

```
Connecting to rmi://16.38.0.140:1200/
20081023-123853.395: com.hp.spain.connection.pool.server.RmiEcpClient: 1:
Finding rmi://16.38.0.140:1200/RmiEcpService
20081023-123853.697: com.hp.spain.connection.pool.server.RmiEcpClient: 1:
Service rmi://16.38.0.140:1200/RmiEcpService found:
com.hp.spain.connection.pool.server.RmiEcpService_Stub[RemoteStub [ref:
[endpoint:[16.38.0.140:1201](remote),objID:[7a84e4:11d1bc3f093:-8000, 0]]]]
20081023-123853.698: com.hp.spain.connection.pool.server.RmiEcpClient: 1:
Executing remote task with: {}
20081023-123853.723: com.hp.spain.connection.pool.server.RmiEcpClient: 1:
{result=The RMI Registry is up.}
```

```
20081023-123853.724: com.hp.spain.connection.pool.server.RmiEcpClient: 1:
Invoking service RmiEcpService
20081023-123853.771: com.hp.spain.connection.pool.server.RmiEcpClient: 1: []
20081023-123853.774: com.hp.spain.connection.pool.server.RmiEcpClient: 1:
POOL_MANAGER_RUNNING = true
20081023-123853.775: com.hp.spain.connection.pool.server.RmiEcpClient: 1:
Return: []
```

3.1.5. Identifying the ECP Service Process

The following command may be used in HP-UX to identify the EC Service Process

```
ps -ex|grep
java.rmi.server.codebase=file:.*/ECP/rmi_pub.*java.security.policy=.*/ECP/conf/RmiEcpService.policy
|cut -c 1-7
```

4. Configuration

4.1. ECP RMI Service Command Line Parameters

The Administrator should use the provided scripts to operate over the ECP Service. However, on some occasions the SA may need to call the Service by hand. The command line of the ECP RMI Server JVM has the following syntax:

```
<java_exe> -server -Djava.rmi.server.codebase=file:<ecp_home>\rmi_pub
-Djava.rmi.server.logCalls=false -
Djava.rmi.server.hostname=<ecp_rmi_server_ip>
-Djava.security.policy=<ecp_home>\conf\RmiEcpService.policy -
Dactivator.dir.config=<ecp_prot_drivers_dir> -classpath <ecp_libs>
com.hp.spain.connection.pool.server.RmiEcpService
<ecp_rmi_registry_server_host> <ecp_rmi_registry_server_port> <ecp_home>
```

<java_exe>: path to the JVM executable file. Of course, it is mandatory.

<ecp_home>: ECP installation directory. This parameter is mandatory. It will be used to establish the ecp.properties and RmiEcpService.policy files location, the ProtocolDrivers.lst file default location, and the default log directory.

<ecp_rmi_server_ip>: IP of the localhost, used by the locally created stubs to access the RMI server. Used by the JVM. This parameter is mandatory.

<ecp_rmi_registry_server_host>: Host name of the host where the RMI registry is located and where the ECP RMI service object should be bound. Normally it should refer to the localhost. This parameter is mandatory.

<ecp_rmi_registry_server_port>: Port number where the RMI registry accepts calls and where the ECP RMI service object should be bound. This parameter is mandatory.

<ecp_libs>: all the .jar and .zip files in the directory <ecp_home>\lib. This parameter is mandatory.

<ecp_prot_drivers_dir>: Directory where the ProtocolDrivers.lst file can be found. This parameter is optional. If omitted, <ecp_home>\conf will be used.

4.2. ecp.properties

The ecp.properties file is the main ECP configuration file. The rest of ECP configuration files should not be modified. From an administrative point of view, not all the values in the ecp.properties are meaningful.

As a consequence, only some of those values will be explained here. For a complete description of the values refer to the document "OVSA SPI for Service Providers - ECP - Developer Reference".

The ecp.properties file should be located in the path <ecp_**home>\conf**, being <ecp_home> the ECP installation directory. The ecp.properties files may contain the following properties (among others).

LOG_DIR: Logs directory. Most of the log data will be stored there. Its default value is

"C:\hp\OpenView\ServiceActivator\var\log" in windows and "/var/opt/OV/ServiceActivator/log/" in HP-UX.

It must end with the path separator character. This directory should exist and the user which executes the ECP RMI Service JVM must have writing permission over it. It will establish the directory where the Pool log files, the ProtocolDriver SpyFile the Configuration load log file.

LOG_MAX_FILE_SIZE: Will configure the RollingFileAppenders (when used) maximum file size (in bytes) before being rolled over to backup files. Its default value is 5242880 bytes (5MB).

LOG_MAX_NUM_FILES: Will configure the RollingFileAppenders (when used) maximum backup index (how many backup files are kept). Its default value is 10.

LOG_DATE_PATTERN: Will establish the type of Appenders used by the pools and configure the pools DailyRollingFileAppenders (when used) rolling date pattern. Its default value is null. It must be null or a valid SimpleDateFormat pattern (see http://java.sun.com/j2se/1.4.2/docs/api/java/text/SimpleDateFormat.html).

LOG_PATTERN: Will configure the messages pattern for the pools and the Configurator. Its default value is null.

DB_DRIVER: Fully qualified class name of a java.sql.Driver to load and register in the JDBC DriverManager. Its default value is null.

DB_USER: The DataBase user on whose behalf the connection is being made. Its default value is null.

DB_PASSWORD: The DataBase user password. Its default value is null.

DB_URL: A JDBC DataBase URL with the form: jdbc:<subprotocol>:<subname>. Its default value is null.

ECP.Msgs.Enable: Whether de ECP should perform JMS monitoring or not. If this option is disabled, no JMS monitoring messages will be sent, and the JMS configuration parameters will be ignored. Its default value is "false".

JMSBrokerReference.broker.uri: URI of the JMS service where ECP JMS monitoring messages will be sent. Ignored if "ECP.Msgs.Enable=false". By default it will start an embedded JMS broker. Its default value is "vm\:(broker\:(tcp\://localhost\:4001)?brokerName\=EmbeddedBroker&useJmx\=true&persistent\=false&populateJMSXUserID\=false&useShutdownHook\=false&deleteAllMessagesOnStartup\=false&enableStatistics\=false)?marshal\=false".

java.naming.factory.initial: The Initial context factory for JMS Administered objects. Ignored if "ECP.Msgs.Enable=false". Its default value is "org.apache.activemq.jndi.ActiveMQInitialContextFactory".

JMSMessageBroker.dest.type: The type of the JMS destination where the ECP JMS Monitoring messages will be sent. Use "temp" to indicate a temporary Destination and "administered" to indicate an administered one. Ignored if "ECP.Msgs.Enable=false". Its default value is "administered".

JMSMessageBroker.dest.name: The JMS destination where the ECP JMS Monitoring messages will be sent. If the destination type in "JMSMessageBroker.dest.type" is temporary, any value will suffice; if the destination type in "JMSMessageBroker.dest.type" is administered, this property must contain the name under which the Destination is registered. Ignored if "ECP.Msgs.Enable=false". Its default value is "ECP.MainTopic".

5. Monitoring

5.1. Logging

From an SA point of view, ECP Service monitoring is log files based. It is possible to programmatically monitoring de ECP though.

5.1.1. ECP Log Directory

During installation an ECP log Directory will be established. All ECP log files (except the ECP pid file) will be relative to that path. By default, that path is "`C:\hp\OpenView\ServiceActivator\var\log`" in windows and "`/var/opt/OV/ServiceActivator/log/`" in HP-UX

5.1.2. Log level

Currently, all ECP log levels are established programmatically or through DB Pool Configuration.

5.1.3. stdout and stderr log

The ECP start script will redirect the standard input and standard error of the ECP JVM to the files `RmiEcpService.stdout` and `RmiEcpService.sterr`, both located in the ECP Log Directory (See ECP Log Directory).

`stdout` and `stderr` will mainly contain log messages not referring or depending on a specific Pool, but on the ECP Service as a whole (start process, shutdown process, etc…). This may vary, as is not uncommon for the ECP Equipment Drivers to log to `stdout`.

5.1.4. Configuration load log

The ECP Configuration load (DB defined Pools instantiation and start during ECP Service startup) will be logged to the file `Configurator.log` located in the ECP Log Directory (See ECP Log Directory).

5.1.5. Pool log

Each Pool will have its own log file. The log file name is established programmatically, but it will be relative to the ECP Log Directory (See ECP Log Directory). The Pool log files will contain the log specifically related to that Pool.

5.1.6. Spy log

Spy log will provide a low level network log. Each Protocol Driver instance, that is, each Connection, may have its own log file. The log file name may be established programmatically, but it will be relative to the ECP Log Directory (See ECP Log Directory). By default, the spy file name will be

```
"TelnerDriverSpyStream_" + hostName + "_" + port + "_" + this.hashCode() +
".log"
```

5.1.7. ECP pid File

On HP-UX, the ECP startup script will leave the new ECP Service JVM process identifier in the path "`/var/opt/OV/ServiceActivator//tmp/ecp.pid`" by default. The value of the var dir (the "`/var/opt/OV/ServiceActivator/`" part may be defined during installation.

5.2. GUI

5.2.1. Pools

To access Pool Listing must select from the menu of Views "Administrator" - "ECP" - "Pool" - "List".

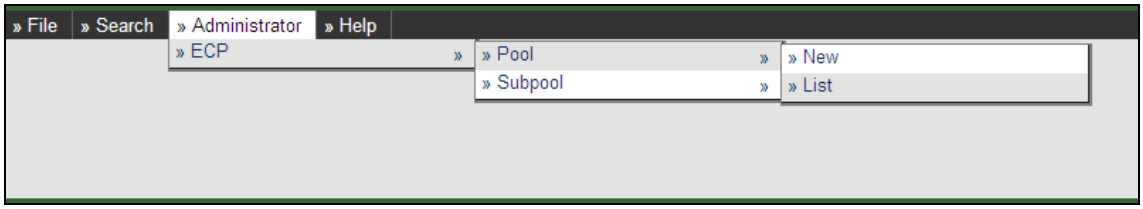


Fig. 3: View, Pool listing

When the Pool Listing is called we will see in the top of the screen two fields which are "Pool Name" and "Log File Name", and in the middle we will see all the pools listed.

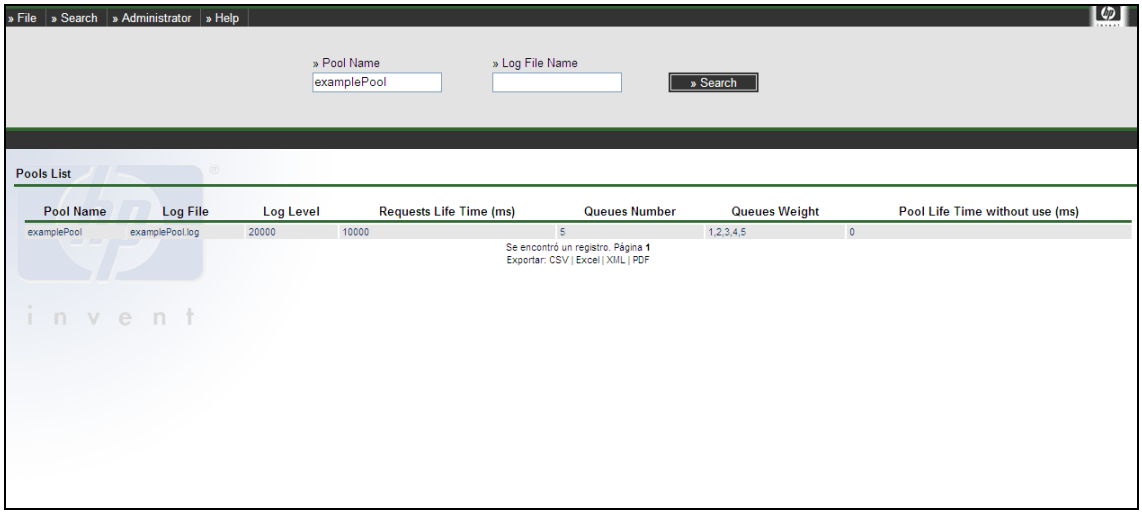


Fig. 4: Operation, Pool listing

We can list pools using one or two of those fields or we can click directly in the "search" button if we want to list all the existing pools.

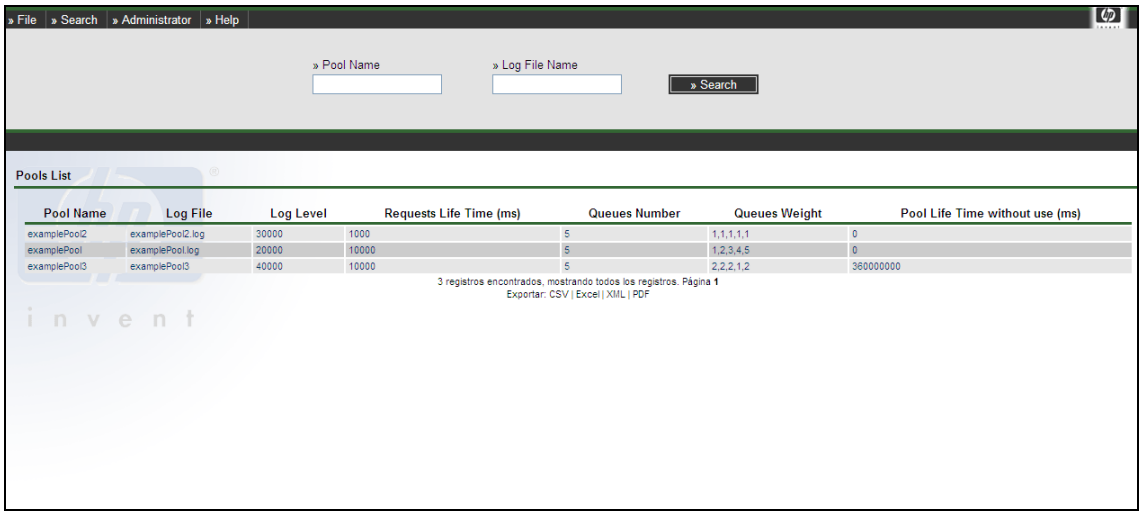


Fig. 5: Operation, Pool listing empty search

If any pool fulfils the search rules it will be shown in the list.

If no pool fulfils the search rules, the Ecp-web will show a screen with an error message.

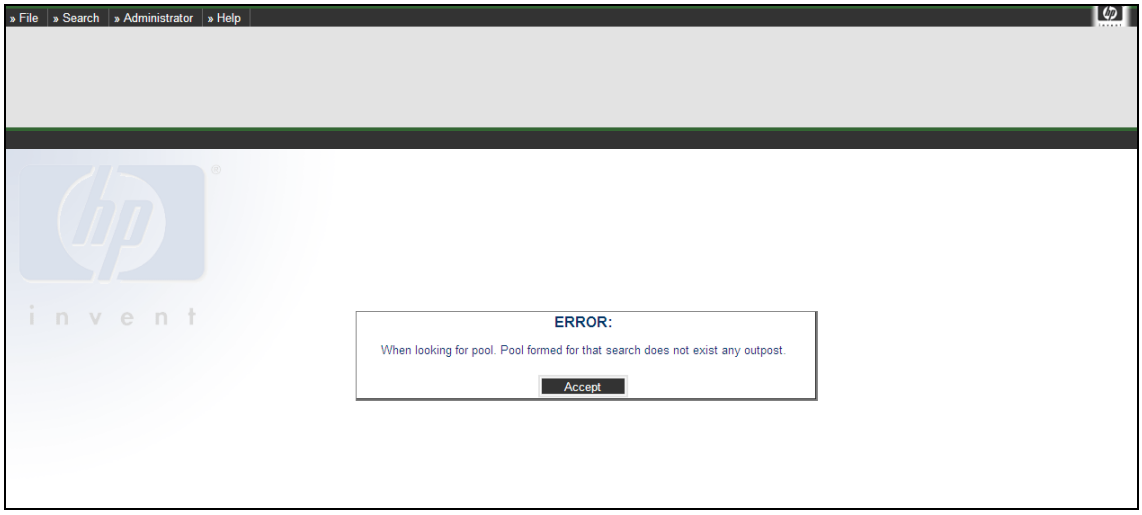


Fig. 6: Operation, Pool listing no result

5.2.2. SubPools

To access Subpool Listing must select from the menu of Views "Administrator" - "ECP" - "Subpool" - "List".

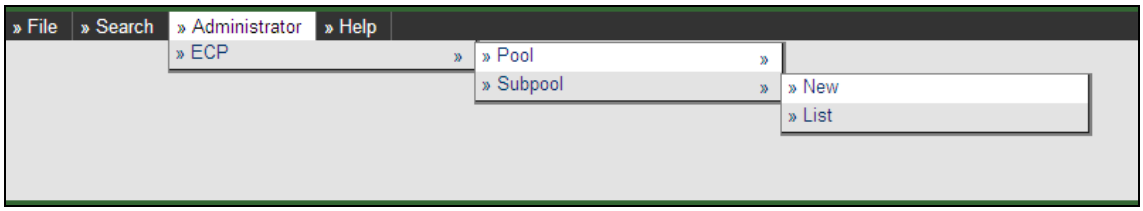


Fig. 37: View, Subpool listing

When the Subpool listing is called we will see in the top of the screen four fields which are "Pool Name", “Protocol”, “Ip address” and "Port", and in the middle we will see all the existing subpools listed.

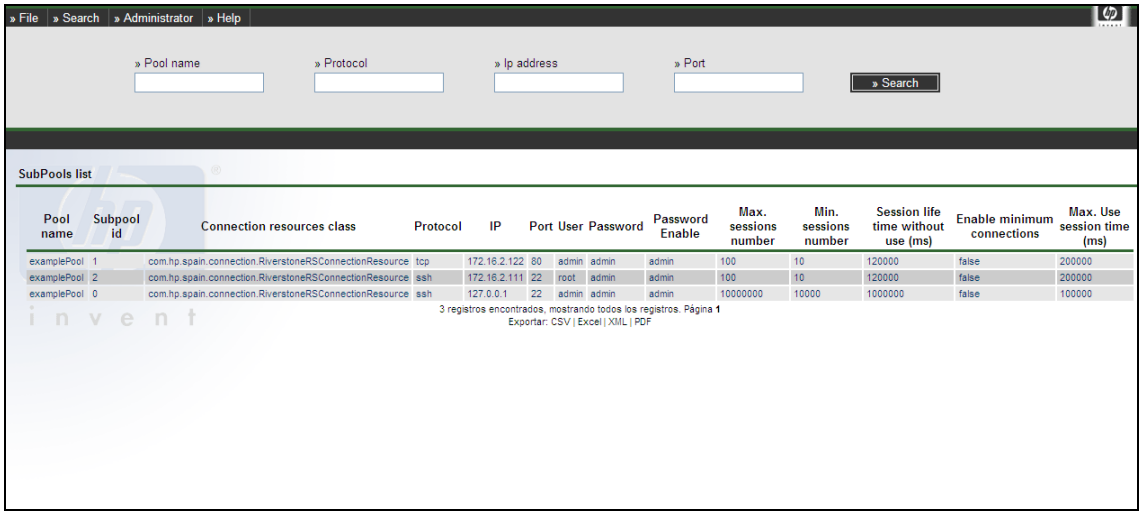


Fig. 38: Operation, Subpool listing

We can list subpools using any of those fields or we can click directly in the "search" button if we want to list all the existing subpools.

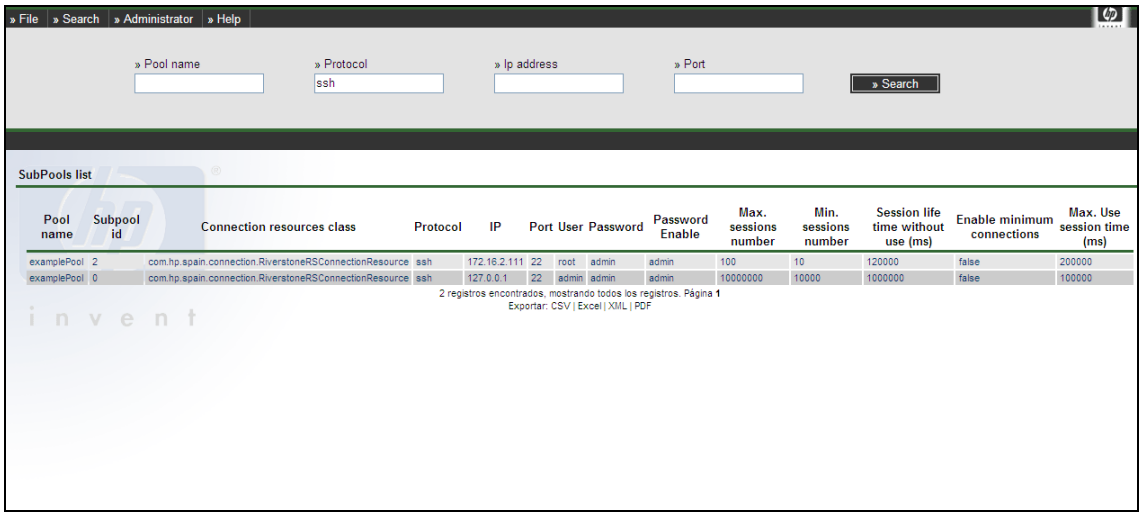


Fig. 39: Operation, Subpool listing success

If any subpool fulfils the search rules it will be showed in the list.

If no subpool fulfils the search rules, the Ecp-web will show a screen with an error message.

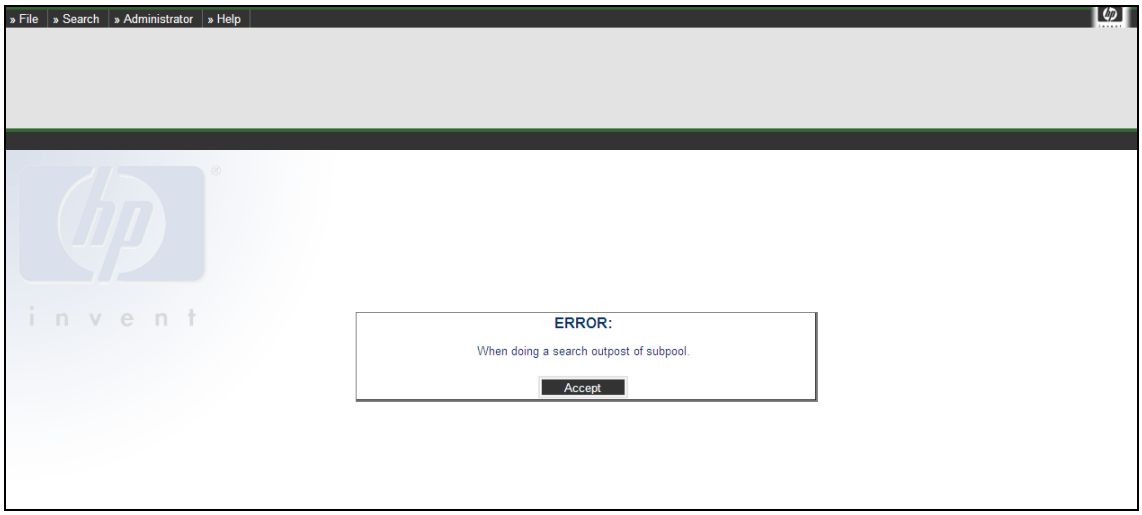


Fig. 40: Operation, Subpool listing no result

5.2.3. SubPool Connections

To list a subpool sessions you must select from the states menu the "List Sessions" option.

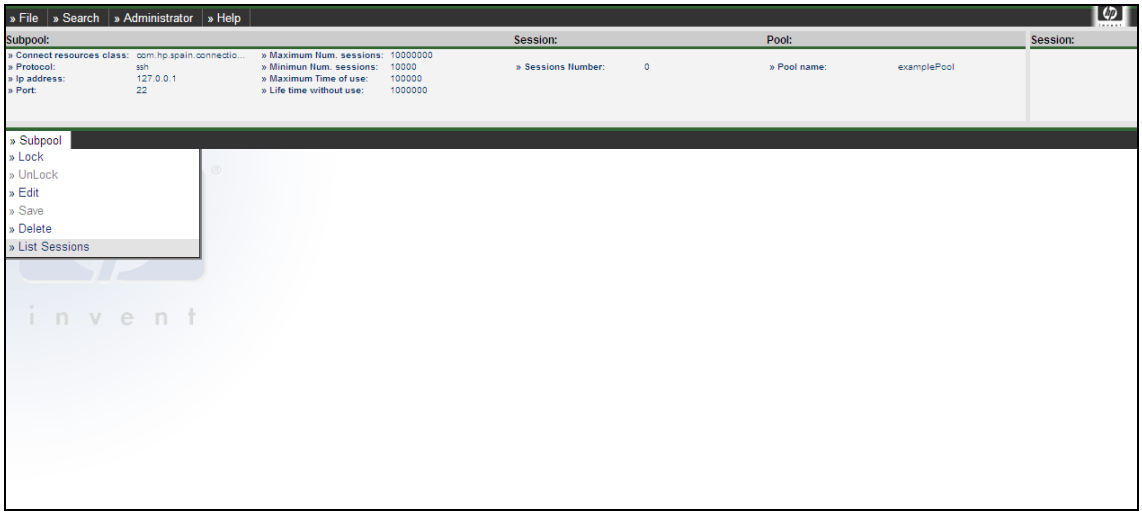


Fig. 52: Status, Subpool sessions list

When the Subpool sessions list action is called, if the subpool has any sessions, the Ecp-web will show us a screen with the information of those sessions. If there is no subpool session we will see the following screen.

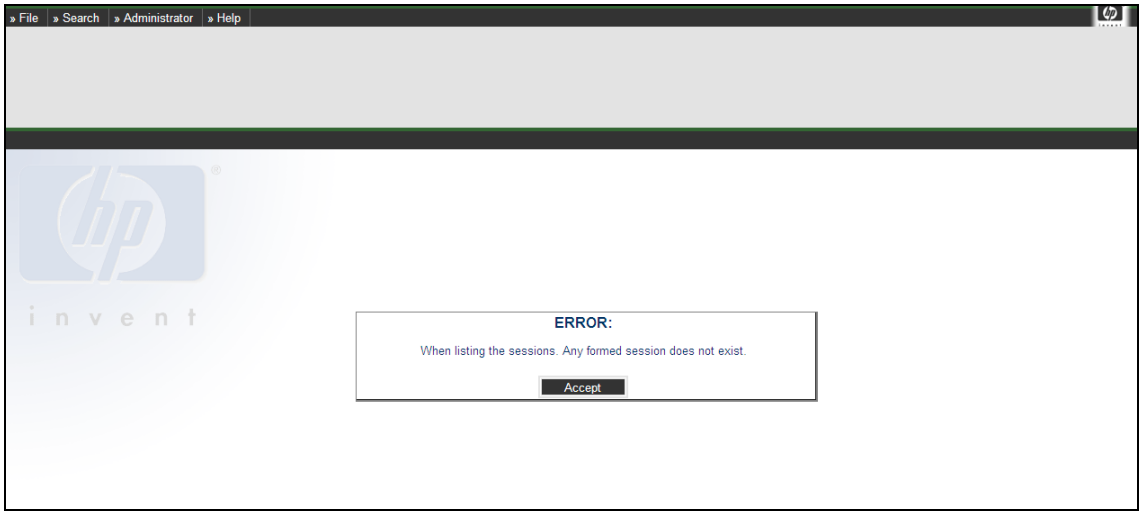
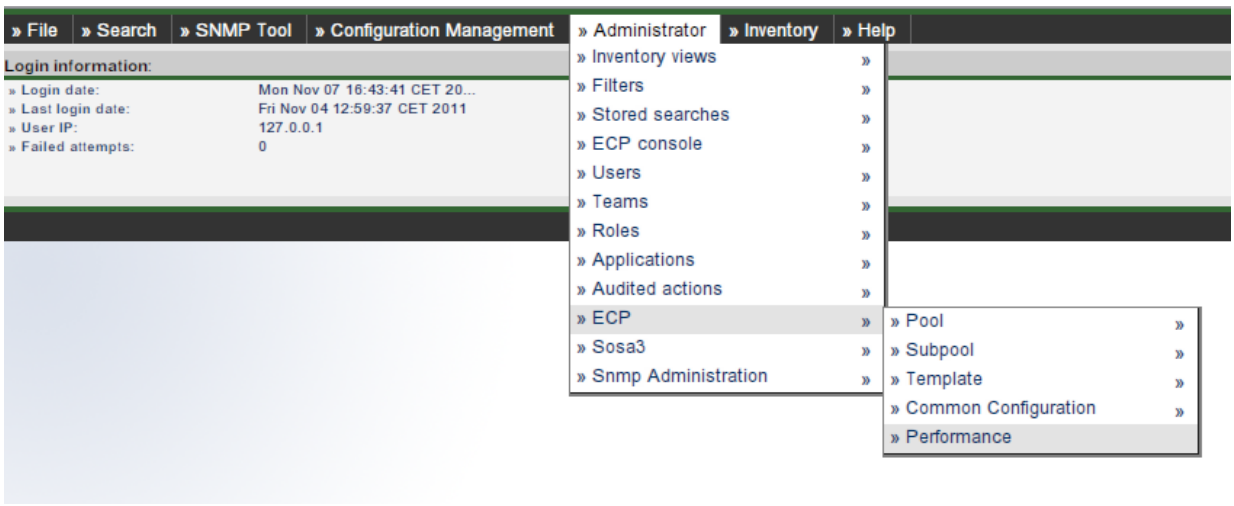


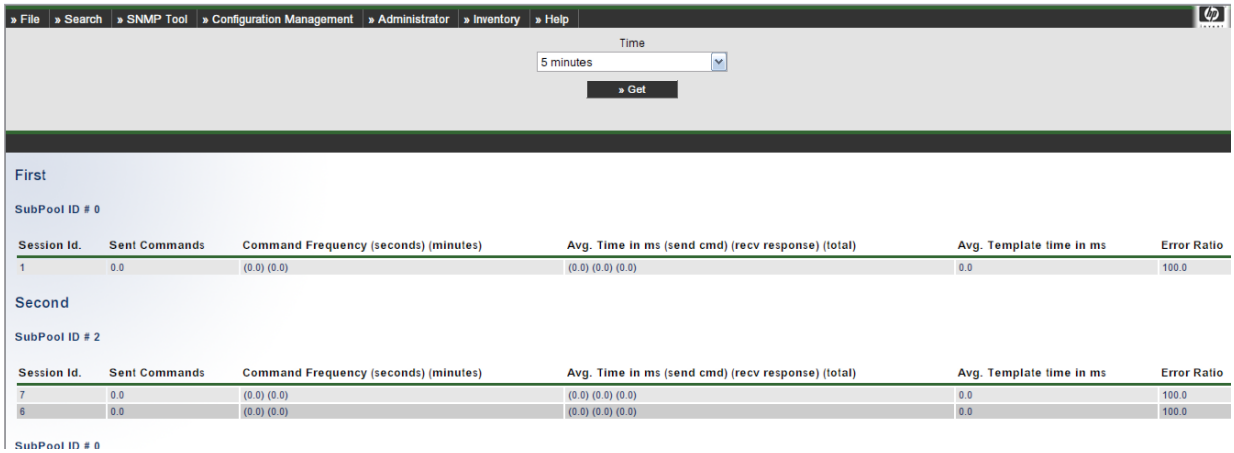
Fig. 53: Operation, Subpool sessions list error

5.2.4. Performance

In order to query performance statistical information, select "Administrator" - "ECP" - "Performance" in the view menu:



You will be presented with a performance summary like the following:



In order to change the period of time in which the displayed statistics are based, select a new period in the “Time” combobox and click on “Get”.

5.3. JMS

JMS messaging is intended for programmatic monitoring. Refer to the Equipment Connections Pools Developer Reference for more details.

Refer to JMS 1.1 Specification and API for further details on JMS concepts and use:
http://java.sun.com/products/jms/docs.html

Refer to Active MQ documentation for further detaisl on Active MQ configuration and functionality
http://activemq.apache.org/using-activemq.html

6. Permissions

6.1. File System

The ECP Service will need the following File System Permissions:

1. Read permission over the ECP home directory for the user executing de ECP Service JVM.
2. Write permission over the ECP Log Directory (See ECP Log Directory) for the user executing de ECP Service JVM.
3. Write permissions over the ECP pid file (See ECP pid File) for the user executing de ECP Service JVM.

6.2. Network

6.2.1. Outbound Connections

The ECP Service will always connect to the following connection endpoints:

1. Target Systems: The ECP Service will establish outbound connections for Commands Template execution. As a consequence, the ECP will need TCP access to all the managed Target Systems through
2. RMI Registry Service: The ECP Service will establish TCP outbound connections to the RMI Registry Service for the ECP RMI Service binding. As a consequence, it will need access to the RMI Registry Service host and port which are configured in the ECP startup command. See `<ecp_rmi_registry_server_host>` and `<ecp_rmi_registry_server_port>` in ECP RMI Service Command Line Parameters.
3. Data Base: The ECP Service will establish TCP outbound connections to the Data Base server. As a consequence, ECP will need access to the Data Base host and port which are configured in the `ecp.properties` file. See `DB_URL` in the ecp.properties.

6.2.2. Inbound Connections

The ECP Service will listen on the following connection endpoints:

1. RMI Registry Service: The ECP will try to start its own Registry Service in the local host and in the configured port. After, the ECP RMI Service will be registered on it (through an incoming connection). See `<ecp_rmi_registry_server_port>` in ECP RMI Service Command Line Parameters.
2. ECP RMI Service: The ECP will bind its service to the configured host and to the RMI Registry Service port +1. See `<ecp_rmi_server_ip>` and `<ecp_rmi_registry_server_port>` in ECP RMI Service Command Line Parameters. Whenever a client executes a template, an Incoming connection will be created to this socket.

7. Troubleshooting

7.1. Not Accessible ECP Service

7.1.1. Symptomps

When trying to connect to the ECP Service through Commands Template execution or status querying (see Checking ECP Service), one of the following exceptions is thrown:

```
java.rmi.ConnectException: Connection refused to host: xx.xx.xx.xx; nested
exception is:
        java.net.ConnectException: Connection refused (errno:239)
        at sun.rmi.transport.tcp.TCPEndpoint.newSocket(TCPEndpoint.java:567)
        at
sun.rmi.transport.tcp.TCPChannel.createConnection(TCPChannel.java:185)
        at
sun.rmi.transport.tcp.TCPChannel.newConnection(TCPChannel.java:171)
        at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:101)
        at
com.hp.spain.connection.pool.server.RmiEcpService_Stub.init(Unknown Source)
```

7.1.2. Causes and Solutions

7.1.2.1. The ECP Service is not started

Check whether the ECP Service is started (see Identifying the ECP Service Process). If no ECP Service process is found, The ECP Service has to be started to fix the problem (see Starting ECP Service).

7.1.2.2. The ECP Service or Client is miss-configured.

Check whether the client is connecting to the valid RMI Registry and ECP Service IPs and ports (See `<ecp_rmi_server_ip>`, `<ecp_rmi_registry_server_host>` and `<ecp_rmi_registry_server_port>` in ECP RMI Service Command Line Parameters)

Make sure that the IPs are exactly the same, because the RMI Registry and ECP Service will only listen on the specified interfaces. That is, if your RMI Registry or ECP Service are listening on the loopback IP (127.0.0.1) but your client is trying to connect to other host IP the client won't be able to connect. If the IPs or ports do not match, configure them correctly.

7.1.2.3. A Firewall is Blocking Access to the ECP

If the ECP Service is started (see The ECP Service is not started) and both client and server are correctly configured (see The ECP Service or Client is miss-configured.), make sure no firewall is blocking the Access to the ECP.

7.2. Equipment Driver Class Not Found

7.2.1. Symptomps

When trying to create a connection to the Target System, the following exception is thrown:

```
java.lang.ClassNotFoundException: com.hp.spain.connection.xxxxxxx
        at java.net.URLClassLoader$1.run(URLClassLoader.java:199)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(URLClassLoader.java:187)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:289)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:274)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:235)
        at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:302)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:141)
```

7.2.2. Causes and Solutions

The Equipment Driver class `com.hp.spain.connection.xxxxxxx` must be present in the classpath at startup time. The ECP startup script will include all the jar files found in the `$ecp_home\lib` directory in the classpath. The Equipment Driver classes and their dependencies must be there at startup and the ECP restarted to include the new classes (see Restarting ECP Service). Copy there all the needed files and restart the ECP Service.

7.3. Target System not Accessible

7.3.1. Symptomps

When trying to create a connection to the Target System, the following exception is thrown:

```
java.io.IOException: Connection timed out (errno:238) (Host: xx.xx.xx.xx;
Port: xx)
        at com.hp.spain.connection.TcpDriver.connect(TcpDriver.java:83)
        at
com.hp.spain.connection.EquipmentDriver.connectServer(EquipmentDriver.java:8
38)
```

7.3.2. Causes and Solutions

7.3.2.1. The SubPool is miss-configured.

Check whether the SubPool is connecting to the valid Target System IP and Port. If the IPs or ports do not match, configure them correctly.

7.3.2.2. A Firewall is Blocking Access to the Target System

If the SubPool is correctly configured, make sure no firewall is blocking the Access to the ECP.