HPSA Extension Pack

Lock Manager

Release V6.0

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 trademark of Oracle and/or its affiliates.

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

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

Oracle® is a trademark of Oracle and/or its affiliates.

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.

Document id: pd001501

Table of Contents

In This Guide

This guide is meant as a user reference guide for the Lock Manager's latest version. It contains all the information about this tool, its features and how to use them.

Audience

The audience for this guide is the Solutions Integrator (SI). The SI 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

Is able to program in Java™ and XML

Understands security issues

Understands the customer's problem domain

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: `InventoryBuilder` *<sourceFiles>*
	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 `InventoryBuilder` 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.
	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 (->), a cascading menu follows	Select Locate:Objects->by Comment.

Install Location Descriptors

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

Descriptor	What the Descriptor Represents
$ACTIVATOR_OPT	The install base 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\
$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/standalone/deployments The Windows location is <drive>:\HP\jboss\stanalone\deployments
$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 install base location of SOSA. The default UNIX location is

	/opt/OV/ServiceActivator/EP/SOSA The default Windows location is <drive>:\HP\OpenView\ServiceActivator\EP\SOSA
$SOSA_BIN	The install location of specific SOSA binary files. The default UNIX location is /opt/OV/ServiceActivator/EP/SOSA/bin The default Windows location is <drive>:\HP\OpenView\ServiceActivator\EP\SOSA\bin\
$SOSA_ETC	The install location of specific SOSA configuration files. The default UNIX location is /opt/OV/ServiceActivator/EP/SOSA/conf The default Windows location is <drive>:\HP\OpenView\ServiceActivator\EP\SOSA\conf\
$ECP_HOME	The install base location of Equipment Connections Pool. The default UNIX location is /opt/OV/ServiceActivator/EP/ECP The default Windows location is <drive>:\HP\OpenView\ServiceActivator\EP\ECP\
$ECP_BIN	The install location of specific Equipment Connections Pool binary files. The default UNIX location is /opt/OV/ServiceActivator/EP/ECP/bin The default Windows location is <drive>:\HP\OpenView\ServiceActivator\EP\ECP\bin\
$ECP_ETC	The install location of specific Equipment Connections Pool configuration files. The default UNIX location is /opt/OV/ServiceActivator/EP/ECP/conf The default Windows location is <drive>:\HP\OpenView\ServiceActivator\EP\ECP\conf\

1 Introduction

1.1 Purpose

This guide is meant as a developer and administrator reference guide for the Lock Manager. It contains all the information about this tool, its features and how to use them.

1.2 Document Scope

This document is focused on the main features oriented for developers.

1.3 Definitions

1.3.1 Acronims

MWFM: Micro Work Flow Manager

HPSA: HP Service Activator

EP: Extension Pack

SC: Solution Container

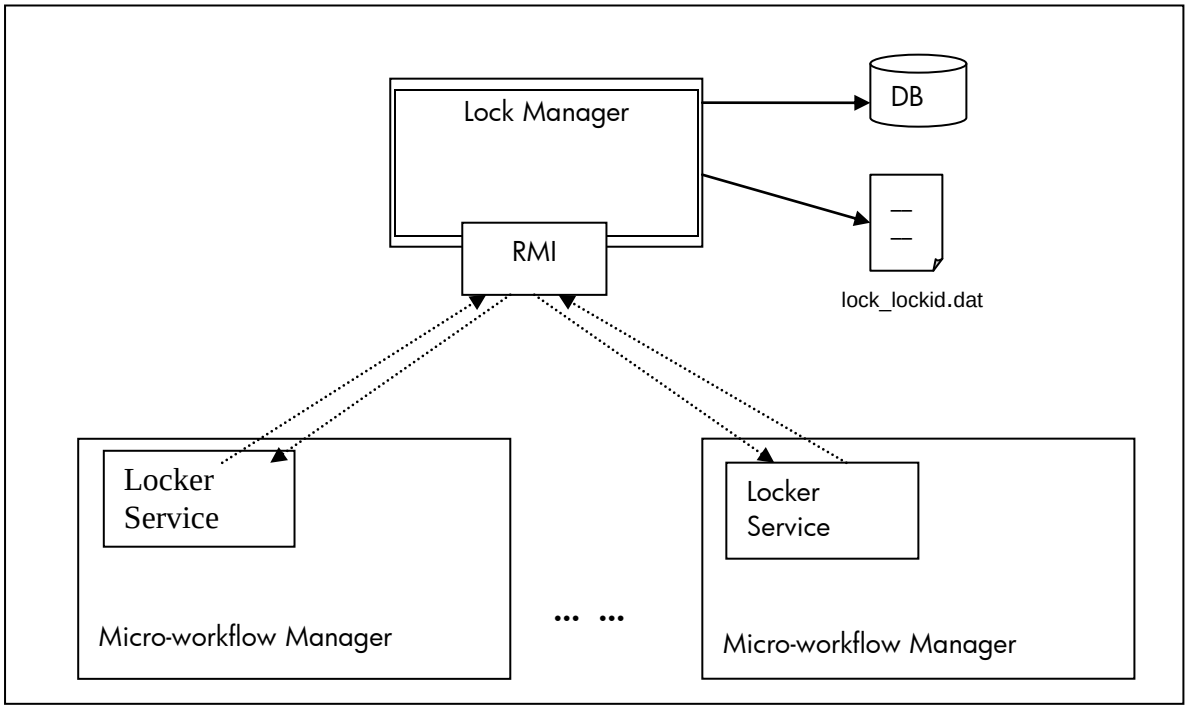
LM: Lock Manager

2 General description

Lock Manager is the service in charge of the distributed management of the active locks in the HPSA. This allows locking common objects from several systems, as they are all being managed by one and the same individual system.

It provides both a web interface and a client shell for the management of the different service elements.

3 Architecture



Next the different elements involved are described:

Lock Manager: Also termed administrator service as it is in charge of the management of the locking requests arriving from several machines. It is installed in one machine only, and possesses an RMI interface that provides methods for locking, unlocking, and notifying about those locks…

This service allows two ways to manage the persistence of its locks, either saving the locking information on a file or on a DB.

Locker service: Also called locking service, it is in charge of the bidirectional communication between the Micro-workflow Manager (MWFM onwards) of the machine where it is installed and the administrator service (Lock Manager). To define a locker service we must configure a *LockModule* module in the mwfm.xml file, that is, in the configuration file of the MWFM where the locking service will reside. This module provides methods for requesting a Lock Manager for single and multiple locks from the MWFM nodes.

When launching the MWFM, the Locker Service contained in the module will connect to the service and will be registered with the URL and name defined in the module. When shutting down the MWFM it will be unregistered.

4 Lock Manager Service Functionality

The Lock Manager provides a distributed locking system, to provide single access to the objects being locked. This is necessary as it is meant to be used in distributed concurrent applications, therefore the possibility of locking a particular object is a necessity for the HPSA.

The Lock Manager comes with several nodes and classes which enable using the Lock Manager from Workflows and also managing the Lock Manager from the HPSA.

5 Lock Manager Node Guide

The nodes that implement the functionality provided by the Lock Manager are contained in the Workflow Transactions project. Most of them receive a transaction as an argument. Consult the document *EP - Developer reference* (Workflow Transaction Module) to see how to manage a transaction.

We can differentiate between:

Nodes with Lock Manager's own functionality such as locking, unlocking, assigning a lock.

Node Name	Description
LockInventory	Locks beans in the inventory. If the lock is not available the request is queued.
LockInventoryWithoutEnqueue	Locks beans in the inventory. If the lock is not available the node returns an error.
UnlockInventory	Unlocks the beans in the inventory.
AssignLockId	Sets the current job as owner of a lock.

Nodes affected by the Lock Manager such as updating the inventory, moving beans to history tables, etc. These kinds of nodes consult the administrator service to check whether the bean used is or not locked.

Node Name	Description
InsertInventory	Inserts beans in the inventory.
UpdateInventory	Updates beans in the inventory.
MoveToHistory	Moves beans from the inventory to the history tables
TestAndSet	Evaluates a condition and if it is fulfilled an action is executed on a bean from the inventory.
WFTransactionHandler	Generic Handler that liberates resources reserved by the flow and synchronizes with the parent flow if the execution has been cancelled, propagating the corresponding error message.

All the nodes return the variable *RET_VALUE*. It is a String with value 0 if the node has been successfully performed, -1 if this is not the case.

A Locker service is the service in charge of the communication with the Lock Manager. One must exist in each machine where a MWFM resides and it is required that the generated locks are administered by the administrator service.

To define a Locker Service we must configure in the mwfm.xml configuration file ($ETC/config) a *LockModule* module. This module will contain the locker.

5.1 Nodes with Lock Manager's own functionality

All the nodes described in this section have a common entry parameter called *lock_id*. It is the multilocking identifier. Internally this identifier refers to a vector that stores all the generated locks. Each lock will be added to the vector. In such a way that, a single identifier can reference all the required locks.

For the locker installation we must install the lock-manager project in every machine where a locker is going to be defined. Also, if we want to use the methods to ask for locks, unlocks etc., from a Lock Manager from the MWFM nodes we must install the ovsa41-wf-transaction project.

5.1.1 Lock Inventory

Node that locks beans in the inventory and adds the corresponding rollback operations in a Workflow Transaction*. If the lock isn't available it queues the request.

* A Workflow Transaction is a flow level transaction, in which operations can be inserted or removed as needed. Each of the different flow nodes or of the child flows can insert their own operations. This transaction is processed in the end handler of the flow that created it.

Parameter Name	Mandatory	Type	Description
WF_TRANSACTION_NAME	Si	String	This name identifies each of the transactions of a group inside the flow hierarchy
bean + i	Si	String	Names of the beans to be locked
primary_key + i	Si	String	Primary keys
array	No	Object	Bean array to be locked
string_array	No	String	Sets its value to YES or TRUE if the array parameter contains a bean array
job_id	Si	String	Variable from the case-packet that contains the JOB_ID. If none is specified its default value will be JOB_ID
lock_id	No	String	Variable from the case-packet that stores the multilock identifier. If none is set the lock_id is used as default.
lock_module	No	String	Name for the module that will be used as locker for this flow, to make the lock. If none is specified the default value used will be (LockModule)

In the previous table we can see that there are two ways of locking a group of beans at the same time in the same node:

- Using the '**bean+i**' and '**primary_key+i**' variables to indicate the different beans that must be locked and their primary keys.

```
<Process-Node>
  <Name>Locks RiverstoneRS</Name>
  <Action>
    <Class-Name>com.hp.spain.node.wftransaction.LockInventory</Class-Name>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction_lock"/>
    <Param name="bean0" value="com.hp.spain.inventory.RiverstoneRS"/>
    <Param name="primary_key0" value="rsId"/>
```

```
    <Param name="bean1" value="com.hp.spain.inventory.RiverstoneRSPort"/>
    <Param name="primary_key1" value="rsPortId"/>
    <Param name="job_id" value="JOB_ID"/>
    <Param name="lock_id" value="lock_id"/>
  </Action>
  <Next-Node>...</Next-Node>
</Process-Node>
```

- Using the variables *array* and *string_array*. *string_array* must be *yes* or *true*, to indicate that a bean array is going to be sent and *array* is the bean array to be locked. If the *string_array* variable is set to *false*, the array variable won't be evaluated, assuming that no bean array exists. By default this variable is set to *false*.

```
<Process-Node>
  <Name>Lock RiverstoneRS</Name>
  <Action>
    <Class-Name>com.hp.spain.node.wftransaction.LockInventory</Class-Name>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction_lock"/>
    <Param name="string_array"  value="TRUE"/>
    <Param name="array" value=" arrayRiverstoneRS "/>
    <Param name="job_id" value="JOB_ID"/>
    <Param name="lock_id" value="lock_id"/>
  </Action>
  <Next-Node>...</Next-Node>
</Process-Node>
```

The *string_array* parameter has assigned the value 'TRUE' whose value must be either true or yes. The array parameter has assigned the variable *arrayRiverstoneRS* which contains the bean array.

In both cases if the *lock_id* parameter is passed empty to the node, a vector with all the generated locks in the current node will be produced. The *lock_id* generated will reference the locks.

If the opposite happens, that is, the *lock_id* passed to the node already contains the previous locks; the current node will update the vector, adding the new locks. The *lock_id* returned will be the same that was passed to the node; however the vector being reference will contain both the old and the new locks.

5.1.2 LockInventoryWithoutEnqueue

The operation of this node is similar to the previous, the difference being that in this case the node returns an error. The entry parameters are the same as in the previous example.

Parameter name	Mandatory	Type	Description
WF_TRANSACTION_NAME	Si	String	This name will identify each of the group transactions inside a flow hierarchy
bean + i	Si	String	Names of the beans to be locked
primary_key + i	Si	String	Primary keys
Array	No	String	Bean array to be locked
string_array	No	String	Set its value to YES or TRUE if the array parameter contains a bean array
job_id	Si	String	Variable from the case-packet which contains the JOB_ID. If it isn't specified

			the default value will be JOB_ID
lock_id	No	String	Variable from the case-packet which stores the lock identifier. If none is set the default value will be the lock_id.
lock_module	No	String	Name of the module that will be used as locker to achieve the lock. If none is specified a default value will be used (LockModule)

```
<Process-Node>
  <Name>Bloquear Equipo</Name>
  <Action>
    <Class-Name>
      com.hp.spain.node.wftransaction.LockInventoryWithoutEnqueue
    </Class-Name>
    <Param name="bean0" value="com.hp.spain.inventory.RiverstoneRS"/>
    <Param name="primary_key0" value="constant:1"/>
    <Param name="job_id" value="JOB_ID"/>
    <Param name="lock_id" value="lock_id_rs"/>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
  </Action>
  <Next-Node>...</Next-Node>
</Process-Node>
```

5.1.3 UnlockInventory

Unlocks the beans in the inventory and executes the delete and update operations previously added to the Workflow Transaction* by the nodes DelayedDelete** and DelayedUpdate**

** The DelayedDelete and DelayedUpdate nodes add the update and delete operations of the inventory beans, becoming active during the execution of the End Handler.

The *lock_id* parameter contains the lock identifier. In order to unlock a bean, the identifier that was passed or was generated in the lock's creation must be used.

Parameter name	Mandatory	Type	Description
WF_TRANSACTION_NAME	Si	String	Este nombre identificará cada una de las transacciones de un grupo dentro de una jerarquía de flujos
Lock_id	No	String	Variable del case-packet que almacena el identificador de bloqueo. Si no se establece se usará lock_id como valor por defecto.
Lock_module	No	String	Nombre del módulo que se usará como locker en este flujo, para realizar el bloqueo. Si no se especifica uno e utilizará su valor por defecto (LockModule)

```
<Process-Node>
  <Name>Desbloquear Equipo</Name>
  <Action>
    <Class-Name>com.hp.spain.node.wftransaction.UnlockInventory</Class-Name>
    <Param name="lock_module" value="MWFM-0"/>
    <Param name="lock_id" value="lock_id_rs"/>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
  </Action>
  <Next-Node>NextNode</Next-Node>
</Process-Node>
```

5.1.4 AssignLockId

Node that establishes the current job as owner of a lock and adds it to a Workflow Transaction.

Parameter Name	Mandatory	Type	Description
WF_TRANSACTION_NAME	Si	String	This name will identify each of the transactions belonging to a group inside a flow hierarchy
Value	Si	String	Variable where the lock identifier is kept.
lock_id	Si	String	Variable where the new lock identifier will be kept.
Soft	No	String	No exceptions are thrown if its value is set to YES or TRUE

```
<Process-Node>
  <Name>Assign Lock Id</Name>
  <Action>
    <Class-Name>com.hp.spain.node.wftransaction.AssignLockId</Class-Name>
    <Param name="value" value="lock_id_previous"/>
    <Param name="lock_id" value="lock_id_rs"/>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
  </Action>
  <Next-Node>NextNode</Next-Node>
</Process-Node>
```

5.2 Nodes affected by the Lock Manager

5.2.1 InsertInventory

Node that inserts beans in the inventory, adding afterwards the corresponding rollback operations to a Workflow Transaction.

Parameter Name	Mandatory	Type	Description
WF_TRANSACTION_NAME	Si	String	This name will identify each of the transactions belonging to a group

			inside a flow hierarchy.
Db	No	String	Name of the DB module to be used. If this parameter is not passed the default value of "db" will be loaded.
Bean	Si	String	Bean name
primary_key	Si	String	Primary Key
attribute + i	Si	String	Attributes to be inserted in the bean
value + i	No	String	Values for the attributes to be inserted in the bean
field + i	No	String	Fields to be updated. They must be in lower case.
variable + i	No	String	Values of the fields to be updated.
bean_object	No	String	Bean generated with the data inserted in the inventory
lock_module	No	String	Name of the locking module to be used.
lock	No	String	The values of YES or TRUE will indicate that the new bean is will be locked.
job_id	No	String	Variable from the case-packet that contains the JOB_ID. If none is specified the default value will be JOB_ID
lock_id	No	String	Variable from the case-packet which stores the lock identifier (optional). If none is speficied the default value used will be lock_id.
soft	No	String	No exceptions are thrown if its value is set to YES or TRUE.
method	No	String	Defines the method to be executed to make the insertion in the inventory. If none is specified the bean's store method is executed.
attributeExt + i	No	String	Extended attributes to be inserted
valueExt + i	No	String	Values for the extended attributes
error_message	No	String	Error message generated when the node is executed

```
<Process-Node>
  <Name>InsertInventory</Name>
  <Action>
```

```
    <Class-Name>com.hp.spain.node.wftransaction.InsertInventory</Class-Name>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
    <Param name="db" value="db"/>
    <Param name="bean" value="com.hp.spain.inventory.WfBDTest"/>
    <Param name="attribute0" value="WorkflowId"/>
    <Param name="attribute1" value="Name"/>
    <Param name="attribute2" value="Description"/>
    <Param name="value0" value="JOB_ID"/>
    <Param name="value1" value="MyName"/>
    <Param name="value2" value="MyDescription"/>
  </Action>
  <Next-Node>...</Next-Node>
</Process-Node>
```

5.2.2 UpdateInventory

Updates beans in the inventory, adding afterwards the corresponding rollback operations in a Workflow Transaction.

Parameter Name	Mandatory	Type	Description
WF_TRANSACTION_NAME	Si	String	This name will identify each of the transactions in a group inside a flow hierarchy
Db	No	String	Name of the DB module to be used. If this parameter is not passed the default value loaded will be “db”
Bean	Si	String	Bean name
primary_key	Si	String	Primary Key
attribute + i	No	String	Attributes to be updated in the bean
value + i	No	String	Attribute values to be updated in the bean
variable + i	No	String	Field values to be updated
bean_object	No	String	Bean generated with the data inserted in the inventory
job_id	No	String	Variable from the case-packet that contains the JOB_ID. If none is specified the default value is JOB_ID
lock_id	No	String	Variable from the case-packet which stores the lock identifier (optional). If none is specified the default value used will be the lock_id.
ignoreLockMan	No	String	Variable that indicates whether we must take into account the locking module. If its value is false, no validation to check that the bean is locked is made before an update.

lock_module	No	String	Name for the module that will be used as locker for this flow, to make the lock. If none is specified the default value is used (LockModule)
Soft	No	String	No exceptions are thrown if its value is set to YES or TRUE.
Findmethod	No	String	Search method implemented to obtain the bean to be updated
Findvariable + i	No	String	Entry variables for the method specified by the findmethod parameter.
Updatemethod	No	String	Method to be executed to make the update. If none is set the default update method used will be the bean's update.
updatePK	No	String	If its value is true or yes it means that the method specified with the updatemethod parameter will be used.
primary_key_att + i	No	String	primaryKeys of the beans to be updated if the update method is used.
attributeExt + i	No	String	Extended attributes to be inserted
valueExt + i	No	String	Values for the extended attributes
error_message	No	String	Error message generated when the node is executed.

```
<Process-Node>
  <Name>UpdateRegistry</Name>
  <Description/>
  <Action>
    <Class-Name>com.hp.spain.node.wftransaction.UpdateInventory</Class-Name>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
    <Param name="db" value="db"/>
    <Param name="bean" value="com.hp.spain.inventory.WfBDTest"/>
    <Param name="primary_key" value="Resultado"/>
    <Param name="attribute0" value="Description"/>
    <Param name="value0" value="Descripcion"/>
  </Action>
  <Next-Node>...</Next-Node>
</Process-Node>
```

5.2.3 MoveToHistory

Moves beans from the inventory to the history tables, adding afterwards the corresponding rollback operations in a Workflow Transaction.

Parameter Name	Mandatory	Type	Description

WF_TRANSACTION_NAME	Si	String	This name identifies each of the transactions of a group inside a flow hierarchy.
Db	No	String	Name of the DB module to be used. If this parameter is not passed the default value "db" will be used
bean + i	Si	String	Bean name
array_bean + i	No	Object	Bean array
primary_key + i	Si	String	Primary Key
bean_class + i		String	
lock_id			Variable from the case-packet that stores the lock identifier (optional). If none is specified the default value will be the lock_id
job_id	No	String	Variable from the case-packet which contains the JOB_ID. If none is specified the default value used will be JOB_ID
Soft	No	String	No exceptions are thrown if its value is set to YES or TRUE

```
<Process-Node>
  <Name>MoveToHistory</Name>
  <Description/>
  <Action>
    <Class-Name>com.hp.spain.node.wftransaction.MoveToHistory</Class-Name>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
    <Param name="db" value="db"/>
    <Param name="bean" value="com.hp.spain.inventory.WfBDTest"/>
    <Param name="primary_key" value="Resultado"/>
    <Param name="attribute0" value="Description"/>
    <Param name="value0" value="Descripcion"/>
  </Action>
  <Next-Node>MensajeModificar</Next-Node>
</Process-Node>
```

5.2.4 TestAndSet

Node that evaluates a condition and if it is fulfilled it executes an action on a bean from the inventory, adding afterwards the corresponding rollback operations to a Workflow Transaction*

Parameter Name	Mandatory	Type	Description
WF_TRANSACTION_NAME	Si	String	This name identifies each of the transactions of a group inside a flow hierarchy.

Db	No	String	Name of the DB module to be used. If this parameter is not passed the default value "db" will be used
Bean	Si	String	Bean name
primary_key	Si	String	Primary Key
Condition	No	String	Condition to be evaluated
action+i	Si	String	Action to be executed in a field. It must have a specific format*
field + i	No	String	Name of the required field
variable + i	No	String	Value of the required field
force_insert	No	String	If true, the insertion of the bean in the DB will be made, unless it already existed.
lock_id	No	String	Variable from the case-packet that stores the lock identifier (optional). If none is specified the default value will be the lock_id
job_id	No	String	Variable from the case-packet which contains the JOB_ID. If none is specified the default value used will be JOB_ID
Soft	No	String	No exceptions are thrown if its value is set to YES or TRUE
error_message	No	String	Error message generated when the node is executed

The action field format must be '**field;operator;value**' where operator can have the following values: '==', '*i=*', '<',' >', '<=', '>=' .

```
<Process-Node>
  <Name>TestAndSet</Name>
  <Description/>
  <Action>
    <Class-Name>com.hp.spain.node.wftransaction.MoveToHistory</Class-Name>
    <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
    <Param name="db" value="db"/>
    <Param name="bean" value="com.hp.spain.inventory.WfBDTest"/>
    <Param name="primary_key" value="pk"/>
    <Param name="condition" value="condition"/>
    <Param name="action0" value="Accion"/>
  </Action>
  <Next-Node>...</Next-Node>
</Process-Node>
```

5.2.5 WFTransaction Handler

Generic Handler that is always called at the end of a flow. It is used to liberate resources reserved by the flow, to synchronize with the parent flow and if the execution was cancelled, it propagates the corresponding error message. Also, it removes the Workflow Transactions created by the flow after processing the ones indicated in its configuration parameters.

The Finish and Cancel variables are the ones defined if the flow finished successfully or with errors. In a successful execution of the flow the value of these variables would be Finish=true and Cancel=false.

In the following table we specify what values must the variables finish and cancel have to take into account the rest of the parameter values.

For example:

If the rollback beans variable is set to true, it indicates that a rollback must be executed of the inserted or modified beans in the inventory. This action will only be executed if the values for the variables are Finish=no and Cancel=true.

Parameter name	Mandatory	Description	Finish/Cancel
finish	No	Boolean that indicates whether the flow finished successfully	
cancel	No	Boolean that indicates whether the flow has been cancelled	
controller	No	Boolean that tells whether the flow is a controller	-/-
WF_TRANSACTION_NAME, WF_TRANSACTION_NAME + i	No	Name of every Workflow Transaction to be processed	-/-
rollback_beans, rollback_beans + i	No	Boolean that indicates whether a rollback of the inserted or modified beans in the inventory should be executed	No/Yes
rollback_moved_beans, rollback_moved_beans + i	No	Boolean that indicates whether a rollback of the beans moved to the history table should be executed	No/Yes
rollback_recovered_beans, rollback_recovered_beans+i	No	Boolean that indicates whether a rollback of the beans that have been recovered from the history table should be executed.	No/Yes
remove_history_beans, remove_history_beans+i	No	Boolean that indicates whether the beans that have been moved to the history should be deleted	Yes/No
unlock_beans, unlock_beans+i	No	Boolean that indicates whether the beans that have been locked by the flow should be unlocked	Yes/Yes
release_resources, release_resources+i	No	Boolean that indicates whether the resources that have been reserved by the flow should be released	No/Yes
concurrent_sync	No	Boolean that indicates whether it must synchronize	No/Yes

		with the parent flow. This synchronization is used when the child flows have been launched by the StartJobConcurrent.	
concurrent_sync_finish	No	Boolean that indicates whether it must synchronize with the parent flow. This synchronization is used when the child flows have been launched by the StartJobConcurrent	Yes/No
delete_concurrent_cp	No	Boolean that indicates whether all the concurrent flow's information from the module's ConcurrentWorkflows queue should be deleted	Yes/Yes
remote_ip	No	Remote IP (necessary for synchronizing with the parent when concurrent flows are launched remotely)	-/-
local_ip	No	Local IP (necessary to synchronize with the parent flow when concurrent flows have been launched locally)	-/-
service_update, service_update+i	No	Boolean that indicates whether the enties in the ServiceInstanceParameters must be updated	Yes/No
delete_delayed, delete_delayed+i	No	Boolean that indicates whether the beans marked by the DelayedDelete node	Yes/No
update_delayed, update_delayed+i	No	Boolean that indicates whether the beans marked with the delay modification should be modified	Yes/No
release_delayed, release_delayed+i	No	Boolean that indicates whether the beans marked with delayed release should be released.	Yes/No
sender_module	No	Name for the module of message dispatch.	No/Yes
force_rollback, force_rollback+i	No	Boolean that indicates that the rollback tasks will be executed even if the flow ends successfully, ignoring the 'finish' parameter	-/-
action_if_killed, action_if_killed+i	No	Action to be executed on the bean if the job is killed using killJob	-/-

do_action_if_killed, do_action_if_killed+i	No	Boolean that indicates whether the associated task should be executed (even if the flow hast been killed)	-/-
db_if_killed, db_if_killed+i	No	Database module to be used	-/-
class_bean_if_killed, class_bean_if_killed+i	No	Bean class on which we want to execute the action	-/-
bean_if_killed, bean_if_killed+i	No	Variable with the bean instance on which we want to execute the action	-/-
field_if_killed, field_if_killed+i	No	Name of the bean's attribute	-/-
value_if_killed, value_if_killed+i	No	Value for the bean's attribute	-/-
lock_id_if_killed, lock_id_if_killed+i	No	Identifier for the bean lock on which the action will be executed	-/-

```
<End-Handler>
  <Class-Name>
    com.hp.spain.node.wftransaction.WFTransactionHandler
  </Class-Name>
  <Param name="Finish" value="Finish"/>
  <Param name="Cancel" value="Cancel"/>
  <Param name="unlock_beans" value="unlock_beans"/>
  <Param name="WF_TRANSACTION_NAME" value="wf_transaction"/>
  <Param name="rollback_beans" value="rollback_beans"/>
</End-Handler>
```

6 Configuration

6.1 Lock Manager Configuration File

Next the properties that are present in the configuration files are described, in case the need arises of modifying the configuration.

There are two configuration files of the Lock Manager:

LockManager.properties located in $LOCK_MANAGER_HOME/bin. This file has a series of generic parameters:

Parameter Name	Description
LOCK_PENDING_PERIOD	Processing time in milliseconds of the waiting locks
LOCK_PENDING_TIMEOUT	Maximum amount of time a lock can remain in a lock queue awaiting notification
DEAD_LOCK_RISK_THRESHOLD_TIME	Time threshold in milliseconds that creates the risk of deadlock for a lock
KEY_MONITOR_WAIT_TIMEOUT	Maximum amount of time in milliseconds a monitor waits before locking a key
ADMINISTRATOR_LOCKER_NAMES	Locker names with administration permission

Next an example configuration of these variables is shown:

LOCK_PENDING_PERIOD = 5000

LOCK_PENDING_TIMEOUT = 600000

DEAD_LOCK_RISK_THRESHOLD_TIME = 60000

KEY_MONITOR_WAIT_TIMEOUT = 0

ADMINISTRATOR_LOCKER_NAMES = SUPERLOCKER_WEB_1, SUPERLOCKER_WEB_2, SUPERLOCKER_CMD

Also the persistence of the locks can be configured in two ways:

Persistence in files: The information of each lock is stored in a file called **multilock_lockid.dat**, where lockid is the lock's identifier. In this case the only thing that can be configured is the persistence directory.

Nombre Parámetro	Descripción
PERSISTENCE_CLASS	Class that implements the persistence in a file
PERSISTENCE_DIR_PATH	Directory for the persistence file

For example:

PERSISTENCE_CLASS = com.hp.spain.lock.manager.FileDataSource

PERSISTENCE_DIR_PATH = C:/hp/LockManager/data

Persistence in a Database: The information about all the locks is stored in a table called HPSA_LOCKS inside the database referred in the build.properties. In this type of persistence, we can configure the JDBC driver, the maximum number of active pools and the connection URI to the database.

Parameter Name	Description
PERSISTENCE_CLASS	Class that implements the persistence in the DB
POOL_JDBCDRIVER	Class that implements the JDBC driver
POOL_MAXACTIVE	Maximum number of active pools
DATABASE_CONNECTION_URI	Connection URL to the DB
USERNAME	(Optional) The DB user name
PASSWORD	(Optional) The DB encrypted password

For example:

PERSISTENCE_CLASS = com.hp.spain.lock.manager.JdbcDataSource

POOL_JDBCDRIVER = oracle.jdbc.driver.OracleDriver

POOL_MAXACTIVE = 10

DATABASE_CONNECTION_URI = jdbc:oracle:thin:manuel/****@172.16.3.49:1521:HPSA

The parameters USERNAME and PASSWORD are optional and must be used when the password must appear encrypted in the configuration file. If it is specified in the DATABASE_CONNECTION_URI then the password must be specified in clear text. Of course, when USERNAME and PASSWORD are specified the DATABASE_CONNECTION_URI must never contain either the user name or the password. The encrypted password can be obtained using the *crypt* tool provided with HPSA.

For example:

DATABASE_CONNECTION_URI = jdbc:oracle:thin:@172.16.3.49:1521:HPSA

USERNAME=manuel

PASSWORD=**********

The DATABASE_CONNECTION_URI also accepts an Oracle RAC URI.

Finally we can configure the log system of the different log files.

Parameter Name	Description
LOG_MAX_FILE_SIZE	Sets the maximum size for the log file
LOG_MAX_NUM_FILES	Sets the maximum number of backup log files that should be stored before being deleted.
LOG_PATTERN	Date pattern to be used in the created log file when a rotation is produced.
LOG_LEVEL	Log level. In ascending order the possible levels are DEBUF, INFO, WARN, ERROR and FATAL

Example:

LOG_MAX_FILE_SIZE = 5242880

LOG_MAX_NUM_FILES = 10

LOG_PATTERN = %d [%t] %-5p %c\{1} - %m %n

LOG_LEVEL = DEBUG

RmiLockManagerService.policy. File where the RMI security policy is configured. Currently all the actions are allowed for every user.

```
grant {
   permission java.security.AllPermission;
```

```
};
```

6.2 Lock Manager configuration inside de MWFM

The Lock Manager is already configured in the HPSA installation, but if the need arises to change any of the values, here each section is described.

This is the Lock Module's configuration in the mwfm.xml file:

```
<Module>
  <Name>LockModule</Name>
  <Class-Name>
    com.hp.spain.engine.module.lock.manager.LockModule
  </Class-Name>
  <Param name="locker_name" value="MWFM-0"/>
  <Param name="locker_service_ip_address" value="127.0.0.1"/>
  <Param name="unlock_pending_period" value="60000"/>
  <Param name="lock_manager_service_url"
         value="rmi://127.0.0.1:1220/RmiLockManagerService"/>
  <Param name="persistence_dir_path"
         value="C:/hp/OpenView/ServiceActivator/var/tmp/lockers"/>
  <Param name="lock_waiter_mode" value="enqueue_jobs"/>
  <Param name="bean_helper_must_check_locks" value="true"/>
  <Param name="debug" value="false"/>
</Module>
```

Each of the elements is described next:

- Name is used to define the Module's name.
- Class-name, defines the class that will implement the module.

Parameters:

- locker-name: name of the locker HPSA will use to communicate with the Lock Manager
- locker_service_ip_address: ip address the locker will use.
- unlock_pending_period: amount of time a pending unlock will be kept waiting.
- lock_manager_service_url: Url for the Lock Manager service.
- persistence_dir_path: directory for file persistence
- lock_waiter_mode: enqueue jobs means locking requests will be enqueued.
- bean_helper_must_check_locks: the bean helper checks locks
- debug: whether debug mode is on.

6.3 Logs

The Lock Manager's log files are located in the $LOCK_MANAGER_HOME/log directory.

Log Name	Description
RmiLockManagerService.stderr	Shows the Service's errors
RmiLockManagerService.stdout	Shows the operations executed by the administrator service such as locking/unlocking and

	registry/unregistry of lockers.
LockManager.log	The logs that are shown are contained in the file RmiLockManagerService.stdout

7 Command Line Tool

The Lock Manager has a command line client that can be used to launch different actions by executing the scripts provided.

Name of the Script	Description
StartServer	Launches the Lock Manager service
StopServer	Stops the Lock Manager service
showStatus	Shows the status of the Lock Manager service
showLocks	Shows all the current active locks that exist in the Lock Manager
getLockInfo	Shows all the relevant information about a lock (lockId, key, status, ownerId). Usage: getLockInfo <lockId>
Unlock	Unlocks an active lock on the Lock Manager. Usage: unlock <lockId>
getLockersInfo	Shows all the active locking services that exist in the LockManager
forceUnregister	Deregisters a locking service from the Lock Manager. Usage: forceUnregister <lockerName>

Note: All the scripts are executed from the directory $LOCK_MANAGER_HOME/bin

8 Web Client

The Lock Manager's web client is a web application, integrated in the application container, developed for the administration of the lock manager service called LockManager.

Its functioning is outlined in the "User Manual for the Administration of the Lock Module" document.

8.1 Configuration

The configuration of the Lock Manager is stored in the **Lock-Manager-Web.properties** file**,** located in the *$JBOSS/server/diagnostic/deploy/hpovact.sar/activator.war/properties/* directory.

This file contains the variables previously configured in the build.properties file of the machine. The variables contained set the IPs of the machine that is running the Lock Manager service and the machine that will act as administrator (machine where the web application resides).

```
#Configuración del servicio RMI Lock Manager
lockmanager.service.host = 172.16.2.121
lockmanager.service.port = 1220
lockmanager.service.name = RmiLockManagerService

locker.service.host = localhost
locker.service.port = 1230
locker.service.name = RmiLockerService-SUPERLOCKER_WEB_1

locker.name = SUPERLOCKER_WEB_1
```

This application in order to establish communication with the Lock Manager creates a locker.

This locker does not need to be configured in a module inside the mwfm.xml file as it won't be used from any flow. It will only be used in the web application and will have to have administration permissions. Because of this, the name of this locker must be one of the names defined in the **ADMINISTRATOR_LOCKER_NAMES** variable of the service's configuration file 'LockManager.properties'. (See chap. 3.4 Configuration).

Next we can see an example configuration:

```
lockmanager.service.host = 172.16.2.117
lockmanager.service.port = 1220
lockmanager.service.name = RmiLockManagerService

locker.service.host = 172.16.3.49
locker.service.port = 1230
locker.service.name = RmiLockerService-SUPERLOCKER_WEB_1

locker.name = SUPERLOCKER_WEB_1
```

9 Common Problems

9.1 Problems starting up the service

Most of the startup problems of the *LockManager* are related to a bad configuration of the service. Next is a list of a series of possible errors:

1- Log belonging to the LockManager.log file

2008-01-04 14:18:48,281 [main] ERROR JdbcDataSource - LockContainer not restored. It's not posible to create a connection to the database
java.sql.SQLException: Excepción de E/S: The Network Adapter could not establish the connection
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:134)
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:179)
…
ERROR LockManagerImpl - Persistence not restored. It's not posible to create a connection to the database
com.hp.spain.lock.manager.DataSourceException: LockContainer not restored. It's not posible to create a connection to the databaseExcepción de E/S: The Network Adapter could not establish the connection.

A connection could not be established with the DB used to store the locks. Check the connection configuration set with the DATABASE_CONNECTION_URI property in the LockManager.properties file.

2- Log belonging to the RmiLockManagerService.stderr log file

java.rmi.ConnectException: ***Connection refused to host: 172.16.3.65****; nested exception is:*
java.net.ConnectException: Connection timed out: connect
…

Could not connect to the server where the service is launched. Check that the value for the variables **RMI_HOST and RMI_IP** configured in the StartServer file is correct. If the IPs are incorrect, they will probably be badly configured in the rest of the script located in the $LOCK_MANAGER_HOME/bin directory.

9.2 Problems starting up the MWFM

1- When starting up, the LockModule tries to connect with the service to register the locker it has configured and the connection fails. See the RmiLockManager.stdout log file:

08-ene-2008 13:42:21: starting module: LockModule with class com.hp.spain.engine.module.lock.manager.LockModule
2008-01-08 13:42:22,359 [main] DEBUG RmiLockManagerService.WaitersFileDataSource - All waiters restored from filedirectory 'C:/hp/OpenView/ServiceAct
ivator/var/tmp/lockers\MWFM-0\waiters'
2008-01-08 13:42:22,390 [main] DEBUG FileSaver - Restoring file 'C:/hp/OpenView/ServiceActivator/var/tmp/lockers\MWFM-0/pendingUnlocks.dat'
…
com.hp.spain.lock.manager.LockerException: ***Connection to lock manager failed***
at com.hp.spain.lock.manager.RmiLockerService.ensureConnection(RmiLockerService.java:171)
at com.hp.spain.lock.manager.RmiLockerService.getLocks(RmiLockerService.java:302)
at

Normally, either the LockManager has not been launched previously or it did not start up correctly.

2- When starting up the LockModule module and connecting with the service, it tries to register the locker it has configured, and the registration fails.

08-ene-2008 16:20:48: ><registering hooks with the queue manager...
java.rmi.ServerException: RemoteException occurred in server thread; nested exception is:
java.rmi.RemoteException: ***Registration refused. Another locker is already registered with the name MWFM-0***
at sun.rmi.server.UnicastServerRef.dispatch(UnicastServerRef.java:292)
at sun.rmi.transport.Transport$1.run(Transport.java:148)
…

In this case two things could have happened:

That another LockModule with the same locker name has already been registered previously. In this case we must modify the name of the locker, as this identifier must be unique.

That in the last mwfm shutdown, the locker was not correctly unregistered for some strange reason. When it tries to register it again it detects that it is already registered. In this case, we can solve it by deleting the locker's persistence directory $VAR/tmp/lockers/<locker name > and retry starting up.

3. When starting up the LockModule module connects with the service and tries to register the locker it has configured.

2008-01-08 16:10:01,234 [main] DEBUG RmiLockManagerService.MultilocksFileDataSource - All multilocks restored from filedirectory 'C:/hp/OpenView/Serv
iceActivator/var/tmp/lockers\MWFM-0\multilocks'
java.rmi.NoSuchObjectException: ***no such object in table***
at sun.rmi.transport.StreamRemoteCall.exceptionReceivedFromServer(StreamRemoteCall.java:247)
at sun.rmi.transport.StreamRemoteCall.executeCall(StreamRemoteCall.java:223)
at sun.rmi.server.UnicastRef.invoke(UnicastRef.java:133)
…
2008-01-08 16:10:01,843 [RmiUnattendedLockerService] WARN RmiUnattendedLockerService - Register on lock manager failed
08-ene-2008 16:10:02: ><all startup activities completed
08-ene-2008 16:10:02: ><Micro Workflow Manager Startup Complete! Check log files for details.
08-ene-2008 16:10:02: ><Releasing worker threads to process workflows

The problem is that an inconsistence exists between what is stored in the HPSA_LOCKS table, which is the table where the LockManager stores its locks, and the persistence files of the locker. To solve this problem we must shutdown the mwfm service, delete the $VAR/tmp/lockers/<locker_name> directory and try to launch it again.

If the inconsistency continues then we will have to again shutdown the mwfm, delete the locker's persistence directory, restart the lock-manager service and try to launch the mwfm again.