Pacemaker 1.1

Configuration Explained

Un ghid de la A la Z despre Opţiunile de Configurare ale Pacemaker

Andrew Beekhof

Pacemaker 1.1 Configuration Explained
Un ghid de la A la Z despre Opţiunile de Configurare ale Pacemaker
Ediție 1

Author	Andrew Beekhof	andrew@beekhof.net
Translator	Dan Frîncu	df.cluster@gmail.com
	Philipp Marek	philipp.marek@linbit.com
	Tanja Roth	taroth@suse.com
	Lars Marowsky-Bree	lmb@suse.com
	Yan Gao	ygao@suse.com
	Thomas Schraitle	toms@suse.com
	Dejan Muhamedagic	dmuhamedagic@suse.com

Scopul acestui document este de a explica în mod definitiv conceptele folosite pentru a configura Pacemaker. Pentru a reuşi acest lucru în cel mai bun mod, se va concentra în mod exclusiv pe sintaxa XML folosită pentru a configura CIB-ul.

For those that are allergic to XML, there exist several unified shells and GUIs for Pacemaker. However these tools will not be covered at all in this document[1], precisely because they hide the XML.

Additionally, this document is NOT a step-by-step how-to guide for configuring a specific clustering scenario. Although such guides exist, the purpose of this document is to provide an understanding of the building blocks that can be used to construct any type of Pacemaker cluster. Try the *Clusters from Scratch*[2] document instead.

[3] An explanation of CC-BY-SA is available at *http://creativecommons.org/licenses/by-sa/3.0/*
[1] I hope, however, that the concepts explained here make the functionality of these tools more easily understood.
[2] http://www.clusterlabs.org/doc

Cuprins

Listă de figuri

Listă de tabele

Listă de exemple

Prefaţă

Cuprins

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

[1] https://fedorahosted.org/liberation-fonts/

File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.

The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.

To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Notă

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Avertisment

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla[2] against the product **Pacemaker.**

When submitting a bug report, be sure to mention the manual's identifier: *Pacemaker_Explained*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

[2] *http://bugs.clusterlabs.org*

Citeşte-mă-Întâi-pe-Mine

Cuprins

1.1. Domeniul de Aplicare al acestui Document

Scopul acestul document este să explice în mod definitiv conceptele folosite pentru a configura Pacemaker. Pentru a atinge acest lucru, se va concentra exclusiv pe sintaxa XML folosită pentru a configura CIB-ul.

For those that are allergic to XML, there exist several unified shells and GUIs for Pacemaker. However these tools will not be covered at all in this document [1] , precisely because they hide the XML.

Additionally, this document is NOT a step-by-step how-to guide for configuring a specific clustering scenario.

Although such guides exist, the purpose of this document is to provide an understanding of the building blocks that can be used to construct any type of Pacemaker cluster.

1.2. Ce este Pacemaker?

Pacemaker is a cluster resource manager.

It achieves maximum availability for your cluster services (aka. resources) by detecting and recovering from node and resource-level failures by making use of the messaging and membership capabilities provided by your preferred cluster infrastructure (either *Corosync*[2] or *Heartbeat*[3].

Pacemaker's key features include:

- Detectarea şi recuperarea eşecurilor la nivel de nod şi serviciu
- Agnostic d.p.d.v. al stocării, nu sunt cerinţe pentru spaţiu de stocare partajat
- Agnostic d.p.d.v. al resurselor, orice poate fi scriptat poate fi folosit într-un cluster
- Suportă *STONITH*[4] pentru asigurarea integrităţii datelor
- Suportă clustere mici şi mari
- Supports both *quorate*[5] and *resource driven*[6] clusters

[1] I hope, however, that the concepts explained here make the functionality of these tools more easily understood.
[2] http://www.corosync.org/
[3] http://linux-ha.org/wiki/Heartbeat
[4] http://en.wikipedia.org/wiki/STONITH
[5] http://en.wikipedia.org/wiki/Quorum_(Distributed_Systems)
[6] http://devresources.linux-foundation.org/dev/clusters/docs/ResourceDrivenClusters.pdf

- Suportă practic orice *configuraţie redundantă*[7]
- Configuraţie replicată în mod automat care poate fi actualizată de pe orice nod
- Abilitatea de a specifica ordonare, colocare şi anti-colocare la nivelul întregului cluster
- Suport pentru tipuri de servicii avansate
  - Clone: pentru servicii care trebuie să fie active pe mai multe noduri
  - Multi-state: pentru servicii cu mai multe moduri de operare (ex. master/slave, primar/secundar)

1.3. Tipuri de Clustere Pacemaker

Pacemaker nu face nici un fel de presupuneri despre mediul vostru, acest aspect îi permite să suporte practic orice *configuraţie redundantă*[8] incluzând Activ/Activ, Activ/Pasiv, N+1, N+M, N-la-1 şi N-la-N.

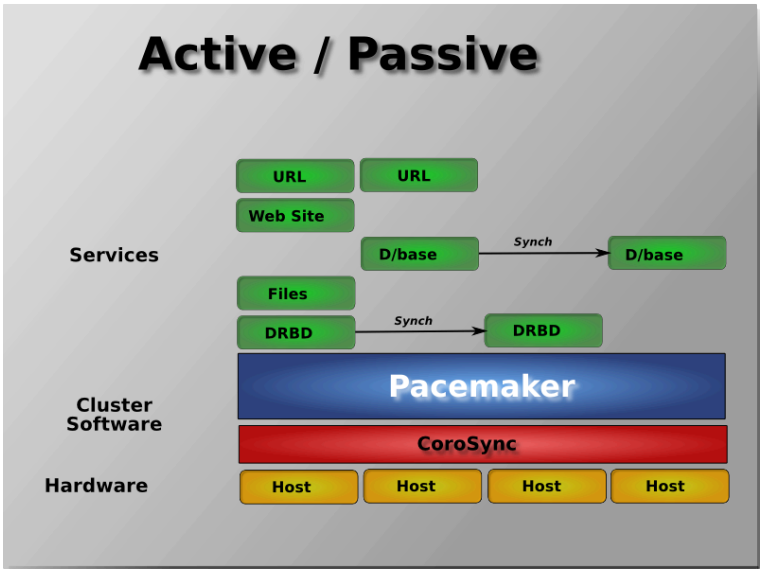


Fig. 1.1. Redundanţă Activă/Pasivă

Clusterele Active/Pasive formate din două noduri care folosesc Pacemaker şi DRBD sunt o soluţie eficientă d.p.d.v. al costului pentru multe situaţii de High Availability.

[7] http://en.wikipedia.org/wiki/High-availability_cluster#Node_configurations
[8] http://en.wikipedia.org/wiki/High-availability_cluster#Node_configurations

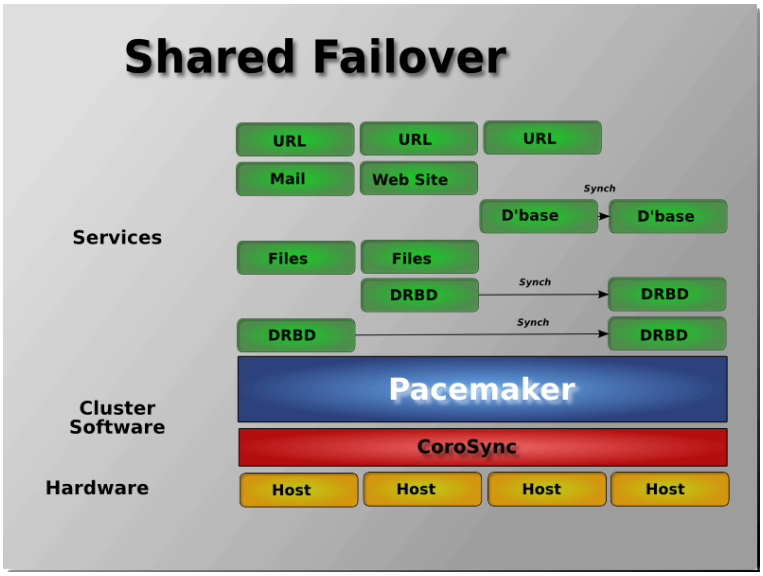


Fig. 1.2. Failover Partajat

Prin suportarea de multe noduri, Pacemaker poate reduce costurile hardware în mod dramatic permiţând mai multor clustere active/pasive să fie combinate şi să împartă un nod comun de backup

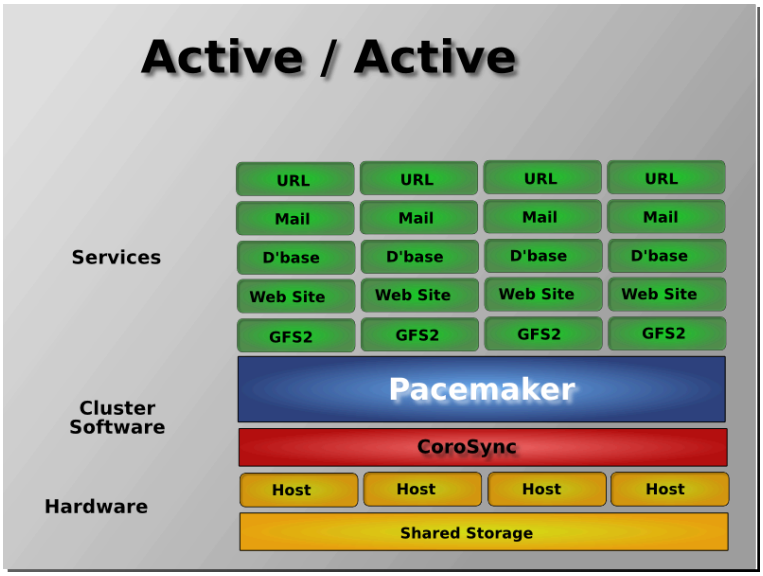


Fig. 1.3. Redundanţă N la N

Când un mediu de stocare partajat este disponibil, fiecare nod poate fi folosit în mod potenţial pentru failover. Pacemaker poate chiar rula mai multe copii ale serviciilor pentru a distribui sarcina de lucru.

1.4. Arhitectura Pacemaker

La cel mai înalt nivel, clusterul este compus din trei părţi:

- Nucleul infrastructurii de cluster care furnizează funcţionalitatea de mesagerie şi apartenenţă (ilustrată cu roşu)
- Non-cluster aware components (illustrated in green).

  In a Pacemaker cluster, these pieces include not only the scripts that knows how to start, stop and monitor resources, but also a local daemon that masks the differences between the different standards these scripts implement.

- A brain (illustrated in blue)

  This component processes and reacts to events from the cluster (nodes leaving or joining) and resources (eg. monitor failures) as well as configuration changes from the administrator. In response to all of these events, Pacemaker will compute the ideal state of the cluster and plot a path to achieve it. This may include moving resources, stopping nodes and even forcing nodes offline with remote power switches.

1.4.1. Vedere Conceptuală a Stivei

Fig. 1.4. Vederea conceptuală a stivei de cluster

When combined with Corosync, Pacemaker also supports popular open source cluster filesystems. footnote:[Even though Pacemaker also supports Heartbeat, the filesystems need to use the stack for messaging and membership and Corosync seems to be what they’re standardizing on.

Technically it would be possible for them to support Heartbeat as well, however there seems little interest in this.]

Due to recent standardization within the cluster filesystem community, they make use of a common distributed lock manager which makes use of Corosync for its messaging capabilities and Pacemaker for its membership (which nodes are up/down) and fencing services.

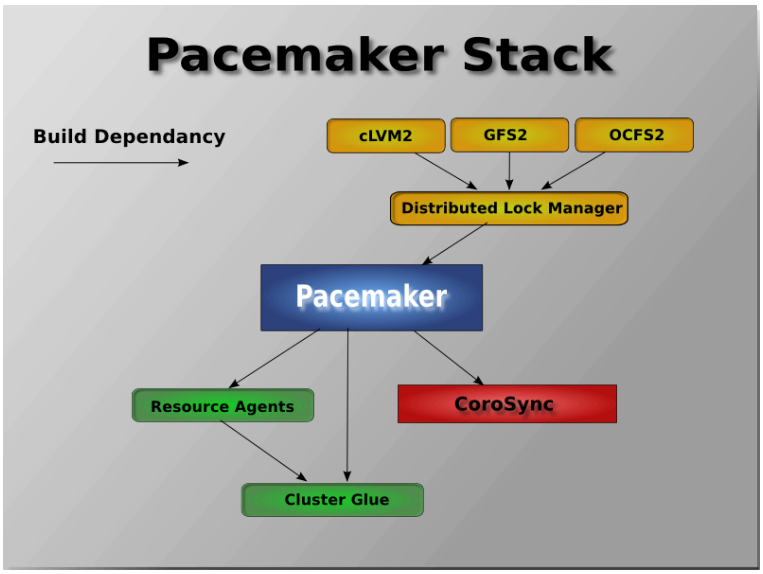


Fig. 1.5. Stiva Pacemaker atunci când rulează pe Corosync

1.4.2. Componente Interne

Pacemaker însuşi este compus din patru componente cheie (ilustrate mai jos cu aceeaşi schemă de culori ca şi în diagrama anterioară):

- CIB (Cluster Information Base)
- CRMd (aka. Cluster Resource Management daemon)
- PEngine (aka. PE sau Policy Engine)
- STONITHd

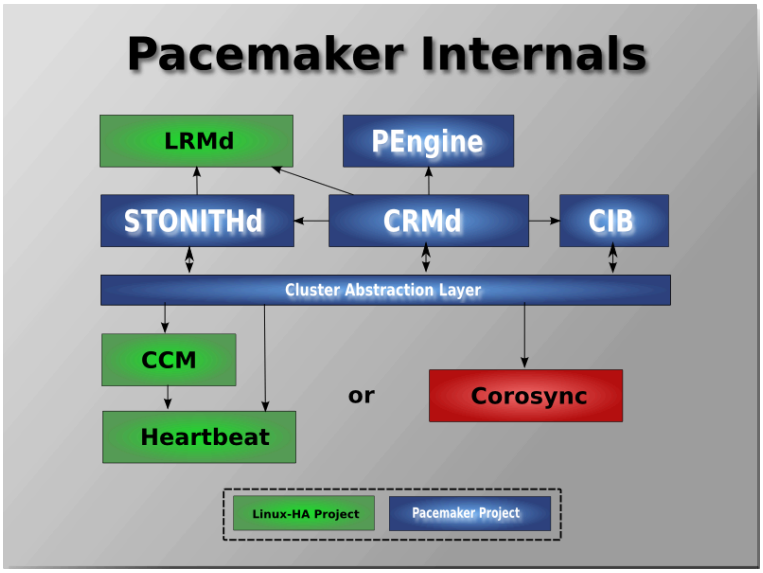


Fig. 1.6. Subsisteme ale unui cluster Pacemaker rulând pe Corosync

The CIB uses XML to represent both the cluster's configuration and current state of all resources in the cluster. The contents of the CIB are automatically kept in sync across the entire cluster and are used by the PEngine to compute the ideal state of the cluster and how it should be achieved.

This list of instructions is then fed to the DC (Designated Controller). Pacemaker centralizes all cluster decision making by electing one of the CRMd instances to act as a master. Should the elected CRMd process (or the node it is on) fail… a new one is quickly established.

The DC carries out PEngine's instructions in the required order by passing them to either the LRMd (Local Resource Management daemon) or CRMd peers on other nodes via the cluster messaging infrastructure (which in turn passes them on to their LRMd process).

Nodurile vecine raportează toate rezultatele operaţiunilor înapoi către DC şi pe baza rezultatelor aşteptate şi a rezultatelor actuale, fie va executa acţiuni care necesitau să aştepte ca şi cele anterioare să termine, sau va anula procesarea şi va ruga PEngine-ul să recalculeze starea ideală a clusterului pe baza rezultatelor neaşteptate.

In some cases, it may be necessary to power off nodes in order to protect shared data or complete resource recovery. For this Pacemaker comes with STONITHd.

STONITH is an acronym for Shoot-The-Other-Node-In-The-Head and is usually implemented with a remote power switch.

In Pacemaker, STONITH devices are modeled as resources (and configured in the CIB) to enable them to be easily monitored for failure, however STONITHd takes care of understanding the STONITH topology such that its clients simply request a node be fenced and it does the rest.

Bazele Configurării

Cuprins

2.1. Aşezarea Configuraţiei

Clusterul este scris folosind notaţie XML şi este împărţit în două secţiuni principale: configurare şi status.

Secţiunea de status conţine istoricul fiecărei resurse de pe fiecare nod şi pe baza acestor date, clusterul poate construi starea curentă completă a clusterului. Sursa autoritativă pentru secţiunea de status este procesul managerului de resurse local (lrmd) pe fiecare nod din cluster iar clusterul va repopula în mod ocazional întreaga secţiune. Din acest motiv nu este scris niciodată pe disc iar administratorii sunt sfătuiţi împotriva modificării în orice fel a acestuia.

Secţiunea de configurare conţine informaţiile mai tradiţionale precum opţiuni ale clusterului, liste de resurse şi indicaţii despre unde ar trebui acestea plasate. Secţiunea de configurare este scopul primar al acestui document.

Secţiunea de configurare în sine este împărţită în patru părţi:

- Opţiuni de configurare (numite **`crm_config`**)
- Noduri
- Resurse
- Relaţii între resurse (numite **`restricţii`**)

> Exemplu 2.1. O configuraţie goală

```
<cib admin_epoch="0" epoch="0" num_updates="0" have-quorum="false">
   <configuration>
      <crm_config/>
      <nodes/>
      <resources/>
      <constraints/>
   </configuration>
   <status/>
</cib>
```

2.2. Starea Curentă a Clusterului

Before one starts to configure a cluster, it is worth explaining how to view the finished product. For this purpose we have created the **crm_mon** utility that will display the current state of an active cluster. It can show the cluster status by node or by resource and can be used in either single-shot or dynamically-updating mode. There are also modes for displaying a list of the operations performed (grouped by node and resource) as well as information about failures.

Folosind acest utilitar, puteţi examina starea clusterului pentru neconcordanţe şi pentru a vedea cum răspunde atunci când provocaţi sau simulaţi eşecuri.

Details on all the available options can be obtained using the **crm_mon --help** command.

Exemplu 2.2. Exemplu de rezultat obţinut din crm_mon

```
============
Last updated: Fri Nov 23 15:26:13 2007
Current DC: sles-3 (2298606a-6a8c-499a-9d25-76242f7006ec)
3 Nodes configured.
5 Resources configured.
============

Node: sles-1 (1186dc9a-324d-425a-966e-d757e693dc86): online
    192.168.100.181    (heartbeat::ocf:IPaddr):    Started sles-1
    192.168.100.182    (heartbeat:IPaddr):         Started sles-1
    192.168.100.183    (heartbeat::ocf:IPaddr):    Started sles-1
    rsc_sles-1         (heartbeat::ocf:IPaddr):    Started sles-1
    child_DoFencing:2  (stonith:external/vmware):  Started sles-1
Node: sles-2 (02fb99a8-e30e-482f-b3ad-0fb3ce27d088): standby
Node: sles-3 (2298606a-6a8c-499a-9d25-76242f7006ec): online
    rsc_sles-2    (heartbeat::ocf:IPaddr):    Started sles-3
    rsc_sles-3    (heartbeat::ocf:IPaddr):    Started sles-3
    child_DoFencing:0    (stonith:external/vmware):    Started sles-3
```

Exemplu 2.3. Exemplu de rezultat obţinut din crm_mon -n

```
============
Last updated: Fri Nov 23 15:26:13 2007
Current DC: sles-3 (2298606a-6a8c-499a-9d25-76242f7006ec)
3 Nodes configured.
5 Resources configured.
============

Node: sles-1 (1186dc9a-324d-425a-966e-d757e693dc86): online
Node: sles-2 (02fb99a8-e30e-482f-b3ad-0fb3ce27d088): standby
Node: sles-3 (2298606a-6a8c-499a-9d25-76242f7006ec): online

Resource Group: group-1
  192.168.100.181    (heartbeat::ocf:IPaddr):    Started sles-1
  192.168.100.182    (heartbeat:IPaddr):        Started sles-1
  192.168.100.183    (heartbeat::ocf:IPaddr):    Started sles-1
rsc_sles-1    (heartbeat::ocf:IPaddr):    Started sles-1
rsc_sles-2    (heartbeat::ocf:IPaddr):    Started sles-3
rsc_sles-3    (heartbeat::ocf:IPaddr):    Started sles-3
Clone Set: DoFencing
  child_DoFencing:0    (stonith:external/vmware):    Started sles-3
  child_DoFencing:1    (stonith:external/vmware):    Stopped
  child_DoFencing:2    (stonith:external/vmware):    Started sles-1
```

Nodul DC (Designated Controller - Controller Desemnat) este locul unde toate deciziile sunt luate şi dacă DC-ul curent eşuează unul nou este ales din nodurile rămase în cluster. Alegerea unui DC nu are nici o semnificaţie pentru administrator dincolo de faptul că logurile acestuia vor fi în general mai interesante.

2.3. Cum Ar Trebui să fie Actualizată Configuraţia

Sunt trei reguli de bază pentru actualizarea configuraţiei clusterului:

- Rule 1 - Never edit the cib.xml file manually. Ever. I'm not making this up.
- Regula 2 - Citiţi Regula 1 din nou.
- Regula 3 - Clusterul va detecta că aţi ignorat regulile 1 & 2 şi va refuza să folosească configuraţia.

Acum că este clar cum să NU actualizăm configuraţia, putem începe să explicăm cum ar trebui să realizăm acest lucru.

Cea mai puternică unealtă pentru modificarea configuraţiei este comanda **`cibadmin`** care comunică cu un cluster funcţional. Cu **`cibadmin`**, utilizatorul poate interoga, adăuga, înlătura, actualiza sau înlocui orice parte a configuraţiei; toate modificările iau efect imediat aşa că nu este necesar să executaţi operaţiuni de tip reîncărcare.

Cel mai simplu mod de a folosi cibadmin este de a-l folosi pentru a salva configuraţia curentă într-un fişier temporar, să editaţi acel fişier cu editorul de text sau XML favorit şi apoi să încărcaţi configuraţia revizuită.

Exemplu 2.4. Folosind cu siguranţă un editor pentru a modifica configuraţia clusterului

```
# cibadmin --query > tmp.xml
# vi tmp.xml
# cibadmin --replace --xml-file tmp.xml
```

Some of the better XML editors can make use of a Relax NG schema to help make sure any changes you make are valid. The schema describing the configuration can normally be found in */usr/lib/heartbeat/pacemaker.rng* on most systems.

Dacă aţi dorit să modificaţi doar secţiunea de resurse, aţi putea alternativ să executaţi

Exemplu 2.5. Folosind cu siguranţă un editor pentru a modifica o subsecţiune din configuraţia clusterlui

```
# cibadmin --query --obj_type resources > tmp.xml
# vi tmp.xml
# cibadmin --replace --obj_type resources --xml-file tmp.xml
```

pentru a evita modificările survenite la orice altă parte a configuraţiei.

2.4. Ştergerea Rapidă a unei Părţi din Configuraţie

Identify the object you wish to delete. Eg. run

Exemplu 2.6. Căutând pentru elemente de configurare legate de STONITH

```
# cibadmin -Q | grep stonith
```

```
<nvpair id="cib-bootstrap-options-stonith-action" name="stonith-action" value="reboot"/>
<nvpair id="cib-bootstrap-options-stonith-enabled" name="stonith-enabled" value="1"/>
<primitive id="child_DoFencing" class="stonith" type="external/vmware">
<lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
<lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
<lrm_resource id="child_DoFencing:1" type="external/vmware" class="stonith">
<lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
<lrm_resource id="child_DoFencing:2" type="external/vmware" class="stonith">
<lrm_resource id="child_DoFencing:0" type="external/vmware" class="stonith">
<lrm_resource id="child_DoFencing:3" type="external/vmware" class="stonith">
```

Next identify the resource's tag name and id (in this case we'll choose **`primitive`** and **`child_DoFencing`**). Then simply execute:

```
# cibadmin --delete --crm_xml '<primitive id="child_DoFencing"/>'
```

2.5. Updatând Configuraţia Fără să Folosim XML

Câteva task-uri comune pot fi efectuate de asemenea cu unul dintre utilitarele de nivel înalt pentru a evita nevoia de a citi sau edita XML.

Pentru a activa stonith de exemplu, aţi putea rula:

```
# crm_attribute --attr-name stonith-enabled --attr-value true
```

Sau pentru a vedea dacă **`somenode`** îi este permis să ruleze resurse, există:

```
# crm_standby --get-value --node-uname somenode
```

Sau pentru a afla locaţia curentă a **`my-test-rsc`**, aţi putea folosi:

```
# crm_resource --locate --resource my-test-rsc
```

2.6. Realizând Modificări de Configurare într-un Sandbox

Often it is desirable to preview the effects of a series of changes before updating the configuration atomically. For this purpose we have created **`crm_shadow`** which creates a "shadow" copy of the configuration and arranges for all the command line tools to use it.

To begin, simply invoke **`crm_shadow`** and give it the name of a configuration to create [1] ; be sure to follow the simple on-screen instructions.

[1] Shadow copies are identified with a name, making it possible to have more than one.

Avertisment

Read the above carefully, failure to do so could result in you destroying the cluster's active configuration!

Exemplu 2.7. Crearea şi prezentarea sandbox-ului activ

```
# crm_shadow --create test
Setting up shadow instance
Type Ctrl-D to exit the crm_shadow shell
shadow[test]:
shadow[test] # crm_shadow --which
test
```

From this point on, all cluster commands will automatically use the shadow copy instead of talking to the cluster's active configuration. Once you have finished experimenting, you can either commit the changes, or discard them as shown below. Again, be sure to follow the on-screen instructions carefully.

For a full list of **crm_shadow** options and commands, invoke it with the <parameter>--help</parameter> option.

Exemplu 2.8. Folosirea unui sandbox pentru a realiza mai multe modificări în mod atomic

```
shadow[test] # crm_failcount -G -r rsc_c001n01
 name=fail-count-rsc_c001n01 value=0
shadow[test] # crm_standby -v on -n c001n02
shadow[test] # crm_standby -G -n c001n02
name=c001n02 scope=nodes value=on
shadow[test] # cibadmin --erase --force
shadow[test] # cibadmin --query
<cib cib_feature_revision="1" validate-
with="pacemaker-1.0" admin_epoch="0" crm_feature_set="3.0" have-quorum="1" epoch="112"
      dc-uuid="c001n01" num_updates="1" cib-last-written="Fri Jun 27 12:17:10 2008">
    <configuration>
       <crm_config/>
       <nodes/>
       <resources/>
       <constraints/>
    </configuration>
    <status/>
</cib>
 shadow[test] # crm_shadow --delete test --force
 Now type Ctrl-D to exit the crm_shadow shell
 shadow[test] # exit
 # crm_shadow --which
 No shadow instance provided
 # cibadmin -Q
<cib cib_feature_revision="1" validate-
with="pacemaker-1.0" admin_epoch="0" crm_feature_set="3.0" have-quorum="1" epoch="110"
       dc-uuid="c001n01" num_updates="551">
    <configuration>
       <crm_config>
          <cluster_property_set id="cib-bootstrap-options">
             <nvpair id="cib-bootstrap-1" name="stonith-enabled" value="1"/>
             <nvpair id="cib-bootstrap-2" name="pe-input-series-max" value="30000"/>
```

Realizarea de modificări într-un sandbox şi apoi verificarea că, configuraţia reală nu este atinsă.

2.7. Testarea Modificărilor Voastre de Configurare

We saw previously how to make a series of changes to a "shadow" copy of the configuration. Before loading the changes back into the cluster (eg. **`crm_shadow --commit mytest --force`**), it is often advisable to simulate the effect of the changes with **`crm_simulate`**, eg.

```
# crm_simulate --live-check -VVVVV --save-graph tmp.graph --save-dotfile tmp.dot
```

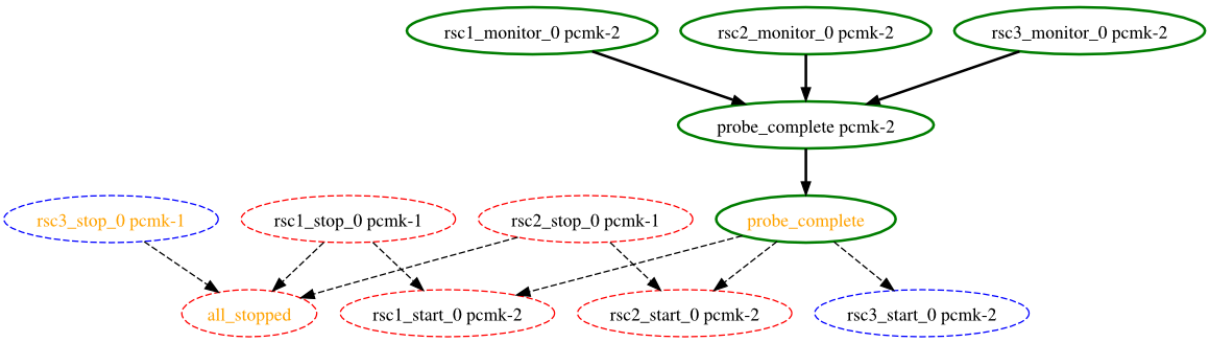
The tool uses the same library as the live cluster to show what it would have done given the supplied input. It's output, in addition to a significant amount of logging, is stored in two files **`tmp.graph`** and **`tmp.dot`**, both are representations of the same thing — the cluster's response to your changes.

In the graph file is stored the complete transition, containing a list of all the actions, their parameters and their pre-requisites. Because the transition graph is not terribly easy to read, the tool also generates a Graphviz dot-file representing the same information.

2.8. Interpretarea rezultatului de ieşire al Graphviz

- Săgeţile indică dependinţele de ordonare
- Săgeţile întrerupte de cratime indică dependinţe care nu sunt prezente în graful de tranziţie
- Acţiunile cu o margine întreruptă cu cratime de orice culoare nu se formează ca parte a grafului de tranziţie
- Acţiunile cu o margine verde se formează ca parte a grafului de tranziţie
- Acţiunile cu o margine roşie sunt cele pe care clusterul ar dori să le execute dar sunt neexecutabile
- Acţiunile cu o margine albastră sunt cele pe care clusterul nu consideră că trebuie executate
- Acţiunile cu text portocaliu sunt acţiuni pseudo/de prefacere pe care clusterul le foloseşte pentru a simplifica graful
- Acţiunile cu text negru sunt trimise la LRM
- Acţiunile resurselor au textul de forma *`rsc_action_intervalnode`*
- Orice acţiune care depine de o acţiune cu o margine roşie nu va putea să se execute.
- Buclele de execuţie sunt *foarte* rele. Vă rugăm să le raportaţi echipei de dezvoltare.

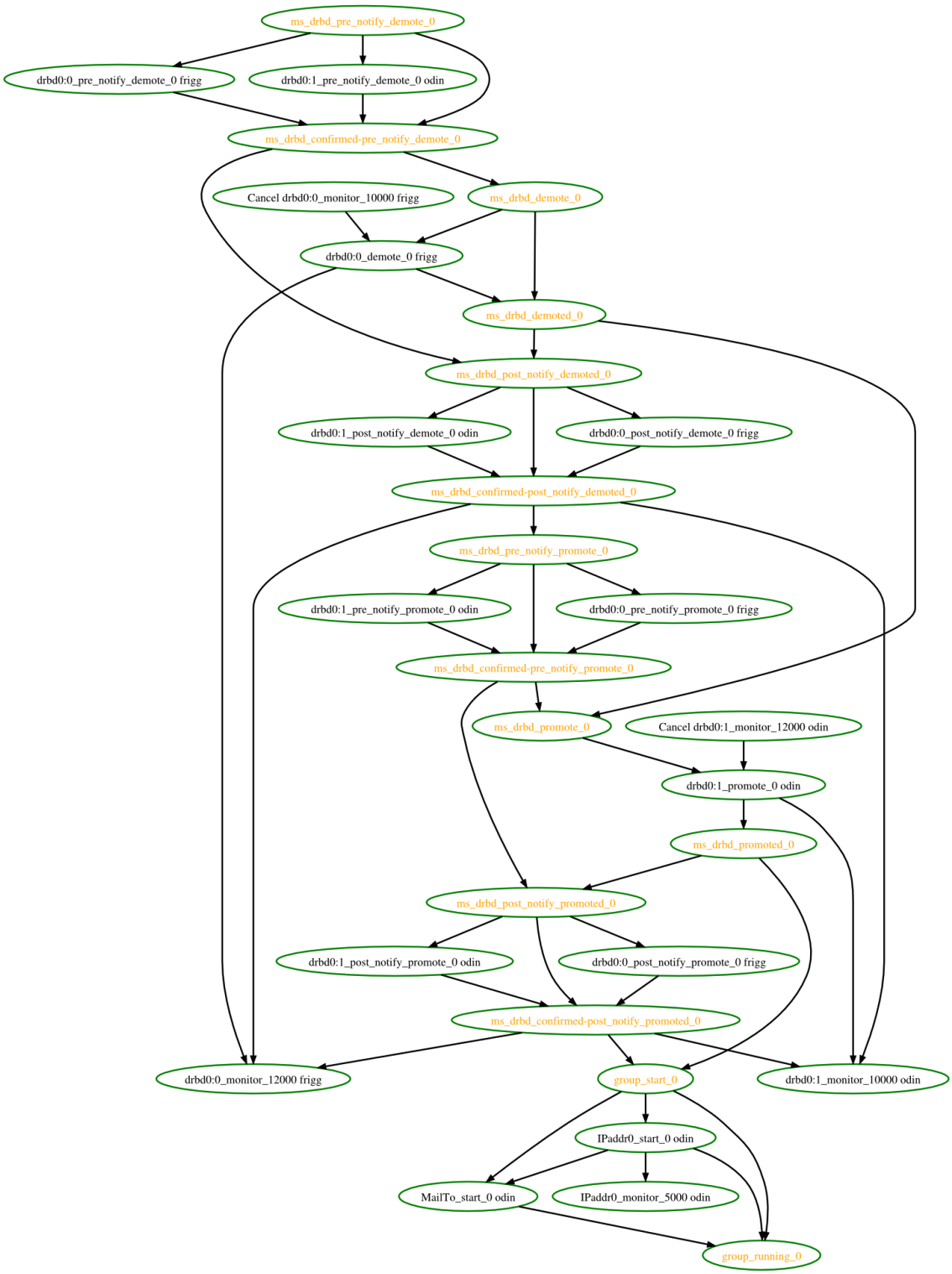
2.8.1. Tranziţia unui Cluster Mic



În exemplul de mai sus, se pare că un nod nou, **node2**, a ajuns online şi clusterul verifică să se asigure că **rsc1**, **rsc2** şi **rsc3**, nu rulează deja acolo (aspect indicat de intrările ***_monitor_0**). Odată ce a realizat acest lucru şi mergând pe presupunerea că resursele nu erau active acolo, ar fi preferat să oprească **rsc1** şi **rsc2** pe **node1** şi să le mute pe **node2**. Totuşi se pare că există o problemă şi clusterul nu poate sau nu îi este permis să execute operaţiunile de oprire, fapt care implică neputinţa de a efectua acţiunile de pornire de asemenea. Pentru un motiv anume clusterul nu vrea să pornească **rsc3** nicăieri.

For information on the options supported by **crm_simulate**, use the **--help** option.

2.8.2. Tranziţii Complexe ale Clusterului



2.9. Trebuie să Actualizez Configuraţia pe toate Nodurile Clusterului?

Nu. Orice modificări sunt sincronizate imediat către ceilalţi membri activi ai clusterului.

Pentru a reduce consumul de lăţime de bandă, clusterul transmite numai actualizările incrementale care rezultă din modificările realizate de voi şi foloseşte hash-uri MD5 pentru a se asigura că fiecare copie este complet consistentă.

Opţiunile Clusterului

Cuprins

3.1. Special Options

Motivul pentru care aceste câmpuri să fie plasate la nivelul cel mai înalt în loc să fie cu restul opţiunilor clusterului este pur şi simplu legat de parsare. Aceste opţiuni sunt folosite de către baza de date cu configuraţii care este, prin design, în principal ignorantă faţă de conţinutul pe care îl deţine. Aşa că decizia a fost luată de a le plasa într-o locaţie uşor de găsit.

3.2. Configuration Version

When a node joins the cluster, the cluster will perform a check to see who has the best configuration based on the fields below. It then asks the node with the highest (**`admin_epoch`**, **`epoch`**, **`num_updates`**) tuple to replace the configuration on all the nodes - which makes setting them, and setting them correctly, very important.

Tabel 3.1. Proprietăţi ale Versiunii Configuraţiei

Câmp	Descriere
`admin_epoch`	Never modified by the cluster. Use this to make the configurations on any inactive nodes obsolete. *Nu setaţi niciodată această valoare la zero*, în astfel de cazuri clusterul nu poate face diferenţa între configuraţia voastră şi cea "goală" folosită atunci când nu este găsit nimic pe disc.
`epoch`	Incremented every time the configuration is updated (usually by the admin)
`num_updates`	Incremented every time the configuration or status is updated (usually by the cluster)

3.3. Alte Câmpuri

Tabel 3.2. Proprietăţi care Controlează Validarea

Câmp	Descriere
`validate-with`	Determines the type of validation being done on the configuration. If set to "none", the cluster will not verify that updates conform to

Câmp	Descriere
	the DTD (nor reject ones that don't). This option can be useful when operating a mixed version cluster during an upgrade.

3.4. Câmpuri Menţinute de către Cluster

Tabel 3.3. Proprietăţi Menţinute de către Cluster

Câmp	Descriere
cib-last-written	Indicates when the configuration was last written to disk. Informational purposes only.
dc-uuid	Indicates which cluster node is the current leader. Used by the cluster when placing resources and determining the order of some events.
have-quorum	Indicates if the cluster has quorum. If false, this may mean that the cluster cannot start resources or fence other nodes. See **no-quorum-policy** below.

Note that although these fields can be written to by the admin, in most cases the cluster will overwrite any values specified by the admin with the "correct" ones. To change the **admin_epoch**, for example, one would use:

```
# cibadmin --modify --crm_xml '<cib admin_epoch="42"/>'
```

Un set complet de câmpuri ar arăta ceva de genul acesta:

Exemplu 3.1. Un exemplu al câmpurilor setate pentru un obiect CIB

```
<cib have-quorum="true" validate-with="pacemaker-1.0"
  admin_epoch="1" epoch="12" num_updates="65"
  dc-uuid="ea7d39f4-3b94-4cfa-ba7a-952956daabee">
```

3.5. Opţiunile Clusterului

Opţiunile clusterului, aşa cum v-aţi aştepta, controlează cum se comportă clusterul când se confruntă cu anumite situaţii.

They are grouped into sets and, in advanced configurations, there may be more than one. [1] For now we will describe the simple case where each option is present at most once.

3.6. Opţiuni Disponibile ale Clusterului

Tabel 3.4. Opţiunile Clusterului

Opţiune	Valoare implicită	Descriere
batch-limit	30	The number of jobs that the TE is allowed to execute in parallel. The "correct" value will depend on the speed and load of your network and cluster nodes.

[1] This will be described later in the section on *Chapter 8, Rules* where we will show how to have the cluster use different sets of options during working hours (when downtime is usually to be avoided at all costs) than it does during the weekends (when resources can be moved to the their preferred hosts without bothering end users)

Opţiune	Valoare implicită	Descriere
migration-limit	-1 (unlimited)	The number of migration jobs that the TE is allowed to execute in parallel on a node.
no-quorum-policy	stop	What to do when the cluster does not have quorum. Allowed values: * ignore - continue all resource management * freeze - continue resource management, but don't recover resources from nodes not in the affected partition * stop - stop all resources in the affected cluster partition * suicide - fence all nodes in the affected cluster partition
symmetric-cluster	TRUE	Can all resources run on any node by default?
stonith-enabled	TRUE	Should failed nodes and nodes with resources that can't be stopped be shot? If you value your data, set up a STONITH device and enable this. Dacă este true, sau nu este setată, clusterul va refuza să pornească resurse decât dacă unul sau mai multe dispozitive STONITH au fost configurate de asemenea.
stonith-action	reboot	Action to send to STONITH device. Allowed values: reboot, off. The value *poweroff* is also allowed, but is only used for legacy devices.
cluster-delay	60s	Round trip delay over the network (excluding action execution). The "correct" value will depend on the speed and load of your network and cluster nodes.
stop-orphan-resources	TRUE	Should deleted resources be stopped?
stop-orphan-actions	TRUE	Should deleted actions be cancelled?
start-failure-is-fatal	TRUE	When set to FALSE, the cluster will instead use the resource's **failcount** and value for **resource-failure-stickiness**.
pe-error-series-max	-1 (all)	The number of PE inputs resulting in ERRORs to save. Used when reporting problems.
pe-warn-series-max	-1 (all)	The number of PE inputs resulting in WARNINGs to save. Used when reporting problems.
pe-input-series-max	-1 (all)	The number of "normal" PE inputs to save. Used when reporting problems.

You can always obtain an up-to-date list of cluster options, including their default values, by running the **pengine metadata** command.

3.7. Querying and Setting Cluster Options

Cluster options can be queried and modified using the **crm_attribute** tool. To get the current value of **cluster-delay**, simply use:

```
# crm_attribute --attr-name cluster-delay --get-value
```

care este scrisă mai simplu ca

```
# crm_attribute --get-value -n cluster-delay
```

If a value is found, you'll see a result like this:

```
# crm_attribute --get-value -n cluster-delay
 name=cluster-delay value=60s
```

Însă dacă nici o valoare nu este găsită, utilitarul va arăta o eroare:

```
# crm_attribute --get-value -n clusta-deway`
name=clusta-deway value=(null)
Error performing operation: The object/attribute does not exist
```

To use a different value, eg. **30**, simply run:

```
# crm_attribute --attr-name cluster-delay --attr-value 30s
```

To go back to the cluster's default value you can delete the value, for example with this command:

```
# crm_attribute --attr-name cluster-delay --delete-attr
```

3.8. Când Opţiunile sunt Listate Mai Mult De O Dată

If you ever see something like the following, it means that the option you're modifying is present more than once.

Exemplu 3.2. Ştergerea unei opţiuni care este listată de două ori

```
# crm_attribute --attr-name batch-limit --delete-attr

Multiple attributes match name=batch-limit in crm_config:
Value: 50          (set=cib-bootstrap-options, id=cib-bootstrap-options-batch-limit)
Value: 100         (set=custom, id=custom-batch-limit)
Please choose from one of the matches above and supply the 'id' with --attr-id
```

In such cases follow the on-screen instructions to perform the requested action. To determine which value is currently being used by the cluster, please refer to *Chapter 8, Rules*.

Nodurile Clusterului

Cuprins

4.1. Definirea unui Nod de Cluster

Fiecare nod în cluster va avea o intrare în secţiunea de noduri conţinând UUID-ul, uname-ul şi tipul acestuia.

Exemplu 4.1. Example Heartbeat cluster node entry

```
<node id="1186dc9a-324d-425a-966e-d757e693dc86" uname="pcmk-1" type="normal"/>
```

Exemplu 4.2. Example Corosync cluster node entry

```
<node id="101" uname="pcmk-1" type="normal"/>
```

In normal circumstances, the admin should let the cluster populate this information automatically from the communications and membership data. However for Heartbeat, one can use the **`crm_uuid`** tool to read an existing UUID or define a value before the cluster starts.

4.2. Where Pacemaker Gets the Node Name

Traditionally, Pacemaker required nodes to be referred to by the value returned by **`uname -n`**. This can be problematic for services that require the **`uname -n`** to be a specific value (ie. for a licence file).

Since version 2.0.0 of Pacemaker, this requirement has been relaxed for clusters using Corosync 2.0 or later. The name Pacemaker uses is:

1. The value stored in *corosync.conf* under **`ring0_addr`** in the **`nodelist`**, if it does not contain an IP address; otherwise
2. The value stored in *corosync.conf* under **`name`** in the **`nodelist`**; otherwise
3. The value of **`uname -n`**

Pacemaker provides the **crm_node -n** command which displays the name used by a running cluster.

If a Corosync nodelist is used, **crm_node --name-for-id $number** is also available to display the name used by the node with the corosync **nodeid** of *$number*, for example: **crm_node --name-for-id 2**.

4.3. Descrierea unui Nod de Cluster

Beyond the basic definition of a node the administrator can also describe the node's attributes, such as how much RAM, disk, what OS or kernel version it has, perhaps even its physical location. This information can then be used by the cluster when deciding where to place resources. For more information on the use of node attributes, see *Chapter 8, Rules*.

Node attributes can be specified ahead of time or populated later, when the cluster is running, using **crm_attribute**.

Below is what the node's definition would look like if the admin ran the command:

Exemplu 4.3. Rezultatul folosirii crm_attribute pentru a specifica pe ce kernel rulează pcmk-1

```
# crm_attribute --type nodes --node-uname pcmk-1 --attr-name kernel --attr-value `uname -r`
```

```
<node uname="pcmk-1" type="normal" id="101">
   <instance_attributes id="nodes-101">
     <nvpair id="kernel-101" name="kernel" value="2.6.16.46-0.4-default"/>
   </instance_attributes>
</node>
```

A simpler way to determine the current value of an attribute is to use **crm_attribute** command again:

```
# crm_attribute --type nodes --node-uname pcmk-1 --attr-name kernel --get-value
```

By specifying **--type nodes** the admin tells the cluster that this attribute is persistent. There are also transient attributes which are kept in the status section which are "forgotten" whenever the node rejoins the cluster. The cluster uses this area to store a record of how many times a resource has failed on that node but administrators can also read and write to this section by specifying **--type status**.

4.4. Corosync

4.4.1. Adding a New Corosync Node

Adding a new node is as simple as installing Corosync and Pacemaker, and copying */etc/corosync/corosync.conf* and */etc/corosync/authkey* (if it exists) from an existing node. You may need to modify the **mcastaddr** option to match the new node's IP address.

Dacă apare în log-uri un mesaj de la Corosync conţinând "Invalid digest", cheile nu sunt consistente între maşini.

4.4.2. Removing a Corosync Node

Because the messaging and membership layers are the authoritative source for cluster nodes, deleting them from the CIB is not a reliable solution. First one must arrange for corosync to forget about the node (*pcmk-1* in the example below).

Pe gazda care va fi înlăturată:

1. Stop the cluster: **`/etc/init.d/corosync stop`**

Ulterior, de pe unul din nodurile rămase active ale clusterului:

1. Tell Pacemaker to forget about the removed host:

   ```
   # crm_node -R pcmk-1
   ```

   This includes deleting the node from the CIB

Notă

This proceedure only works for versions after 1.1.8

4.4.3. Replacing a Corosync Node

The five-step guide to replacing an existing cluster node:

1. Asiguraţi-vă că nodul vechi este oprit de tot.
2. Daţi maşinii noi acelaşi hostname şi adresă IP ca celei vechi
3. Instalaţi soft-ul de cluster :-)
4. Copy */etc/corosync/corosync.conf* and */etc/corosync/authkey* (if it exists) to the new node
5. Porniţi noul nod în cluster

Dacă apare în log-uri un mesaj de la Corosync conţinând "Invalid digest", cheile nu sunt consistente între maşini.

4.5. CMAN

4.5.1. Adding a New CMAN Node

4.5.2. Removing a CMAN Node

4.6. Heartbeat

4.6.1. Adding a New Heartbeat Node

Provided you specified **`autojoin any`** in *ha.cf*, adding a new node is as simple as installing heartbeat and copying *ha.cf* and *authkeys* from an existing node.

If you don't want to use **`autojoin`**, then after setting up *ha.cf* and *authkeys*, you must use **`hb_addnode`** before starting the new node.

4.6.2. Removing a Heartbeat Node

Because the messaging and membership layers are the authoritative source for cluster nodes, deleting them from the CIB is not a reliable solution.

First one must arrange for Heartbeat to forget about the node (pcmk-1 in the example below).

Pe gazda care va fi înlăturată:

1. Stop the cluster: **`/etc/init.d/corosync stop`**

Ulterior, de pe unul din nodurile rămase active ale clusterului:

1. Tell Heartbeat the node should be removed

```
# hb_delnode pcmk-1
```

1. Tell Pacemaker to forget about the removed host:

```
# crm_node -R pcmk-1
```

Notă

This proceedure only works for versions after 1.1.8

4.6.3. Replacing a Heartbeat Node

The seven-step guide to replacing an existing cluster node:

1. Asiguraţi-vă că nodul vechi este oprit de tot.
2. Daţi maşinii noi acelaşi hostname ca şi celei vechi
3. Go to an active cluster node and look up the UUID for the old node in */var/lib/heartbeat/hostcache*
4. Instalaţi soft-ul de cluster
5. Copy *ha.cf* and *authkeys* to the new node

6. On the new node, populate it's UUID using **`crm_uuid -w`** and the UUID from step 2
7. Porniţi noul nod în cluster

Resursele Clusterului

Cuprins

5.1. Ce este o Resursă de Cluster

The role of a resource agent is to abstract the service it provides and present a consistent view to the cluster, which allows the cluster to be agnostic about the resources it manages.

The cluster doesn't need to understand how the resource works because it relies on the resource agent to do the right thing when given a **`start`**, **`stop`** or **`monitor`** command.

Din acest motiv este imperativ ca agenţii de resursă să fie testaţi corespunzător.

În mod normal agenţii de resursă vin sub forma de scripturi de shell, însă aceştia pot fi scrişi folosind orice limbaj de programare (cum ar fi C, Python sau Perl) cu care este confortabil autorul.

5.2. Clase de Resurse Suportate

There are five classes of agents supported by Pacemaker:

- OCF
- LSB
- Upstart
- Systemd
- Fencing
- Service

Version 1 of Heartbeat came with its own style of resource agents and it is highly likely that many people have written their own agents based on its conventions. [1]

Although deprecated with the release of Heartbeat v2, they were supported by Pacemaker up until the release of 1.1.8 to enable administrators to continue to use these agents.

5.2.1. Open Cluster Framework

The OCF standard [2] [3] is basically an extension of the Linux Standard Base conventions for init scripts to:

- suporte parametri
- să îi facă auto descriptivi şi
- extensibili

OCF specs have strict definitions of the exit codes that actions must return. [4]

The cluster follows these specifications exactly, and giving the wrong exit code will cause the cluster to behave in ways you will likely find puzzling and annoying. In particular, the cluster needs to distinguish a completely stopped resource from one which is in some erroneous and indeterminate state.

Parameters are passed to the script as environment variables, with the special prefix **`OCF_RESKEY_`**. So, a parameter which the user thinks of as ip it will be passed to the script as **`OCF_RESKEY_ip`**. The number and purpose of the parameters is completely arbitrary, however your script should advertise any that it supports using the **`meta-data`** command.

Clasa OCF este cea mai preferată din moment ce este un standard al industriei, foarte flexibilă (permiţând parametrii să fie pasaţi agenţilor într-o manieră care nu ţine cont de poziţia acestora) şi auto-descriptivă.

For more information, see the *reference*[5] and *Anexa B, Mai Multe Despre Agenţii de Resursă OCF*.

5.2.2. Linux Standard Base

LSB resource agents are those found in */etc/init.d*.

Generally they are provided by the OS/distribution and, in order to be used with the cluster, they must conform to the LSB Spec. [6]

Many distributions claim LSB compliance but ship with broken init scripts. For details on how to check if your init script is LSB-compatible, see *Anexa G, Este acest script de init compatibil LSB?*. The most common problems are:

[1] See *http://wiki.linux-ha.org/HeartbeatResourceAgent* for more information
[2] *http://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD* - at least as it relates to resource agents.
[3] The Pacemaker implementation has been somewhat extended from the OCF Specs, but none of those changes are incompatible with the original OCF specification.
[4] Included with the cluster is the ocf-tester script, which can be useful in this regard.
[5] http://www.linux-ha.org/wiki/OCF_Resource_Agents
[6] See *http://refspecs.linux-foundation.org/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html* for the LSB Spec (as it relates to init scripts).

- Neimplementarea în vreun fel a operaţiunii status
- Neobservarea status-urilor corecte de ieşire pentru acţiuni start/stop/status
- Pornirea unei resurse deja pornite returnează o eroare (acest aspect încalcă specificaţia LSB)
- Oprirea unei resurse deja oprită returnează o eroare (acest aspect încalcă specificaţia LSB)

5.2.3. Systemd

Some newer distributions have replaced the old *SYS-V*[7] style of initialization daemons (and scripts) with an alternative called *Systemd*[8].

Pacemaker is able to manage these services *if they are present*.

Instead of **`init scripts`**, systemd has **`unit files`**. Generally the services (or unit files) are provided by the OS/distribution but there are some instructions for converting from init scripts at: *http://0pointer.de/blog/projects/systemd-for-admins-3.html*

Notă

Remember to make sure the computer is **`not`** configured to start any services at boot time that should be controlled by the cluster.

5.2.4. Upstart

Some newer distributions have replaced the old *SYS-V*[9] style of initialization daemons (and scripts) with an alternative called *Upstart*[10].

Pacemaker is able to manage these services *if they are present*.

Instead of **`init scripts`**, upstart has **`jobs`**. Generally the services (or jobs) are provided by the OS/distribution.

Notă

Remember to make sure the computer is **`not`** configured to start any services at boot time that should be controlled by the cluster.

5.2.5. System Services

[7] http://en.wikipedia.org/wiki/Init#SysV-style
[8] http://www.freedesktop.org/wiki/Software/systemd
[9] http://en.wikipedia.org/wiki/Init#SysV-style
[10] http://upstart.ubuntu.com

Since there are now many "common" types of system services (**systemd**, **upstart**, and **lsb**), Pacemaker supports a special alias which intelligently figures out which one applies to a given cluster node.

This is particularly useful when the cluster contains a mix of **systemd**, **upstart**, and **lsb**.

In order, Pacemaker will try to find the named service as:

1. an LSB (SYS-V) init script
2. a Systemd unit file
3. an Upstart job

5.2.6. STONITH

Mai există o clasă adiţională, STONITH, care este folosită exclusiv pentru resurse relevante în procesul de evacuare forțată. Acest aspect este discutat mai târziu în *Chapter 13, STONITH*.

5.3. Resource Properties

Aceste valori îi spun clusterului care script să îl folosească pentru resursă, unde să găsească acel script şi care sunt standardele la care acesta aderă.

Tabel 5.1. Proprietăţile unei Resurse Primitive

Câmp	Descriere
id	Your name for the resource
class	The standard the script conforms to. Allowed values: **ocf**, **service**, **upstart**, **systemd**, **lsb**, **stonith**
type	The name of the Resource Agent you wish to use. Eg. *IPaddr* or *Filesystem*
provider	The OCF spec allows multiple vendors to supply the same ResourceAgent. To use the OCF resource agents supplied with Heartbeat, you should specify **heartbeat** here.

Resource definitions can be queried with the **crm_resource** tool. For example

```
# crm_resource --resource Email --query-xml
```

might produce:

Exemplu 5.1. An example system resource

```
<primitive id="Email" class="service" type="exim"/>
```

Notă

One of the main drawbacks to system services (such as LSB, Systemd and Upstart) resources is that they do not allow any parameters!

Exemplu 5.2. Un exemplu de resursă OCF

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
   <instance_attributes id="params-public-ip">
      <nvpair id="public-ip-addr" name="ip" value="1.2.3.4"/>
   </instance_attributes>
</primitive>
```

5.4. Opţiuni ale Resurselor

Options are used by the cluster to decide how your resource should behave and can be easily set using the **--meta** option of the **crm_resource** command.

Tabel 5.2. Opţiuni pentru o Resursă Primitivă

Câmp	Valoarea implicită	Descriere
priority	**0**	If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active.
target-role	**Started**	În ce stare ar trebui să încerce clusterul să menţină această resursă? Valori permise: * *Stopped* - Force the resource to be stopped * *Started* - Allow the resource to be started (In the case of *multi-state* resources, they will not promoted to master) * *Master* - Allow the resource to be started and, if appropriate, promoted
is-managed	**TRUE**	Is the cluster allowed to start and stop the resource? Allowed values: **true**, **false**
resource-stickiness	Calculated	How much does the resource prefer to stay where it is? Defaults to the value of **resource-stickiness** in the **rsc_defaults** section
requires	Calculated	Under what conditions can the resource be started. (*Since 1.1.8*) Defaults to **fencing** unless **stonith-enabled** is *false* or **class** is *stonith* - under those conditions the default is **quorum**. Possible values: * *nothing* - can always be started * *quorum* - The cluster can only start this resource if a majority of the configured nodes are active * *fencing* - The cluster can only start this resource if a majority of the configured nodes are active *and* any failed or unknown nodes have been powered off. * *unfencing* - The cluster can only start this resource if a majority of the configured nodes are active *and* any failed or unknown nodes have been powered off *and* only on nodes that have been *unfenced* indexterm: Option[requires,Resource]

Câmp	Valoarea implicită	Descriere
migration-threshold	**INFINITY** (dezactivat)	How many failures may occur for this resource on a node, before this node is marked ineligible to host this resource.
failure-timeout	**0** (dezactivat)	How many seconds to wait before acting as if the failure had not occurred, and potentially allowing the resource back to the node on which it failed.
multiple-active	**stop_start**	Ce ar trebui să realizeze clusterul dacă găseşte vreodată o resursă activă pe mai mult de un nod. Valori permise: * *block* - mark the resource as unmanaged * *stop_only* - stop all active instances and leave them that way * *stop_start* - stop all active instances and start the resource in one location only

Dacă aţi efectuat următoarele comenzi pe resursa LSB Email anterioară

```
# crm_resource --meta --resource Email --set-parameter priority --property-value 100
# crm_resource --meta --resource Email --set-parameter multiple-active --property-value block
```

definiţia rezultantă a resursei ar fi

Exemplu 5.3. O resursă LSB cu opţiuni ale clusterului

```
<primitive id="Email" class="lsb" type="exim">
   <meta_attributes id="meta-email">
      <nvpair id="email-priority" name="priority" value="100"/>
      <nvpair id="email-active" name="multiple-active" value="block"/>
   </meta_attributes>
</primitive>
```

5.5. Setarea de Valori Implicite Globale pentru Opţiunile Clusterului

To set a default value for a resource option, simply add it to the **rsc_defaults** section with **crm_attribute**. Thus,

```
# crm_attribute --type rsc_defaults --attr-name is-managed --attr-value false
```

ar preveni clusterul de a porni sau opri orice resurse din configuraţie (cu excepţia cazului când resursele individuale au fost activate în mod specific şi aveau **is-managed** setat pe **true**).

5.6. Atributele Instanţelor

Scripturile unor clase de resurse (LSB nefiind una dintre acestea) pot primi parametri care determină cum se comportă şi care instanţe ale unui serviciu controlează acestea.

If your resource agent supports parameters, you can add them with the **crm_resource** command. For instance

```
# crm_resource --resource Public-IP --set-parameter ip --property-value 1.2.3.4
```

ar crea o intrare în resursă precum aceasta:

Exemplu 5.4. Un exemplu de resursă OCF cu atribute de instanţă

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
   <instance_attributes id="params-public-ip">
      <nvpair id="public-ip-addr" name="ip" value="1.2.3.4"/>
   </instance_attributes>
</primitive>
```

For an OCF resource, the result would be an environment variable called **`OCF_RESKEY_ip`** with a value of **`1.2.3.4`**.

The list of instance attributes supported by an OCF script can be found by calling the resource script with the **`meta-data`** command. The output contains an XML description of all the supported attributes, their purpose and default values.

Exemplu 5.5. Afişarea metadata pentru template-ul agentului de resursă Dummy

```
# export OCF_ROOT=/usr/lib/ocf
# $OCF_ROOT/resource.d/pacemaker/Dummy meta-data
```

```
<?xml version="1.0"?>
  <!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
  <resource-agent name="Dummy" version="0.9">
    <version>1.0</version>

    <longdesc lang="en-US">
      This is a Dummy Resource Agent. It does absolutely nothing except
      keep track of whether its running or not.
      Its purpose in life is for testing and to serve as a template for RA writers.
    </longdesc>
    <shortdesc lang="en-US">Dummy resource agent</shortdesc>

    <parameters>
      <parameter name="state" unique="1">
        <longdesc lang="en-US">
          Location to store the resource state in.
        </longdesc>
        <shortdesc lang="en-US">State file</shortdesc>
        <content type="string" default="/var/run/Dummy-{OCF_RESOURCE_INSTANCE}.state" />
      </parameter>

      <parameter name="dummy" unique="0">
        <longdesc lang="en-US">
          Dummy attribute that can be changed to cause a reload
        </longdesc>
        <shortdesc lang="en-US">Dummy attribute that can be changed to cause a reload</
shortdesc>
        <content type="string" default="blah" />
      </parameter>
    </parameters>

    <actions>
      <action name="start"        timeout="90" />
      <action name="stop"         timeout="100" />
      <action name="monitor"      timeout="20" interval="10",height="0" start-delay="0" />
      <action name="reload"       timeout="90" />
```

```
      <action name="migrate_to"   timeout="100" />
      <action name="migrate_from" timeout="90" />
      <action name="meta-data"    timeout="5" />
      <action name="validate-all" timeout="30" />
    </actions>
  </resource-agent>
```

5.7. Operaţiile Resurselor

5.7.1. Monitorizarea Resurselor pentru Defecţiuni

By default, the cluster will not ensure your resources are still healthy. To instruct the cluster to do this, you need to add a **`monitor`** operation to the resource's definition.

Exemplu 5.6. O resursă OCF cu o verificare recurentă a sănătăţii

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
  <operations>
     <op id="public-ip-check" name="monitor" interval="60s"/>
  </operations>
  <instance_attributes id="params-public-ip">
     <nvpair id="public-ip-addr" name="ip" value="1.2.3.4"/>
  </instance_attributes>
</primitive>
```

Tabel 5.3. Proprietăţile unei Operaţii

Câmp	Descriere
`id`	Your name for the action. Must be unique.
`name`	The action to perform. Common values: **`monitor`**, **`start`**, **`stop`**
`interval`	How frequently (in seconds) to perform the operation. Default value: **`0`**, meaning never.
`timeout`	How long to wait before declaring the action has failed.
`on-fail`	Acţiunea pe care să o execute dacă vreodată această acţiune eşuează. Valori permise: * *ignore* - Pretend the resource did not fail * *block* - Don't perform any further operations on the resource * *stop* - Stop the resource and do not start it elsewhere * *restart* - Stop the resource and start it again (possibly on a different node) * *fence* - STONITH the node on which the resource failed * *standby* - Move *all* resources away from the node on which the resource failed The default for the **`stop`** operation is **`fence`** when STONITH is enabled and **`block`** otherwise. All other operations default to **`stop`**.
`enabled`	If **`false`**, the operation is treated as if it does not exist. Allowed values: **`true`**, **`false`**

5.7.2. Setarea de Valori Implicite Globale pentru Operaţiuni

To set a default value for a operation option, simply add it to the **op_defaults** section with **crm_attribute**. Thus,

```
# crm_attribute --type op_defaults --attr-name timeout --attr-value 20s
```

would default each operation's **timeout** to 20 seconds. If an operation's definition also includes a value for **timeout**, then that value would be used instead (for that operation only).

5.7.2.1. Când Resursele Durează Mult Timp să Pornească/Oprească

There are a number of implicit operations that the cluster will always perform - **start**, **stop** and a non-recurring **monitor** operation (used at startup to check the resource isn't already active). If one of these is taking too long, then you can create an entry for them and simply specify a new value.

Exemplu 5.7. O resursă OCF cu intervale customizate pentru acţiunile implicite ale acesteia

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
  <operations>
     <op id="public-ip-startup" name="monitor" interval="0" timeout="90s"/>
     <op id="public-ip-start" name="start" interval="0" timeout="180s"/>
     <op id="public-ip-stop" name="stop" interval="0" timeout="15min"/>
  </operations>
  <instance_attributes id="params-public-ip">
     <nvpair id="public-ip-addr" name="ip" value="1.2.3.4"/>
  </instance_attributes>
</primitive>
```

5.7.2.2. Operaţiuni de Monitorizare Multiple

Atâta timp cât o pereche de două operaţiuni (pentru o singură resursă) nu au acelaşi nume sau acelaşi interval puteţi avea cât de multe operaţiuni de monitorizare pe cât doriţi. În acest fel puteţi realiza o verificare superficială a stării de sănătate la fiecare minut şi unele progresiv mai intense la intervale mai mari.

To tell the resource agent what kind of check to perform, you need to provide each monitor with a different value for a common parameter. The OCF standard creates a special parameter called **OCF_CHECK_LEVEL** for this purpose and dictates that it is *"made available to the resource agent without the normal **OCF_RESKEY** prefix"*.

Indiferent de ce nume alegeţi, puteţi să îl specificaţi adăugând un bloc **instance_attributes** la tag-ul op. Ţineţi cont că acest lucru este datoria fiecărui agent de resursă să verifice dacă parametrul există şi să decidă cum să îl folosească.

Exemplu 5.8. O resursă OCF cu două verificări de sănătate recurente, efectuând nivele diferite de verificări - specificate via **OCF_CHECK_LEVEL**.

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
   <operations>
      <op id="public-ip-health-60" name="monitor" interval="60">
         <instance_attributes id="params-public-ip-depth-60">
            <nvpair id="public-ip-depth-60" name="OCF_CHECK_LEVEL" value="10"/>
         </instance_attributes>
      </op>
      <op id="public-ip-health-300" name="monitor" interval="300">
         <instance_attributes id="params-public-ip-depth-300">
            <nvpair id="public-ip-depth-300" name="OCF_CHECK_LEVEL" value="20"/>
```

```
      </instance_attributes>
    </op>
  </operations>
  <instance_attributes id="params-public-ip">
      <nvpair id="public-ip-level" name="ip" value="1.2.3.4"/>
  </instance_attributes>
</primitive>
```

5.7.2.3. Dezactivarea unei Operaţiuni de Monitorizare

The easiest way to stop a recurring monitor is to just delete it. However, there can be times when you only want to disable it temporarily. In such cases, simply add **`enabled="false"`** to the operation's definition.

Exemplu 5.9. Exemplu de resursă OCF cu o verificare a sănătăţii dezactivată

```
<primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
  <operations>
    <op id="public-ip-check" name="monitor" interval="60s" enabled="false"/>
  </operations>
  <instance_attributes id="params-public-ip">
    <nvpair id="public-ip-addr" name="ip" value="1.2.3.4"/>
  </instance_attributes>
</primitive>
```

Acest lucru poate fi realizat din linia de comanda executând

```
# cibadmin -M -X '<op id="public-ip-check" enabled="false"/>'
```

Once you've done whatever you needed to do, you can then re-enable it with

Restricţiile Resurselor

Cuprins

6.1. Scoruri

Scorurile de toate tipurile sunt parte integrală din cum funcţionează clusterul. Practic orice de la mutarea unei resurse la a decide care resursă să fie oprită într-un cluster degradat este atinsă prin manipularea scorurilor în vreun fel.

Scores are calculated on a per-resource basis and any node with a negative score for a resource can’t run that resource. After calculating the scores for a resource, the cluster then chooses the node with the highest one.

6.1.1. Matematica Infinitului

`INFINITY` este definit în mod curent ca 1,000,000 şi adunarea/scăderea din acesta urmează următoarele 3 reguli de bază:

- Orice valoare + **`INFINITY`** = **`INFINITY`**
- Orice valoare - **`INFINITY`** = -**`INFINITY`**
- **`INFINITY`** - **`INFINITY`** = -**`INFINITY`**

6.2. Decizând pe Care Noduri Poate Rula o Resursă

There are two alternative strategies for specifying which nodes a resources can run on. One way is to say that by default they can run anywhere and then create location constraints for nodes that are not

allowed. The other option is to have nodes "opt-in"… to start with nothing able to run anywhere and selectively enable allowed nodes.

6.2.1. Opţiuni

Tabel 6.1. Opţiuni pentru Restricţii Simple de Locaţie

Câmp	Descriere
id	A unique name for the constraint
rsc	A resource name
node	A node's name
score	Positive values indicate the resource should run on this node. Negative values indicate the resource should not run on this node. Values of +/- **INFINITY** change "should"/"should not" to "must"/"must not".

6.2.2. Asymmetrical "Opt-In" Clusters

Pentru a crea un cluster opt-in, porniţi prin a împiedica resursele din a rula oriunde în mod implicit:

```
# crm_attribute --attr-name symmetric-cluster --attr-value false
```

Apoi începeţi să activaţi noduri. Fragmentul următor spune că serverul web preferă **sles-1**, baza de date preferă **sles-2** şi ambele pot face failover pe **sles-3** dacă nodul lor cu cea mai mare preferinţă eşuează.

Exemplu 6.1. Exemplu de restricţii de locaţie opt-in

```
<constraints>
    <rsc_location id="loc-1" rsc="Webserver" node="sles-1" score="200"/>
    <rsc_location id="loc-2" rsc="Webserver" node="sles-3" score="0"/>
    <rsc_location id="loc-3" rsc="Database" node="sles-2" score="200"/>
    <rsc_location id="loc-4" rsc="Database" node="sles-3" score="0"/>
</constraints>
```

6.2.3. Symmetrical "Opt-Out" Clusters

To create an opt-out cluster, start by allowing resources to run anywhere by default:

```
# crm_attribute --attr-name symmetric-cluster --attr-value true
```

Apoi începeţi să dezactivaţi noduri. Următorul fragment este echivalentul configuraţiei opt-in de mai sus.

Exemplu 6.2. Exemplu de restricţii de locaţie opt-out

```
<constraints>
    <rsc_location id="loc-1" rsc="Webserver" node="sles-1" score="200"/>
    <rsc_location id="loc-2-dont-run" rsc="Webserver" node="sles-2" score="-INFINITY"/>
```

```
    <rsc_location id="loc-3-dont-run" rsc="Database" node="sles-1" score="-INFINITY"/>
    <rsc_location id="loc-4" rsc="Database" node="sles-2" score="200"/>
</constraints>
```

Fie că ar trebui să alegeţi opt-in sau opt-out depinde atât de preferinţele voastre personale cât şi de structura clusterului vostru. Dacă majoritatea resurselor pot rula pe majoritatea nodurilor, atunci un aranjament opt-out va rezulta într-o configuraţie mai simplă. Pe partea cealaltă, dacă majoritatea resurselor pot rula doar pe un subset mic de noduri o configuraţie opt-in ar putea fi mai simplă.

6.2.4. Dacă Două Noduri Au Acelaşi Scor

Dacă două noduri au acelaşi scor, atunci clusterul va alege unul. Această alegere poate părea aleatorie şi ar putea să nu fie ceea ce s-a intenţionat, în schimb clusterului nu i-a fost dată suficientă informaţie pentru a şti care a fost intenţia.

Exemplu 6.3. Exemplu de două resurse care preferă două noduri în mod egal

```
<constraints>
    <rsc_location id="loc-1" rsc="Webserver" node="sles-1" score="INFINITY"/>
    <rsc_location id="loc-2" rsc="Webserver" node="sles-2" score="INFINITY"/>
    <rsc_location id="loc-3" rsc="Database" node="sles-1" score="500"/>
    <rsc_location id="loc-4" rsc="Database" node="sles-2" score="300"/>
    <rsc_location id="loc-5" rsc="Database" node="sles-2" score="200"/>
</constraints>
```

In the example above, assuming no other constraints and an inactive cluster, Webserver would probably be placed on sles-1 and Database on sles-2. It would likely have placed Webserver based on the node's uname and Database based on the desire to spread the resource load evenly across the cluster. However other factors can also be involved in more complex configurations.

6.3. Specificând Ordinea în care Resursele ar Trebui să Pornească/Oprească

The way to specify the order in which resources should start is by creating **`rsc_order`** constraints.

Tabel 6.2. Proprietăţile unei Restricţii de Ordonare

Câmp	Descriere
`id`	A unique name for the constraint
`first`	The name of a resource that must be started before the **`then`** resource is allowed to.
`then`	The name of a resource. This resource will start after the **`first`** resource.
`kind`	How to enforce the constraint. (*Since 1.1.2*) * Optional - Just a suggestion. Only applies if both resources are starting/stopping. * Mandatory - Always. If *first* is stopping or cannot be started, *then* must be stopped. * Serialize - Ensure that no two stop/start actions occur concurrently for a set of resources.

Câmp	Descriere
symmetrical	If true, which is the default, stop the resources in the reverse order. Default value: *true*

6.3.1. Ordonarea Obligatorie

Când resursa **then** nu poate rula fără ca resursa **first** să fie activă, ar trebui să folosiţi restricţii obligatorii. Pentru a specifica faptul că o restricţie este obligatorie, folosiţi scoruri mai mari decât zero. Acest lucru va asigura că resursa "then" va reacţiona atunci când resursa "first" îşi schimbă starea.

- Dacâ resursa **first** rula şi este oprită, resursa **then** va fi oprită de asemenea (dacă rulează)
- Dacă resursa **first** nu rula şi nu poate fi pornită, resursa **then** va fi oprită (dacă rulează)
- Dacă resursa **first** este (re)pornită în timp ce resursa **then** rulează, resursa **then** va fi oprită şi repornită

6.3.2. Ordonare Recomandată

Pe altă parte, când **score="0"** este specificat pentru o restricţie, restricţia este considerată opţională şi are efect doar când ambele resurse sunt oprite sau pornite. Orice modificare a stării resursei **first** nu va avea nici un efect asupra resursei **then**.

Exemplu 6.4. Exemplu de restricţie de ordonare obligatorie şi recomandată

```
<constraints>
    <rsc_order id="order-1" first="Database" then="Webserver" />
    <rsc_order id="order-2" first="IP" then="Webserver" score="0"/>
</constraints>
```

Informaţii adiţionale despre restricţiile de ordonare pot fi găsite în documentul *Ordonarea Explicată*[1]

6.4. Plasarea Resurselor Relativă la alte Resurse

When the location of one resource depends on the location of another one, we call this colocation.

There is an important side-effect of creating a colocation constraint between two resources: it affects the order in which resources are assigned to a node. If you think about it, it's somewhat obvious. You can't place A relative to B unless you know where B is. [2]

So when you are creating colocation constraints, it is important to consider whether you should colocate A with B or B with A.

Another thing to keep in mind is that, assuming A is collocated with B, the cluster will also take into account A's preferences when deciding which node to choose for B.

For a detailed look at exactly how this occurs, see the *Colocation Explained*[3] document.

[1] http://www.clusterlabs.org/mediawiki/images/d/d6/Ordering_Explained.pdf
[2] While the human brain is sophisticated enough to read the constraint in any order and choose the correct one depending on the situation, the cluster is not quite so smart. Yet.
[3] http://www.clusterlabs.org/mediawiki/images/6/61/Colocation_Explained.pdf

6.4.1. Opţiuni

Tabel 6.3. Proprietăţile unei Restricţii de Colocare

Câmp	Descriere
id	A unique name for the constraint.
rsc	The colocation source. If the constraint cannot be satisfied, the cluster may decide not to allow the resource to run at all.
with-rsc	The colocation target. The cluster will decide where to put this resource first and then decide where to put the resource in the **rsc** field.
score	Positive values indicate the resource should run on the same node. Negative values indicate the resources should not run on the same node. Values of +/- **INFINITY** change "should" to "must".

6.4.2. Plasament Obligatoriu

Mandatory placement occurs any time the constraint's score is **+INFINITY** or **-INFINITY**. In such cases, if the constraint can't be satisfied, then the **rsc** resource is not permitted to run. For **score=INFINITY**, this includes cases where the **with-rsc** resource is not active.

Dacă aveţi nevoie ca **resource1** să ruleze întotdeauna pe aceeaşi maşină ca şi **resource2**, aţi adăuga următoarea restricţie:

Un exemplu de restricţie de colocare

```
<rsc_colocation id="colocate" rsc="resource1" with-rsc="resource2" score="INFINITY"/>
```

Remember, because **INFINITY** was used, if **resource2** can't run on any of the cluster nodes (for whatever reason) then **resource1** will not be allowed to run.

Alternatively, you may want the opposite… that **resource1** cannot run on the same machine as **resource2**. In this case use **score="-INFINITY"**

Un exemplu de restricţie anti-colocare

```
<rsc_colocation id="anti-colocate" rsc="resource1" with-rsc="resource2" score="-INFINITY"/>
```

Din nou, specificând **-INFINTY**, restricţia este imutabilă. Deci singurul loc rămas pentru a rula este cel unde **resource2** este deja, atunci **resource1** nu poate rula nicăieri.

6.4.3. Plasament Recomandat

If mandatory placement is about "must" and "must not", then advisory placement is the "I'd prefer if" alternative. For constraints with scores greater than **-INFINITY** and less than **INFINITY**, the cluster will try and accommodate your wishes but may ignore them if the alternative is to stop some of the cluster resources.

Ca şi în viaţă, unde dacă destule persoane preferă ceva acest lucru devine obligatoriu, restricţiile de colocare recomandate se pot combina cu alte elemente ale configuraţiei pentru a se comporta ca şi cum ar fi obligatorii.

Un exemplu de restricţie de colocare doar recomandată

```
<rsc_colocation id="colocate-maybe" rsc="resource1" with-rsc="resource2" score="500"/>
```

6.5. Ordonarea Seturilor de Resurse

O situaţie comună este ca un administrator să creeze un lanţ de resurse ordonate, precum:

Exemplu 6.5. Un lanţ de resurse ordonate

```
<constraints>
    <rsc_order id="order-1" first="A" then="B" />
    <rsc_order id="order-2" first="B" then="C" />
    <rsc_order id="order-3" first="C" then="D" />
</constraints>
```

6.6. Set Ordonat

Fig. 6.1. Reprezentarea vizuală a ordinii de pornire a celor patru resurse pentru restricţiile de mai sus

Pentru a simplifica această situaţie, există un format alternativ pentru ordonarea restricţiilor

Exemplu 6.6. Un lanţ de resurse ordonate exprimate ca un set

```
<constraints>
    <rsc_order id="order-1">
      <resource_set id="ordered-set-example" sequential="true">
        <resource_ref id="A"/>
        <resource_ref id="B"/>
        <resource_ref id="C"/>
        <resource_ref id="D"/>
      </resource_set>
    </rsc_order>
</constraints>
```

Notă

Seturile de resurse au aceeaşi semantică de ordonare ca şi grupurile.

Exemplu 6.7. Un grup de resurse cu regulile de ordonare echivalente

```
<group id="dummy">
    <primitive id="A" .../>
    <primitive id="B" .../>
    <primitive id="C" .../>
```

```
    <primitive id="D" .../>
</group>
```

În timp ce formatul bazat pe seturi nu este mai puţin verbose, este semnificativ mai uşor de a nimeri şi menţine. Poate fi extins pentru a permite seturi ordonate de resurse (ne)ordonate. În exemplul de mai jos, **`rscA`** şi **`rscB`** pot porni ambele în paralel, la fel pot si **`rscC`** şi **`rscD`**, însă **`rscC`** şi **`rscD`** pot porni doar odată ce *ambele* **`rscA`** *şi* **`rscB`** sunt active.

Exemplu 6.8. Seturi ordonate de resurse neordonate

```
<constraints>
    <rsc_order id="order-1">
      <resource_set id="ordered-set-1" sequential="false">
        <resource_ref id="A"/>
        <resource_ref id="B"/>
      </resource_set>
      <resource_set id="ordered-set-2" sequential="false">
        <resource_ref id="C"/>
        <resource_ref id="D"/>
      </resource_set>
    </rsc_order>
  </constraints>
```

6.7. Două Seturi de Resurse Neordonate

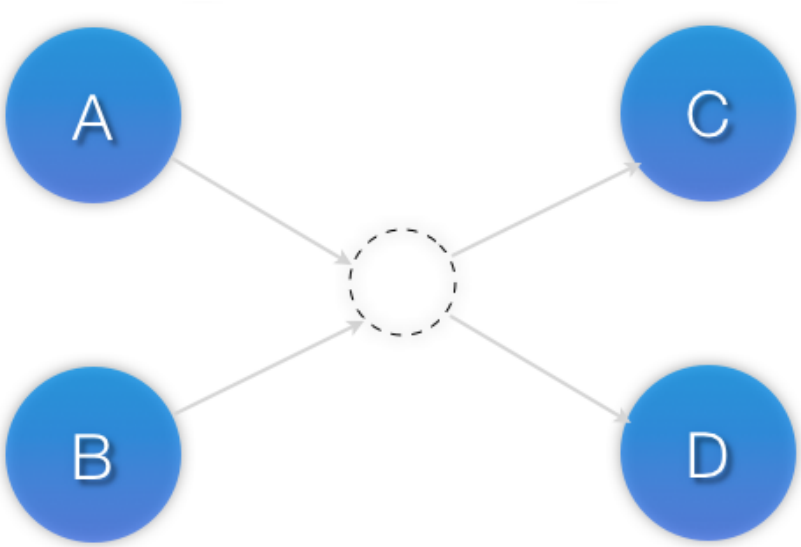


Fig. 6.2. Reprezentarea vizuală a ordinii de pornire pentru două seturi de resurse neordonate

Of course either set — or both sets — of resources can also be internally ordered (by setting **`sequential="true"`**) and there is no limit to the number of sets that can be specified.

Exemplu 6.9. Utilizări avansate ale ordonării seturilor - Trei seturi ordonate, două din care sunt neordonate intern

```
<constraints>
    <rsc_order id="order-1">
      <resource_set id="ordered-set-1" sequential="false">
        <resource_ref id="A"/>
        <resource_ref id="B"/>
```

```
      </resource_set>
      <resource_set id="ordered-set-2" sequential="true">
        <resource_ref id="C"/>
        <resource_ref id="D"/>
      </resource_set>
      <resource_set id="ordered-set-3" sequential="false">
        <resource_ref id="E"/>
        <resource_ref id="F"/>
      </resource_set>
    </rsc_order>
</constraints>
```

6.8. Trei Seturi de Resurse

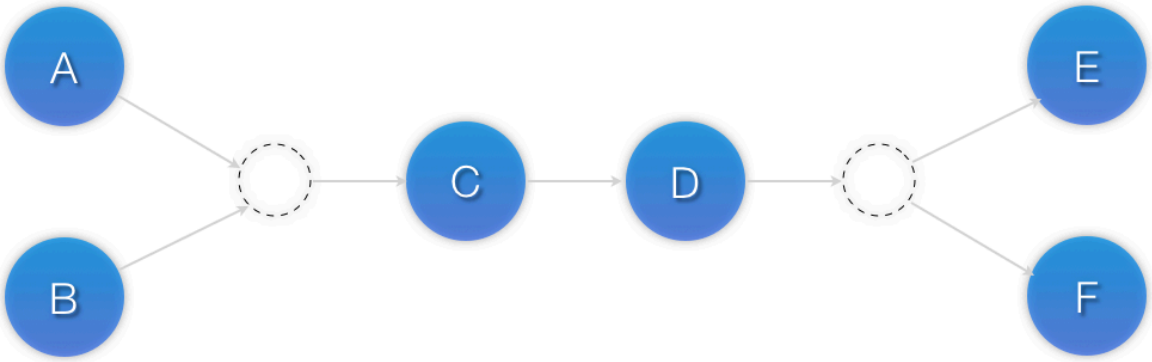


Fig. 6.3. Reprezentarea vizuală a ordinii de pornire pentru cele trei seturi definite mai sus

6.9. Colocarea Seturilor de Resurse

O altă situaţie comună este ca un administrator să creeze un set de resurse colocate. Anterior acest lucru era posibil fie prin definirea unui grup de resurse (Vedeţi *Secţiune 10.1, „Groups - A Syntactic Shortcut”*) care nu putea întotdeauna să exprime designul în mod corect; sau prin definirea fiecărei relaţii ca o restricţie individuală, cauzând o explozie de restricţii pe măsură ce numărul resurselor şi al combinaţiilor creştea.

Exemplu 6.10. Un lanţ de resurse colocate

```
<constraints>
    <rsc_colocation id="coloc-1" rsc="B" with-rsc="A" score="INFINITY"/>
    <rsc_colocation id="coloc-2" rsc="C" with-rsc="B" score="INFINITY"/>
    <rsc_colocation id="coloc-3" rsc="D" with-rsc="C" score="INFINITY"/>
</constraints>
```

To make things easier, we allow an alternate form of colocation constraints using **`resource_sets`**. Just like the expanded version, a resource that can’t be active also prevents any resource that must be collocated with it from being active. For example, if **`B was`** not able to run, then both **`C (+and by inference +D)`** must also remain stopped.

Exemplu 6.11. Lanţul echivalent de restricţii de colocare exprimat folosind **`resource_sets`**

```
<constraints>
    <rsc_colocation id="coloc-1" score="INFINITY" >
      <resource_set id="collocated-set-example" sequential="true">
        <resource_ref id="A"/>
```

```
        <resource_ref id="B"/>
        <resource_ref id="C"/>
        <resource_ref id="D"/>
      </resource_set>
    </rsc_colocation>
</constraints>
```

Notă

Seturile de resurse au aceleaşi restricţii semantice ca şi grupurile.

O resursă de grup cu regulile echivalente de colocare

```
<group id="dummy">
    <primitive id="A" .../>
    <primitive id="B" .../>
    <primitive id="C" .../>
    <primitive id="D" .../>
</group>
```

This notation can also be used in this context to tell the cluster that a set of resources must all be located with a common peer, but have no dependencies on each other. In this scenario, unlike the previous, **`B would`** be allowed to remain active even if **`A or C`** (or both) were inactive.

Exemplu 6.12. Folosirea seturilor de colocare pentru a specifica un nod comun.

```
<constraints>
    <rsc_colocation id="coloc-1" score="INFINITY" >
      <resource_set id="collocated-set-1" sequential="false">
        <resource_ref id="A"/>
        <resource_ref id="B"/>
        <resource_ref id="C"/>
      </resource_set>
      <resource_set id="collocated-set-2" sequential="true">
        <resource_ref id="D"/>
      </resource_set>
    </rsc_colocation>
</constraints>
```

Of course there is no limit to the number and size of the sets used. The only thing that matters is that in order for any member of set N to be active, all the members of set N+1 must also be active (and naturally on the same node); and if a set has **`sequential="true"`**, then in order for member M to be active, member M+1 must also be active. You can even specify the role in which the members of a set must be in using the set's role attribute.

Exemplu 6.13. Un lanţ de colocare unde membrii setului mijlociu nu au interdependenţe şi ultimul are statusul de master.

```
<constraints>
    <rsc_colocation id="coloc-1" score="INFINITY" >
      <resource_set id="collocated-set-1" sequential="true">
        <resource_ref id="A"/>
        <resource_ref id="B"/>
```

```
      </resource_set>
      <resource_set id="collocated-set-2" sequential="false">
        <resource_ref id="C"/>
        <resource_ref id="D"/>
        <resource_ref id="E"/>
      </resource_set>
      <resource_set id="collocated-set-2" sequential="true" role="Master">
        <resource_ref id="F"/>
        <resource_ref id="G"/>
      </resource_set>
    </rsc_colocation>
</constraints>
```

6.10. Încă Trei Seturi de Resurse

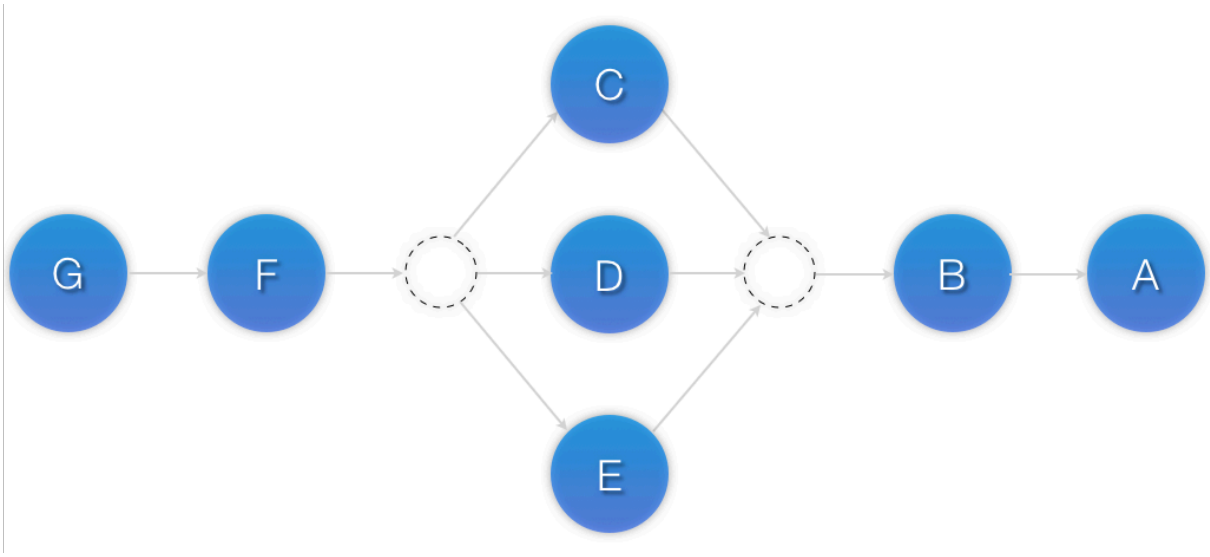


Fig. 6.4. Reprezentarea vizuală a unui lanţ de colocare unde membrii setului mijlociu nu au interdependenţe

Receiving Notification for Cluster Events

Cuprins

A Pacemaker cluster is an event driven system. In this context, an event is a resource failure or configuration change (not exhaustive).

The **`ocf:pacemaker:ClusterMon`** resource can monitor the cluster status and triggers alerts on each cluster event. This resource runs **`crm_mon`** in the background at regular intervals (configurable) and uses **`crm_mon`** capabilities to send emails (SMTP), SNMP traps or to execute an external program via the **`extra_options`** parameter.

Notă

Depending on your system settings and compilation settings, SNMP or email alerts might be unavailable. Check **`crm_mon --help`** output to see if these options are available to you. In any case, executing an external agent will always be available, and you can have this agent to send emails, SNMP traps, or whatever action you develop.

7.1. Configuring SNMP Notifications

Requires an IP to send SNMP traps to, and a SNMP community. Pacemaker MIB is found in */usr/share/snmp/mibs/PCMK-MIB.txt*

Exemplu 7.1. Configuring ClusterMon to send SNMP traps

```
<clone id="ClusterMon-clone">
    <primitive class="ocf" id="ClusterMon-SNMP" provider="pacemaker" type="ClusterMon">
        <instance_attributes id="ClusterMon-instance_attributes">
            <nvpair id="ClusterMon-instance_attributes-user" name="user" value="root"/>
            <nvpair id="ClusterMon-instance_attributes-update" name="update" value="30"/>
            <nvpair id="ClusterMon-instance_attributes-extra_options" name="extra_options"
 value="-S snmphost.example.com -C public"/>
        </instance_attributes>
    </primitive>
</clone>
```

7.2. Configuring Email Notifications

Requires a user to send mail alerts to. "Mail-From", SMTP relay and Subject prefix can also be configured.

Exemplu 7.2. Configuring ClusterMon to send email alerts

```
<clone id="ClusterMon-clone">
    <primitive class="ocf" id="ClusterMon-SMTP" provider="pacemaker" type="ClusterMon">
        <instance_attributes id="ClusterMon-instance_attributes">
            <nvpair id="ClusterMon-instance_attributes-user" name="user" value="root"/>
            <nvpair id="ClusterMon-instance_attributes-update" name="update" value="30"/>
            <nvpair id="ClusterMon-instance_attributes-extra_options" name="extra_options"
 value="-T pacemaker@example.com -F pacemaker@node2.example.com -P PACEMAKER -H
 mail.example.com"/>
        </instance_attributes>
    </primitive>
</clone>
```

7.3. Configuring Notifications via External-Agent

Requires a program (external-agent) to run when resource operations take place, and an external-recipient (IP address, Email address, URI). When triggered, the external-agent is fed with dynamically filled environnement variables describing precisely the cluster event that occurred. By making smart usage of these variables in your external-agent code, you can trigger any action.

Exemplu 7.3. Configuring ClusterMon to execute an external-agent

```
<clone id="ClusterMon-clone">
    <primitive class="ocf" id="ClusterMon" provider="pacemaker" type="ClusterMon">
        <instance_attributes id="ClusterMon-instance_attributes">
            <nvpair id="ClusterMon-instance_attributes-user" name="user" value="root"/>
            <nvpair id="ClusterMon-instance_attributes-update" name="update" value="30"/>
            <nvpair id="ClusterMon-instance_attributes-extra_options" name="extra_options"
 value="-E /usr/local/bin/example.sh -e 192.168.12.1"/>
        </instance_attributes>
    </primitive>
</clone>
```

Tabel 7.1. Environment Variables Passed to the External Agent

Environment Variable	Description
CRM_notify_recipient	The static external-recipient from the resource definition.
CRM_notify_node	The node on which the status change happened.
CRM_notify_rsc	The name of the resource that changed the status.
CRM_notify_task	The operation that caused the status change.
CRM_notify_desc	The textual output relevant error code of the operation (if any) that caused the status change.
CRM_notify_rc	The return code of the operation.
CRM_notify_target_rc	The expected return code of the operation.
CRM_notify_status	The numerical representation of the status of the operation.

Reguli

Cuprins

Regulile pot fi folosite pentru a face configuraţia voastră mai dinamică. Un exemplu comun este de a seta o valoare pentru **resource-stickiness** în timpul orelor de program, pentru a împiedica resursele de a fi mutate înapoi la locaţia cea mai preferată a acestora, iar altă valoare în weekend-uri când nu este nimeni în preajmă să detecteze o întrerupere a serviciului.

O altă utilizare a regulilor ar putea fi pentru a asigna maşini la grupuri de procesare diferite (folosind un atribut de nod) bazat pe timp şi mai apoi să folosească acel atribut pentru a crea o restricţie de locaţie.

Each rule can contain a number of expressions, date-expressions and even other rules. The results of the expressions are combined based on the rule's **boolean-op** field to determine if the rule ultimately evaluates to **true** or **false**. What happens next depends on the context in which the rule is being used.

Tabel 8.1. Proprietăţile unei Reguli

Câmp	Descriere
role	Limits the rule to apply only when the resource is in that role. Allowed values: *Started*, **Slave,** and **Master**. NOTE: A rule with **role="Master"** can not determine the initial location of a clone instance. It will only affect which of the active instances will be promoted.
score	The score to apply if the rule evaluates to **true**. Limited to use in rules that are part of location constraints.
score-attribute	The node attribute to look up and use as a score if the rule evaluates to **true**. Limited to use in rules that are part of location constraints.
boolean-op	How to combine the result of multiple expression objects. Allowed values: *and* and **or**.

8.1. Node Attribute Expressions

Obiectele de expresie sunt folosite pentru a controla o resursă pe baza atributelor definite de un nod sau de mai multe noduri. Adiţional la orice atribute adăugate de administrator, fiecare nod are un atribut de nod predefinit numit **#uname** care poate fi folosit de asemenea.

Tabel 8.2. Proprietăţile unei Expresii

Câmp	Descriere
value	User supplied value for comparison
attribute	The node attribute to test
type	Determines how the value(s) should be tested. Allowed values: *string*, **integer**, **version**
operation	Comparaţia pe care să o efectueze. Valori permise: * *lt* - True if the node attribute's value is less than **value** * *gt* - True if the node attribute's value is greater than **value** * *lte* - True if the node attribute's value is less than or equal to **value** * *gte* - True if the node attribute's value is greater than or equal to **value** * *eq* - True if the node attribute's value is equal to **value** * *ne* - True if the node attribute's value is not equal to **value** * *defined* - True if the node has the named attribute * *not_defined* - True if the node does not have the named attribute

8.2. Time/Date Based Expressions

După cum sugerează numele, **date_expressions** sunt folosite pentru a controla o resursă sau opţiune a clusterului pe baza timpului/dăţii curente. Acestea pot conţine opţional obiecte **date_spec** şi/sau **duration** în funcţie de context.

Tabel 8.3. Proprietăţile unei Expresii de Dată

Câmp	Descriere
start	A date/time conforming to the ISO8601 specification.
end	A date/time conforming to the ISO8601 specification. Can be inferred by supplying a value for **start** and a **duration**.
operation	Compară data/ora curentă cu data de început şi/sau sfârşit, în funcţie de context. Valori permise: * *gt* - True if the current date/time is after **start** * *lt* - True if the current date/time is before **end** * *in-range* - True if the current date/time is after **start** and before **end** * *date-spec* - performs a cron-like comparison to the current date/time

Notă

As these comparisons (except for **date_spec**) include the time, the **eq**, **neq**, **gte** and **lte** operators have not been implemented since they would only be valid for a single second.

8.2.1. Date Specifications

Obiectele **date_spec** sunt folosite pentru a crea expresii similare cu cele create de cron în relaţie cu timpul. Fiecare câmp poate conţine un singur număr sau un singur şir. În loc să aibe valoarea implicită zero, orice câmp a cărui valoare nu este furnizată este ignorat.

De exemplu, **monthdays="1"** se potriveşte pentru prima zi din fiecare lună şi **hours="09-17"** se potriveşte orelor între 9am şi 5pm (inclusiv). Însă la acest moment nu se poate specifica **weekdays="1,2"** sau **weekdays="1-2,5-6"** din moment ce conţin şiruri multiple. În funcţie de cerere, acest aspect ar putea fi implementat într-un release viitor.

Tabel 8.4. Proprietăţile unei Specificaţii de Dată

Câmp	Descriere
id	A unique name for the date
hours	Allowed values: 0-23
monthdays	Allowed values: 0-31 (depending on month and year)
weekdays	Allowed values: 1-7 (1=Monday, 7=Sunday)
yeardays	Allowed values: 1-366 (depending on the year)
months	Allowed values: 1-12
weeks	Allowed values: 1-53 (depending on weekyear)
years	Year according the Gregorian calendar
weekyears	May differ from Gregorian years; Eg. **2005-001 Ordinal** is also **2005-01-01 Gregorian** is also **2004-W53-6 Weekly**
moon	Allowed values: 0-7 (0 is new, 4 is full moon). Seriously, you can use this. This was implemented to demonstrate the ease with which new comparisons could be added.

8.2.2. Durations

Duratele sunt folosite pentru a calcula valoarea pentru **end** atunci când una nu este furnizată în operaţiunile in_range. Acestea conţin aceleaşi câmpuri ca şi obiectele **date_spec** dar fără limitări (ex. puteţi avea o durată de 19 zile). Ca şi în cazul **date_specs**, orice câmp care nu este furnizat este ignorat.

8.3. Exemple de Expresii Bazate pe Timp

A small sample of how time based expressions can be used.

Exemplu 8.1. Adevărat dacă acum este oricând în anul 2005

```
<rule id="rule1">
   <date_expression id="date_expr1" start="2005-001" operation="in_range">
    <duration years="1"/>
   </date_expression>
</rule>
```

Exemplu 8.2. Equivalent expression

```
<rule id="rule2">
   <date_expression id="date_expr2" operation="date_spec">
    <date_spec years="2005"/>
   </date_expression>
</rule>
```

Exemplu 8.3. 9am-5pm, Lun-Vineri

```
<rule id="rule3">
   <date_expression id="date_expr3" operation="date_spec">
    <date_spec hours="9-16" days="1-5"/>
   </date_expression>
</rule>
```

Vă rugăm să luați aminte că **16** se potrivește cu **16:59:59**, deoarece valoarea numerică (ora) încă se potrivește!

Exemplu 8.4. 9am-6pm, Lun-Vineri sau toată ziua sâmbătă

```
<rule id="rule4" boolean_op="or">
   <date_expression id="date_expr4-1" operation="date_spec">
    <date_spec hours="9-16" days="1-5"/>
   </date_expression>
   <date_expression id="date_expr4-2" operation="date_spec">
    <date_spec days="6"/>
   </date_expression>
</rule>
```

Exemplu 8.5. 9am-5pm sau 9pm-12pm, Lun-Vineri

```
<rule id="rule5" boolean_op="and">
   <rule id="rule5-nested1" boolean_op="or">
    <date_expression id="date_expr5-1" operation="date_spec">
     <date_spec hours="9-16"/>
    </date_expression>
    <date_expression id="date_expr5-2" operation="date_spec">
     <date_spec hours="21-23"/>
    </date_expression>
   </rule>
   <date_expression id="date_expr5-3" operation="date_spec">
    <date_spec days="1-5"/>
   </date_expression>
  </rule>
```

Exemplu 8.6. Zilele de Luni în Martie 2005

```
<rule id="rule6" boolean_op="and">
   <date_expression id="date_expr6-1" operation="date_spec">
    <date_spec weekdays="1"/>
   </date_expression>
   <date_expression id="date_expr6-2" operation="in_range"
     start="2005-03-01" end="2005-04-01"/>
  </rule>
```

Notă

Because no time is specified, 00:00:00 is implied.

This means that the range includes all of 2005-03-01 but none of 2005-04-01. You may wish to write **`end="2005-03-31T23:59:59"`** to avoid confusion.

Exemplu 8.7. O lună plină pe data de Vineri 13

```
<rule id="rule7" boolean_op="and">
   <date_expression id="date_expr7" operation="date_spec">
    <date_spec weekdays="5" monthdays="13" moon="4"/>
   </date_expression>
</rule>
```

8.4. Using Rules to Determine Resource Location

If the constraint's outer-most rule evaluates to **`false`**, the cluster treats the constraint as if it was not there. When the rule evaluates to **`true`**, the node's preference for running the resource is updated with the score associated with the rule.

Dacă acest lucru sună cunoscut, este datorită faptului că deja folosiţi o sintaxă simplificată pentru regulile de restricţie a locaţiei. Consideraţi următoarea restricţie de locaţie:

Exemplu 8.8. Împiedică myApacheRsc de a rula pe c001n03

```
<rsc_location id="dont-run-apache-on-c001n03" rsc="myApacheRsc"
              score="-INFINITY" node="c001n03"/>
```

Această restricţie poate fi scrisă mai elaborat ca:

Exemplu 8.9. Împiedică myApacheRsc de a rula pe c001n03 - versiunea extinsă

```
<rsc_location id="dont-run-apache-on-c001n03" rsc="myApacheRsc">
    <rule id="dont-run-apache-rule" score="-INFINITY">
      <expression id="dont-run-apache-expr" attribute="#uname"
        operation="eq" value="c00n03"/>
    </rule>
```

```
</rsc_location>
```

Avantajul folosirii formei extinse este că se pot adăuga clauze suplimentare la regulă, cum ar fi limitarea regulii astfel încât să se aplice doar în momente specifice ale zilei sau zilelor săptămânii (acest lucru este discutat în secţiunile următoare).

It also allows us to match on node properties other than its name. If we rated each machine's CPU power such that the cluster had the following nodes section:

Exemplu 8.10. Un exemplu de secţiune de noduri pentru utilizarea cu score-attribute

```
<nodes>
   <node id="uuid1" uname="c001n01" type="normal">
      <instance_attributes id="uuid1-custom_attrs">
        <nvpair id="uuid1-cpu_mips" name="cpu_mips" value="1234"/>
      </instance_attributes>
   </node>
   <node id="uuid2" uname="c001n02" type="normal">
      <instance_attributes id="uuid2-custom_attrs">
        <nvpair id="uuid2-cpu_mips" name="cpu_mips" value="5678"/>
      </instance_attributes>
   </node>
</nodes>
```

atunci am putea împiedica resursele de a rula pe maşini fără nivelul de procesare cerut cu această regulă

```
<rule id="need-more-power-rule" score="-INFINITY">
   <expression id=" need-more-power-expr" attribute="cpu_mips"
               operation="lt" value="3000"/>
</rule>
```

8.4.1. Folosind `score-attribute` în loc de `score`

Când folosim **`score-attribute`** în loc de **`score`**, fiecare nod care se potriveşte regulii va avea scorul acestuia ajustat în mod diferit, în funcţie de valoarea sa pentru atributul numit de nod. Prin urmare în exemplul anterior, dacă o regulă folosea **`score-attribute="cpu_mips"`**, **`c001n01`** ar avea preferinţa proprie de a rula resursa crescută cu **`1234`** în timp ce **`c001n02`** ar avea preferinţa crescută cu **`5678`**.

8.5. Folosind Reguli pentru a Controla Opţiunile Resurselor

Often some cluster nodes will be different from their peers; sometimes these differences (the location of a binary or the names of network interfaces) require resources to be configured differently depending on the machine they're hosted on.

Prin definirea de obiecte multiple **`instance_attributes`** pentru resursă şi prin adăugarea unei reguli pentru fiecare, putem gestiona facil aceste cazuri speciale.

În exemplul de mai jos, **`mySpecialRsc`** va folosi eth1 şi portul 9999 când va rula pe **`node1`**, eth2 şi portul 8888 pe **`node2`** şi va avea valorea implicită eth0 şi portul 9999 pentru toate celelalte noduri.

Exemplu 8.11. Definirea de opţiuni de resursă diferite pe baza numelui nodului

```
<primitive id="mySpecialRsc" class="ocf" type="Special" provider="me">
```

```
   <instance_attributes id="special-node1" score="3">
    <rule id="node1-special-case" score="INFINITY" >
     <expression id="node1-special-case-expr" attribute="#uname"
       operation="eq" value="node1"/>
    </rule>
    <nvpair id="node1-interface" name="interface" value="eth1"/>
   </instance_attributes>
   <instance_attributes id="special-node2" score="2" >
    <rule id="node2-special-case" score="INFINITY">
     <expression id="node2-special-case-expr" attribute="#uname"
       operation="eq" value="node2"/>
    </rule>
    <nvpair id="node2-interface" name="interface" value="eth2"/>
    <nvpair id="node2-port" name="port" value="8888"/>
   </instance_attributes>
   <instance_attributes id="defaults" score="1" >
    <nvpair id="default-interface" name="interface" value="eth0"/>
    <nvpair id="default-port" name="port" value="9999"/>
   </instance_attributes>
</primitive>
```

Ordinea în care obiectele **`instance_attributes`** sunt evaluate este determinată de către scorul acestora (cel mai mare la cel mai mic). Dacă nu este furnizat, scorul va avea valoarea implicită zero şi obiectele cu un scor egal sunt procesate în ordinea listată. Dacă obiectul **`instance_attributes`** nu are o **`rule`** sau are o **`rule`** care este evaluată la **`true`**, atunci pentru orice parametru pentru care resursa nu are încă o valoare, resursa va folosi valorile parametrilor definite de obiectul **`instance_attributes`**.

8.6. Using Rules to Control Cluster Options

Controlarea opţiunilor clusterului este reuşită în mare parte prin aceeaşi manieră ca şi specificarea diverselor opţiuni pentru resurse pe noduri diferite.

The difference is that because they are cluster options, one cannot (or should not, because they won't work) use attribute based expressions. The following example illustrates how to set a different **`resource-stickiness`** value during and outside of work hours. This allows resources to automatically move back to their most preferred hosts, but at a time that (in theory) does not interfere with business activities.

Exemplu 8.12. Schimbarea **`resource-stickiness`** în timpul orelor de lucru

```
<rsc_defaults>
   <meta_attributes id="core-hours" score="2">
      <rule id="core-hour-rule" score="0">
        <date_expression id="nine-to-five-Mon-to-Fri" operation="date_spec">
          <date_spec id="nine-to-five-Mon-to-Fri-spec" hours="9-16" weekdays="1-5"/>
        </date_expression>
      </rule>
      <nvpair id="core-stickiness" name="resource-stickiness" value="INFINITY"/>
   </meta_attributes>
   <meta_attributes id="after-hours" score="1" >
      <nvpair id="after-stickiness" name="resource-stickiness" value="0"/>
   </meta_attributes>
</rsc_defaults>
```

8.7. Asigurarea că Regulile Bazate pe Timp Iau Efect

A Pacemaker cluster is an event driven system. As such, it won't recalculate the best place for resources to run in unless something (like a resource failure or configuration change) happens. This can mean that a location constraint that only allows resource X to run between 9am and 5pm is not enforced.

Dacă vă bazaţi pe reguli bazate pe timp, este esenţial să setaţi opţiunea **`cluster-recheck-interval`**. Aceasta spune clusterului să recalculeze periodic starea ideală a clusterului. De exemplu, dacă setaţi **`cluster-recheck-interval=5m`**, atunci cândva între 9:00 şi 9:05 clusterul va detecta că trebuie să pornească resursa X, iar între 17:00 şi 17:05 va realiza că trebuie să o oprească.

Note that the timing of the actual start and stop actions depends on what else needs to be performed first .

Configuraţii Avansate

Cuprins

9.1. Conectarea de pe o Maşină la Distanţă

Provided Pacemaker is installed on a machine, it is possible to connect to the cluster even if the machine itself is not in the same cluster. To do this, one simply sets up a number of environment variables and runs the same commands as when working on a cluster node.

Tabel 9.1. Variabile de Mediu Folosite pentru Conectare la Instanţe la Distanţă ale CIB-ului

Environment Variable	Descriere
CIB_user	The user to connect as. Needs to be part of the **hacluster** group on the target host. Defaults to *$USER*.
CIB_passwd	The user's password. Read from the command line if unset.
CIB_server	The host to contact. Defaults to *localhost*.
CIB_port	The port on which to contact the server; required.
CIB_encrypted	Encrypt network traffic; defaults to *true*.

So, if **c001n01** is an active cluster node and is listening on **1234** for connections, and **someguy** is a member of the **hacluster** group, then the following would prompt for **someguy**'s password and return the cluster's current configuration:

```
# export CIB_port=1234; export CIB_server=c001n01; export CIB_user=someguy;
# cibadmin -Q
```

Din motive de securitate, clusterul nu ascultă pentru conexiuni la distanţă în mod implicit. Dacă doriţi să permiteţi accesul de la distanţă, trebuie să setaţi opţiunile primare **remote-tls-port** (criptat) sau **remote-clear-port** (necriptat) (ex. acelea stocate în tag-ul cib, precum **num_updates** şi **epoch**).

Tabel 9.2. Extra top-level CIB options for remote access

Câmp	Descriere
remote-tls-port	Listen for encrypted remote connections on this port. Default: *none*
remote-clear-port	Listen for plaintext remote connections on this port. Default: *none*

9.2. Specificând Când Acţiunile Recurente sunt Efectuate

În mod implicit, acţiunile recurente sunt programate în mod relativ faţă de când a fost pornită resursa. Deci dacă resursa voastră a fost pornită la 14:32 şi aveţi un backup setat să fie executat la fiecare 24 de ore, atunci backup-ul va rula întotdeauna în mijlocul zilei de lucru - deloc de dorit.

To specify a date/time that the operation should be relative to, set the operation's **`interval-origin`**. The cluster uses this point to calculate the correct **`start-delay`** such that the operation will occur at *origin + (interval * N)*.

So, if the operation's interval is 24h, it's interval-origin is set to **`02:00`** and it is currently **`14:32`**, then the cluster would initiate the operation with a start delay of 11 hours and 28 minutes. If the resource is moved to another node before 2am, then the operation is of course cancelled.

Valoarea specificată pentru interval şi **`interval-origin`** poate fi orice dată/timp ce se conformează cu *standardul ISO8601*[1]. Spre exemplu, pentru a specifica o operaţiune care ar rula în prima zi de Luni din 2009 şi în fiecare zi de Luni de atunci înainte aţi adauga:

Exemplu 9.1. Specificând o Bază pentru Intervalele Acţiunilor Recurente

```
<op id="my-weekly-action" name="custom-action" interval="P7D" interval-origin="2009-
W01-1"/>
```

9.3. Mutarea Resurselor

9.3.1. Intervenţie Manuală

There are primarily two occasions when you would want to move a resource from it's current location: when the whole node is under maintenance, and when a single resource needs to be moved.

În cazul în care totul trebuie mutat, din moment ce totul ajunge eventual să ţină de un scor, aţi putea crea restricţii pentru fiecare resursă pe care o aveţi împiedicând-o din a mai rula pe acel nod. În timp ce configuraţia poate părea complicată în anumite momente, nici chiar noi nu am solicita acest lucru de la administratori.

Instead one can set a special node attribute which tells the cluster "don't let anything run here". There is even a helpful tool to help query and set it, called **`crm_standby`**. To check the standby status of the current machine, simply run:

```
# crm_standby --get-value
```

O valoarea de **`true`** indică faptul că nodul *NU* poate găzdui nici un fel de resurse, în timp ce o valoare de **`false`** spune că acesta *POATE*.

You can also check the status of other nodes in the cluster by specifying the **`--node-uname`** option:

```
# crm_standby --get-value --node-uname sles-2
```

To change the current node's standby status, use **`--attr-value`** instead of **`--get-value`**.

[1] http://en.wikipedia.org/wiki/ISO_8601

```
# crm_standby --attr-value
```

Again, you can change another host's value by supplying a host name with **--node-uname**.

When only one resource is required to move, we do this by creating location constraints. However, once again we provide a user friendly shortcut as part of the **crm_resource** command, which creates and modifies the extra constraints for you. If **Email** was running on **sles-1** and you wanted it moved to a specific location, the command would look something like:

```
# crm_resource -M -r Email -H sles-2
```

În culise, utilitarul va crea următoarea restricţie de locaţie:

```
<rsc_location rsc="Email" node="sles-2" score="INFINITY"/>
```

It is important to note that subsequent invocations of **crm_resource -M** are not cumulative. So, if you ran these commands

```
# crm_resource -M -r Email -H sles-2
# crm_resource -M -r Email -H sles-3
```

atunci ar fi ca şi când nu aţi fi efectuat niciodată prima comandă.

Pentru a permite resursei să se mute înapoi, folosiţi:

```
# crm_resource -U -r Email
```

Note the use of the word *allow*. The resource can move back to its original location but, depending on **resource-stickiness**, it might stay where it is. To be absolutely certain that it moves back to **sles-1**, move it there before issuing the call to **crm_resource -U**:

```
# crm_resource -M -r Email -H sles-1
# crm_resource -U -r Email
```

Ca alternativă, dacă vă pasă doar că resursa ar trebui să fie mutată din locaţia ei curentă, încercaţi

```
# crm_resource -M -r Email`
```

Care va crea în schimb o restricţie negativă, precum

```
<rsc_location rsc="Email" node="sles-1" score="-INFINITY"/>
```

This will achieve the desired effect, but will also have long-term consequences. As the tool will warn you, the creation of a **-INFINITY** constraint will prevent the resource from running on that node until **crm_resource -U** is used. This includes the situation where every other cluster node is no longer available!

În anumite cazuri, precum cel în care **resource-stickiness** este setată la **INFINITY**, este posibil să ajungeţi la problema descrisă în *Secțiune 6.2.4, „Dacă Două Noduri Au Acelaşi Scor”*. Utilitarul poate detecta unele din aceste cazuri şi le tratează prin crearea atât de restricţii pozitive cât şi negative. Ex.

Email preferă **sles-1** cu un scor de **-INFINITY**

Email preferă **sles-2** cu un scor de **INFINITY**

care are aceleaşi consecinţe pe termen lung precum am discutat anterior.

9.3.2. Mutarea Resurselor Datorită Eşecului

New in 1.0 is the concept of a migration threshold. [2]

Simply define **`migration-threshold=N`** for a resource and it will migrate to a new node after N failures. There is no threshold defined by default. To determine the resource's current failure status and limits, use **`crm_mon --failcounts`**.

By default, once the threshold has been reached, this node will no longer be allowed to run the failed resource until the administrator manually resets the resource's failcount using **`crm_failcount`** (after hopefully first fixing the failure's cause). However it is possible to expire them by setting the resource's **`failure-timeout`** option.

Prin urmare setarea **`migration-threshold=2`** şi **`failure-timeout=60s`** ar conduce resursa la mutarea pe un nod nou după 2 eşecuri şi potenţial îi va permite să se mute înapoi (în funcţie de scorurile de adezivitate şi restrictie) după un minut.

Sunt două excepţii la conceptul de prag de migrare şi se întâmplă atunci când o resursă fie eşuează să pornească sau eşuează să se oprească. Eşecurile de pornire fac failcount-ul să fie setat la **`INFINITY`** şi prin urmare provoacă mutarea imediată a resursei.

Eşecurile la oprire sunt puţin diferite şi cruciale. Dacă o resursă eşuează de a se opri şi STONITH este activat, atunci clusterul va evacua nodul pentru a putea să pornească resursa în altă parte. Dacă STONITH nu este activat, atunci clusterul nu are nici o mod de a continua şi nu va încerca să pornească resursa în altă parte, dar va încerca să o oprească din nou după ce se depaşeşte timpul limită al eşecului.

Important

Vă rugăm să citiţi *Secţiune 8.7, „Asigurarea că Regulile Bazate pe Timp Iau Efect”* înainte de activarea acestei opţiuni.

9.3.3. Mutarea Resurselor Din Cauza Schimbărilor de Conectivitate

Setarea clusterului pentru a muta resursele când conectivitatea externă este pierdută, este un proces în doi paşi.

9.3.3.1. Spuneţi Pacemaker-ului să monitorizeze conectivitatea

To do this, you need to add a **`ping`** resource to the cluster. The **`ping`** resource uses the system utility of the same name to a test if list of machines (specified by DNS hostname or IPv4/IPv6 address) are reachable and uses the results to maintain a node attribute normally called **`pingd`**. [3]

[2] The naming of this option was perhaps unfortunate as it is easily confused with true migration, the process of moving a resource from one node to another without stopping it. Xen virtual guests are the most common example of resources that can be migrated in this manner.

[3] The attribute name is customizable; that allows multiple ping groups to be defined.

Notă

Older versions of Heartbeat required users to add ping nodes to *ha.cf* - this is no longer required.

Important

Older versions of Pacemaker used a custom binary called *pingd* for this functionality; this is now deprecated in favor of *ping*.

If your version of Pacemaker does not contain the ping agent, you can download the latest version from *https://github.com/ClusterLabs/pacemaker/tree/master/extra/resources/ping*

Normally the resource will run on all cluster nodes, which means that you'll need to create a clone. A template for this can be found below along with a description of the most interesting parameters.

Tabel 9.3. Common Options for a *ping* Resource

Câmp	Descriere
dampen	The time to wait (dampening) for further changes to occur. Use this to prevent a resource from bouncing around the cluster when cluster nodes notice the loss of connectivity at slightly different times.
multiplier	The number of connected ping nodes gets multiplied by this value to get a score. Useful when there are multiple ping nodes configured.
host_list	The machines to contact in order to determine the current connectivity status. Allowed values include resolvable DNS host names, IPv4 and IPv6 addresses.

Exemplu 9.2. An example ping cluster resource that checks node connectivity once every minute

```
<clone id="Connected">
   <primitive id="ping" provider="pacemaker" class="ocf" type="ping">
    <instance_attributes id="ping-attrs">
      <nvpair id="pingd-dampen" name="dampen" value="5s"/>
      <nvpair id="pingd-multiplier" name="multiplier" value="1000"/>
      <nvpair id="pingd-hosts" name="host_list" value="my.gateway.com www.bigcorp.com"/>
    </instance_attributes>
    <operations>
      <op id="ping-monitor-60s" interval="60s" name="monitor"/>
    </operations>
   </primitive>
</clone>
```

Important

You're only half done. The next section deals with telling Pacemaker how to deal with the connectivity status that **`ocf:pacemaker:ping`** is recording.

9.3.3.2. Spuneţi Pacemaker-ului cum să interpreteze datele de conectivitate

Notă

Before reading the following, please make sure you have read and understood *Chapter 8, Rules* above.

There are a number of ways to use the connectivity data provided by Heartbeat. The most common setup is for people to have a single ping node, to prevent the cluster from running a resource on any unconnected node.

Exemplu 9.3. Don't run on unconnected nodes

```
<rsc_location id="WebServer-no-connectivity" rsc="Webserver">
   <rule id="ping-exclude-rule" score="-INFINITY" >
    <expression id="ping-exclude" attribute="pingd" operation="not_defined"/>
   </rule>
</rsc_location>
```

Un setup mai complex este acela de a avea un număr de noduri de ping configurate. Poţi solicita clusterului să ruleze resursele pe nodurile care se pot conecta la toate (sau doar un subset din) acestea

Exemplu 9.4. Run only on nodes connected to three or more ping nodes; this assumes **`multiplier`** is set to 1000:

```
<rsc_location id="WebServer-connectivity" rsc="Webserver">
   <rule id="ping-prefer-rule" score="-INFINITY" >
    <expression id="ping-prefer" attribute="pingd" operation="lt" value="3000"/>
   </rule>
</rsc_location>
```

Instead you can tell the cluster only to *prefer* nodes with the best connectivity. Just be sure to set **`multiplier`** to a value higher than that of **`resource-stickiness`** (and don't set either of them to **`INFINITY`**).

Exemplu 9.5. Preferă nodul cu cele mai multe noduri de ping conectate

```
<rsc_location id="WebServer-connectivity" rsc="Webserver">
```

```
   <rule id="ping-prefer-rule" score-attribute="pingd" >
    <expression id="ping-prefer" attribute="pingd" operation="defined"/>
   </rule>
</rsc_location>
```

Este probabil mai simplu să vă gândiţi la acest lucru în termenii simplelor restricţii în care clusterul traduce acest lucru. De exemplu, dacă **`sles-1`** este conectat la toate cele 5 noduri de ping dar **`sles-2`** este conectat doar la 2, atunci ar fi la fel ca şi când aţi avea următoarele restricţii în configuraţia voastră:

Exemplu 9.6. Cum traduce clusterul restricţia pingd

```
<rsc_location id="ping-1" rsc="Webserver" node="sles-1" score="5000"/>
<rsc_location id="ping-2" rsc="Webserver" node="sles-2" score="2000"/>
```

The advantage is that you don’t have to manually update any constraints whenever your network connectivity changes.

Puteţi de asemenea să combinaţi conceptele de mai sus în ceva chiar mai complex de atât. Exemplul de mai jos arată cum puteţi prefera nodul cu cele mai multe noduri de ping conectate cu cerinţa că acestea trebuie să aibe conectivitate la minim trei (presupunând că **`multiplicatorul`** este setat la 1000).

Exemplu 9.7. Un exemplu mai complex pentru alegerea locaţiei pe baza conectivităţii

```
<rsc_location id="WebServer-connectivity" rsc="Webserver">
   <rule id="ping-exclude-rule" score="-INFINITY" >
    <expression id="ping-exclude" attribute="pingd" operation="lt" value="3000"/>
   </rule>
   <rule id="ping-prefer-rule" score-attribute="pingd" >
    <expression id="ping-prefer" attribute="pingd" operation="defined"/>
   </rule>
</rsc_location>
```

9.3.4. Migrarea Resurselor

Unele resurse, precum oaspeţii virtuali de Xen, sunt capabili să se mute în altă locaţie fără pierderea stării. Numim acest lucru migrarea resurselor şi este diferit de practica normală a opririi resurselor pe prima maşină şi apoi pornirea acestora în altă parte.

Not all resources are able to migrate, see the Migration Checklist below, and those that can, won’t do so in all situations. Conceptually there are two requirements from which the other prerequisites follow:

- resursa trebuie să fie activă şi sănătoasă în locaţia veche
- tot ce este necesar pentru ca resursa să funcţioneze trebuie să fie disponibil atât la vechea cât şi la noua locaţie

Clusterul este capabil să acomodeze atât modele de migrare de tip push cât şi pull prin solicitarea ca agentul de resursă să suporte două noi acţiuni: **`migrate_to`** (efectuată pe locaţia curentă) şi **`migrate_from`** (efectuată pe destinaţie).

În migrarea de tip push, procesul de pe locaţia curentă se transferă pe locaţia nouă unde este activat mai târziu. În acest scenariu, majoritatea muncii ar fi realizată în acţiunea **`migrate_to`** şi activarea ar avea loc în timpul **`migrate_from`**.

În egală măsură pentru pull, acţiunea **migrate_to** este practic fără conţinut şi **migrate_from** realizează majoritatea muncii, extrăgând starea relevantă a resursei de la locaţia veche şi activând-o.

Nu există un mod greşit sau corect de a implementa migrarea serviciului vostru, atâta timp cât funcţionează.

9.3.4.1. Lista de Migrare

- Resursa nu poate fi o clonă.
- Resursa trebuie să folosească un agent de stilul OCF.
- Resursa nu trebuie să fie într-o stare degradată sau să fi eşuat.
- The resource must not, directly or indirectly, depend on any primitive or group resources.
- Resursa trebuie să suporte două noi acţiuni: **migrate_to** şi **migrate_from** şi trebuie să le anunţe în meta-informaţiile proprii.
- Resursa trebuie să aibe meta-atributul **allow-migrate** setat pe **true** (nu pe valoarea implicită)

Dacă resursa depinde de o clonă, iar la momentul când resursa trebuie să fie mutată, clona are instanţe care se opresc şi instanţe care pornesc, atunci resursa va fi mutată în modul tradiţional. Policy Engine-ul nu este capabil încă să modeleze această situaţie în mod corect aşa că ia calea (mai puţin optimă) dar mai sigură.

9.4. Refolosirea Regulilor, Opţiunilor şi a Setului de Operaţiuni

Câteodată un număr de restricţii trebuie să folosească acelaşi set de reguli şi resursele trebuie sa seteze aceleaşi opţiuni şi parametri. Pentru a simplifica această situaţie, puteţi face referinţă la un obiect existent folosind un **id-ref** în loc de un id.

Deci dacă pentru o resursă aveţi

```
<rsc_location id="WebServer-connectivity" rsc="Webserver">
   <rule id="ping-prefer-rule" score-attribute="pingd" >
    <expression id="ping-prefer" attribute="pingd" operation="defined"/>
   </rule>
</rsc_location>
```

Then instead of duplicating the rule for all your other resources, you can instead specify:

Exemplu 9.8. Realizând referinţe către reguli din alte restricţii

```
<rsc_location id="WebDB-connectivity" rsc="WebDB">
      <rule id-ref="ping-prefer-rule"/>
</rsc_location>
```

Important

Clusterul va insista că **`regula`** există undeva. Încercând să adaugaţi o referinţă către o regulă ce nu există va cauza un eşec al validării, la fel ca şi încercarea de a înlătura **`regula`** care este referenţiată în altă parte.

The same principle applies for **`meta_attributes`** and **`instance_attributes`** as illustrated in the example below:

Exemplu 9.9. Referenţierea atributelor, opţiunilor şi operaţiunilor din alte resurse

```
<primitive id="mySpecialRsc" class="ocf" type="Special" provider="me">
   <instance_attributes id="mySpecialRsc-attrs" score="1" >
     <nvpair id="default-interface" name="interface" value="eth0"/>
     <nvpair id="default-port" name="port" value="9999"/>
   </instance_attributes>
   <meta_attributes id="mySpecialRsc-options">
     <nvpair id="failure-timeout" name="failure-timeout" value="5m"/>
     <nvpair id="migration-threshold" name="migration-threshold" value="1"/>
     <nvpair id="stickiness" name="resource-stickiness" value="0"/>
   </meta_attributes>
   <operations id="health-checks">
     <op id="health-check" name="monitor" interval="60s"/>
     <op id="health-check" name="monitor" interval="30min"/>
   </operations>
</primitive>
<primitive id="myOtherlRsc" class="ocf" type="Other" provider="me">
   <instance_attributes id-ref="mySpecialRsc-attrs"/>
   <meta_attributes id-ref="mySpecialRsc-options"/>
   <operations id-ref="health-checks"/>
</primitive>
```

9.5. Reîncărcarea Serviciilor După Schimbarea unei Definiţii

Clusterul detectează în mod automat schimbări ale definiţiei serviciilor pe care le gestionează. Totuşi, răspunsul normal este să oprească serviciul (folosind definiţia veche) şi să îl pornească din nou (cu definiţia nouă). Acest lucru funcţionează bine, dar unele servicii sunt inteligente şi li se poate spune să folosească un set nou de opţiuni fară să repornească.

Pentru a profita de această capabilitate, agentul vostru de resursă trebuie să:

1. Accept the **`reload`** operation and perform any required actions. *The steps required here depend completely on your application!*

   Exemplu 9.10. The DRBD Agent's Control logic for Supporting the **`reload`** Operation

   ```
   case $1 in
       start)
           drbd_start
           ;;
       stop)
           drbd_stop
           ;;
       reload)
           drbd_reload
   ```

```
        ;;
    monitor)
        drbd_monitor
        ;;
    *)
        drbd_usage
        exit $OCF_ERR_UNIMPLEMENTED
        ;;
esac
exit $?
```

2. Anunţă operaţiunea **`reload`** în secţiunea de **`actions`** din meta-informaţiile proprii

Exemplu 9.11. Anunţarea Suportului Operaţiunii de **`reload`** a Agentului DRBD

```
<?xml version="1.0"?>
  <!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
  <resource-agent name="drbd">
    <version>1.1</version>

    <longdesc>
      Master/Slave OCF Resource Agent for DRBD
    </longdesc>

    ...

    <actions>
      <action name="start"   timeout="240" />
      <action name="reload"  timeout="240" />
      <action name="promote" timeout="90" />
      <action name="demote"  timeout="90" />
      <action name="notify"  timeout="90" />
      <action name="stop"    timeout="100" />
      <action name="meta-data"    timeout="5" />
      <action name="validate-all" timeout="30" />
    </actions>
  </resource-agent>
```

3. Promovaţi unul sau mai mulţi parametri care pot intra în vigoare folosing **`reload`**.

   Orice parametru cu **`unique`** setat pe 0 este eligibil să fie folosit în acest fel.

Exemplu 9.12. Parametru care poate fi schimbat folosind reload

```
<parameter name="drbdconf" unique="0">
    <longdesc>Full path to the drbd.conf file.</longdesc>
    <shortdesc>Path to drbd.conf</shortdesc>
    <content type="string" default="${OCF_RESKEY_drbdconf_default}"/>
</parameter>
```

Odată ce aceste cerinţe au fost satisfăcute, clusterul automat va şti să reîncarce, în loc să restarteze, resursa când un câmp non-unic se schimbă.

Notă

Meta-informaţiile sunt recitite când resursa este pornită. Acest lucru înseamnă că resursa va fi restartată prima data, deşi aţi schimbat un parametru cu **`unique=0`**

Notă

Dacă atât un câmp unic si non-unic sunt schimbate simultan, resursa tot va fi restartată.

Tipuri Avansate de Resurse

Cuprins

10.1. Groups - A Syntactic Shortcut

Unul dintre cele mai comune elemente ale unui cluster este un set de resurse care trebuie plasate împreună, pornesc secvenţial şi se opres în ordine inversă. Pentru a simplifica această configuraţie suportăm conceptul de grupuri.

Exemplu 10.1. Un exemplu de grup

```
<group id="shortcut">
   <primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
    <instance_attributes id="params-public-ip">
       <nvpair id="public-ip-addr" name="ip" value="1.2.3.4"/>
    </instance_attributes>
   </primitive>
   <primitive id="Email" class="lsb" type="exim"/>
  </group>
```

Deşi exemplul de mai sus conţine doar două resurse, nu este nici o limită asupra numărului de resurse pe care le poate conţine un grup. Exemplul este de asemenea suficient pentru a explica proprietăţile fundamentale ale unui grup:

- Resursele sunt pornite în ordinea în care apar (întâi **`Public-IP`**, apoi **`Email`**)
- Resursele sunt oprite în ordine inversă faţă de cea în care apar (întâi **`Email`**, apoi **`Public-IP`**)

If a resource in the group can't run anywhere, then nothing after that is allowed to run, too.

- If **`Public-IP`** can't run anywhere, neither can **`Email`**;
- but if **`Email`** can't run anywhere, this does not affect **`Public-IP`** in any way

Grupul de deasupra este echivalent logic cu a scrie:

Exemplu 10.2. Cum vede clusterul un grup de resurse

```
<configuration>
   <resources>
    <primitive id="Public-IP" class="ocf" type="IPaddr" provider="heartbeat">
     <instance_attributes id="params-public-ip">
        <nvpair id="public-ip-addr" name="ip" value="1.2.3.4"/>
     </instance_attributes>
    </primitive>
    <primitive id="Email" class="lsb" type="exim"/>
   </resources>
   <constraints>
      <rsc_colocation id="xxx" rsc="Email" with-rsc="Public-IP" score="INFINITY"/>
      <rsc_order id="yyy" first="Public-IP" then="Email"/>
   </constraints>
</configuration>
```

În mod evident pe măsură ce grupul creşte, efortul de configurare redus poate deveni semnificativ.

Un alt exemplu (tipi) al unui grup este un volum DBRD, un mount de sistem de fișiere, o adresă IP și o aplicație care le folosește.

10.1.1. Group Properties

Tabel 10.1. Proprietăţile unui Grup de Resurse

Câmp	Descriere
`id`	Your name for the group

10.1.2. Group Options

Options inherited from *primitive* resources: **`priority, target-role, is-managed`**

10.1.3. Group Instance Attributes

Groups have no instance attributes, however any that are set here will be inherited by the group's children.

10.1.4. Group Contents

Groups may only contain a collection of *Secțiune 5.3, „Resource Properties”* cluster resources. To refer to the child of a group resource, just use the child's id instead of the group's.

10.1.5. Group Constraints

Although it is possible to reference the group's children in constraints, it is usually preferable to use the group's name instead.

Exemplu 10.3. Exemple de restricţii care implică grupuri

```
<constraints>
    <rsc_location id="group-prefers-node1" rsc="shortcut" node="node1" score="500"/>
    <rsc_colocation id="webserver-with-group" rsc="Webserver" with-rsc="shortcut"/>
    <rsc_order id="start-group-then-webserver" first="Webserver" then="shortcut"/>
</constraints>
```

10.1.6. Group Stickiness

Stickiness, the measure of how much a resource wants to stay where it is, is additive in groups. Every active resource of the group will contribute its stickiness value to the group's total. So if the default **`resource-stickiness`** is 100, and a group has seven members, five of which are active, then the group as a whole will prefer its current location with a score of 500.

10.2. Clones - Resources That Get Active on Multiple Hosts

Clones were initially conceived as a convenient way to start N instances of an IP resource and have them distributed throughout the cluster for load balancing. They have turned out to quite useful for a number of purposes including integrating with Red Hat's DLM, the fencing subsystem, and OCFS2.

Puteţi clona orice resursă atâta timp cât agentul de resursă suportă acest lucru.

Există trei tipuri de resurse clonate.

- Anonime
- Unice la nivel global
- Stateful

Clonele anonime sunt tipul cel mai simplu. Aceste resurse se comportă absolut identic oriunde rulează. Din această cauză, poate exista doar o copie a unei clone anonime activă per maşină.

Clonele unice la nivel global sunt entităţi distincte. O copie a unei clone rulând pe o maşină nu este echivalentă cu o altă instanţă pe alt nod. Nici nu ar fi echivalente oricare două copii pe acelaşi nod.

Clonele stateful sunt discutate mai târziu în *Secţiune 10.3, „Multi-state - Resources That Have Multiple Modes”*.

Exemplu 10.4. Un exemplu de clonă

```
<clone id="apache-clone">
    <meta_attributes id="apache-clone-meta">
       <nvpair id="apache-unique" name="globally-unique" value="false"/>
    </meta_attributes>
    <primitive id="apache" class="lsb" type="apache"/>
</clone>
```

10.2.1. Clone Properties

Tabel 10.2. Proprietăţile unei Resurse Clonă

Câmp	Descriere
id	Your name for the clone

10.2.2. Clone Options

Options inherited from *primitive* resources: **priority, target-role, is-managed**

Tabel 10.3. Opţiuni de configurare specifice clonei

Câmp	Descriere
clone-max	How many copies of the resource to start. Defaults to the number of nodes in the cluster.
clone-node-max	How many copies of the resource can be started on a single node; default *1*.
notify	When stopping or starting a copy of the clone, tell all the other copies beforehand and when the action was successful. Allowed values: *false*, **true**
globally-unique	Does each copy of the clone perform a different function? Allowed values: *false*, **true**
ordered	Should the copies be started in series (instead of in parallel). Allowed values: *false*, **true**
interleave	Changes the behavior of ordering constraints (between clones/masters) so that instances can start/stop as soon as their peer instance has (rather than waiting for every instance of the other clone has). Allowed values: *false*, **true**

10.2.3. Clone Instance Attributes

Clones have no instance attributes; however, any that are set here will be inherited by the clone's children.

10.2.4. Clone Contents

Clonele trebuie să conţină fix un grup sau o resursă obişnuită.

Avertisment

You should never reference the name of a clone's child. If you think you need to do this, you probably need to re-evaluate your design.

10.2.5. Clone Constraints

In most cases, a clone will have a single copy on each active cluster node. If this is not the case, you can indicate which nodes the cluster should preferentially assign copies to with resource location constraints. These constraints are written no differently to those for regular resources except that the clone's id is used.

Restricţiile de ordonare se comportă uşor diferit în cazul clonelor. În exemplele de mai jos, **apache-stats** va aştepta până ce toate copiile clonelor care trebuie să fie pornite au făcut acest lucru înainte ca aceasta să fie pornită la rândul ei. Doar dacă *nici o* copie nu poate fi pornită va fi împiedicată **apache-stats** din a fi activă. În plus, clona va aştepta ca apache-stats să fie oprită înainte de a opri clona.

Colocation of a regular (or group) resource with a clone means that the resource can run on any machine with an active copy of the clone. The cluster will choose a copy based on where the clone is running and the resource's own location preferences.

Colocarea între clone este posibilă de asemenea. În astfel de cazuri, setul de locaţii permise pentru clonă este limitat la nodurile pe care clona alături de care va fi colocată este (sau va fi) activă. Alocarea este mai apoi efectuată în mod normal.

Exemplu 10.5. Exemple de restricţii implicând clone

```
<constraints>
    <rsc_location id="clone-prefers-node1" rsc="apache-clone" node="node1" score="500"/>
    <rsc_colocation id="stats-with-clone" rsc="apache-stats" with="apache-clone"/>
    <rsc_order id="start-clone-then-stats" first="apache-clone" then="apache-stats"/>
</constraints>
```

10.2.6. Clone Stickiness

To achieve a stable allocation pattern, clones are slightly sticky by default. If no value for **resource-stickiness** is provided, the clone will use a value of 1. Being a small value, it causes minimal disturbance to the score calculations of other resources but is enough to prevent Pacemaker from needlessly moving copies around the cluster.

10.2.7. Clone Resource Agent Requirements

Orice resursă poate fi utilizată ca o clonă anonimă deoarece nu necesită vreun suport adiţional din partea agentului de resursă. Dacă are logică să facă acest lucru depinde de resursa voastră şi de agentul de resursă aferent.

Globally unique clones do require some additional support in the resource agent. In particular, it must only respond with other probes for instances of the clone should result in they should return one of the other OCF error codes.

Copiile unei clone sunt identificate prin sufixarea a două puncte şi a unui delimitator numeric. Ex. **apache:2**

Resource agents can find out how many copies there are by examining the **OCF_RESKEY_CRM_meta_clone_max** environment variable and which copy it is by examining **OCF_RESKEY_CRM_meta_clone**.

You should not make any assumptions (based on **OCF_RESKEY_CRM_meta_clone**) about which copies are active. In particular, the list of active copies will not always be an unbroken sequence, nor always start at 0.

10.2.7.1. Clone Notifications

Suportarea notificărilor necesită acţiunea **notify** să fie implementată. Odată ce este suportată, acţiunii de notificare îi vor fi trimise un număr de variabile suplimentare care, atunci când sunt

combinate cu un context adiţional, pot fi folosite pentru a calcula starea curentă a clusterului şi ceea ce urmează să i se întâmple.

Tabel 10.4. Variabile de mediu furnizate împreună cu acţiunile de notificare ale Clonei

Variabilă	Descriere
OCF_RESKEY_CRM_meta_notify_type	Allowed values: **pre**, **post**
OCF_RESKEY_CRM_meta_notify_operation	Allowed values: **start**, **stop**
OCF_RESKEY_CRM_meta_notify_start_resource	Resources to be started
OCF_RESKEY_CRM_meta_notify_stop_resource	Resources to be stopped
OCF_RESKEY_CRM_meta_notify_active_resource	Resources that are running
OCF_RESKEY_CRM_meta_notify_inactive_resource	Resources that are not running
OCF_RESKEY_CRM_meta_notify_start_uname	Nodes on which resources will be started
OCF_RESKEY_CRM_meta_notify_stop_uname	Nodes on which resources will be stopped
OCF_RESKEY_CRM_meta_notify_active_uname	Nodes on which resources are running
OCF_RESKEY_CRM_meta_notify_inactive_uname	Nodes on which resources are not running

The variables come in pairs, such as **OCF_RESKEY_CRM_meta_notify_start_resource** and **OCF_RESKEY_CRM_meta_notify_start_uname** and should be treated as an array of whitespace separated elements.

Drept urmare pentru a indica faptul că, **clone:0** va fi pornită pe **sles-1**, **clone:2** va fi pornită pe **sles-3** şi **clone:3** va fi pornită pe **sles-2**, clusterul va seta

Exemplu 10.6. Exemple de variabile de notificare

```
OCF_RESKEY_CRM_meta_notify_start_resource="clone:0 clone:2 clone:3"
OCF_RESKEY_CRM_meta_notify_start_uname="sles-1 sles-3 sles-2"
```

10.2.7.2. Interpretarea Corespunzătoare a Variabilelor de Mediu de Notificare

Pre-notificare (oprire)

- Active resources: **$OCF_RESKEY_CRM_meta_notify_active_resource**
- Inactive resources: **$OCF_RESKEY_CRM_meta_notify_inactive_resource**
- Resources to be started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**

Post-notificare (oprire) / Pre-notificare (pornire)

- Resurse active:
  - **$OCF_RESKEY_CRM_meta_notify_active_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resurse inactive:

  - **$OCF_RESKEY_CRM_meta_notify_inactive_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resources that were started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources that were stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**

Post-notificare (pornire)
- Resurse active:
  - **$OCF_RESKEY_CRM_meta_notify_active_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resurse inactive:
  - **$OCF_RESKEY_CRM_meta_notify_inactive_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources that were started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources that were stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**

10.3. Multi-state - Resources That Have Multiple Modes

Resursele multi-state sunt o specializare a Clonelor (vă rugăm să vă asiguraţi că înţelegeţi secţiunea referitoare la clone înainte de a continua) care permite instanţelor să se afle în unul din două moduri operaţionale; aceste moduri sunt numite **Master** şi **Slave** dar pot însemna orice doriţi să însemne. Singura limitare este că atunci când o instanţă este pornită, trebuie să o facă în starea **Slave**.

10.3.1. Multi-state Properties

Tabel 10.5. Proprietăţile unei Resurse Multi-State

Câmp	Descriere
id	Your name for the multi-state resource

10.3.2. Multi-state Options

Options inherited from *primitive* resources: **priority**, **target-role**, **is-managed**

Options inherited from *clone* resources: **clone-max**, **clone-node-max**, **notify**, **globally-unique**, **ordered**, **interleave**

Tabel 10.6. Opţiuni specifice de configurare pentru resurse multi-state

Câmp	Descriere
master-max	How many copies of the resource can be promoted to **master** status; default 1.

Câmp	Descriere
master-node-max	How many copies of the resource can be promoted to **master** status on a single node; default 1.

10.3.3. Multi-state Instance Attributes

Multi-state resources have no instance attributes; however, any that are set here will be inherited by master's children.

10.3.4. Multi-state Contents

Masterii trebuie să conţină exact un grup sau o resursă obişnuită.

Avertisment

You should never reference the name of a master's child. If you think you need to do this, you probably need to re-evaluate your design.

10.3.5. Monitorizarea Resurselor Multi-State

Tipul normal de acţiuni de monitorizare nu sunt suficiente pentru a monitoriza o resursă multi-state în starea **Master**. Pentru a detecta eşecurile instanţei **Master**, trebuie să definiţi o acţiune de monitorizare adiţională cu **role="Master"**.

Important

Este imperativ ca *fiecare* operaţiune de monitorizare să aibă un interval diferit!

Acest lucru este necesar deoarece Pacemaker diferențiază în mod curent între operațiuni numai după resursă și interval; deci dacă de ex. o resursă master/slave are același interval de monitorizare pentru ambele roluri, Pacemaker ar ignora rolul când ar verifica statusul - fapt care ar cauza coduri de ieșire neașteptate și respectiv complicații inutile.

Exemplu 10.7. Monitorizarea ambelor stări ale unei resurse multi-state

```
<master id="myMasterRsc">
   <primitive id="myRsc" class="ocf" type="myApp" provider="myCorp">
    <operations>
     <op id="public-ip-slave-check" name="monitor" interval="60"/>
     <op id="public-ip-master-check" name="monitor" interval="61" role="Master"/>
    </operations>
   </primitive>
</master>
```

10.3.6. Multi-state Constraints

In most cases, a multi-state resources will have a single copy on each active cluster node. If this is not the case, you can indicate which nodes the cluster should preferentially assign copies to with resource

location constraints. These constraints are written no differently to those for regular resources except that the master's id is used.

Când ţinem cont de resursele multi-state în restricţii, pentru majoritatea scopurilor este suficient să le tratăm ca pe clone. Excepţia survine atunci când sunt folosite câmpurile **rsc-role** şi/sau **with-rsc-role** (pentru restricţii de colocare) şi câmpurile **first-action** şi/sau **then-action** (pentru restricţii de ordonare).

Tabel 10.7. Opţiuni de restricţionare adiţionale relevante la resurse multi-state

Câmp	Descriere
rsc-role	An additional attribute of colocation constraints that specifies the role that **rsc** must be in. Allowed values: *Started*, **Master**, **Slave**.
with-rsc-role	An additional attribute of colocation constraints that specifies the role that **with-rsc** must be in. Allowed values: *Started*, **Master**, **Slave**.
first-action	An additional attribute of ordering constraints that specifies the action that the **first** resource must complete before executing the specified action for the **then** resource. Allowed values: *start*, **stop**, **promote**, **demote**.
then-action	An additional attribute of ordering constraints that specifies the action that the **then** resource can only execute after the **first-action** on the **first** resource has completed. Allowed values: **start**, **stop**, **promote**, **demote**. Defaults to the value (specified or implied) of **first-action**.

În exemplul de mai jos, **myApp** va aştepta până când una din copiile bazei de date a fost pornită şi promovată la master înainte de a fi ea însăşi pornită. Doar dacă nici o copie nu poate fi promovată va fi împiedicată **apache-stats** de a fi activă. În mod adiţional, baza de date va aştepta ca **myApp** să fie oprită înainte să fie degradată.

Exemplu 10.8. Exemple de restricţii implicând resurse multi-state

```
<constraints>
   <rsc_location id="db-prefers-node1" rsc="database" node="node1" score="500"/>
   <rsc_colocation id="backup-with-db-slave" rsc="backup"
     with-rsc="database" with-rsc-role="Slave"/>
   <rsc_colocation id="myapp-with-db-master" rsc="myApp"
     with-rsc="database" with-rsc-role="Master"/>
   <rsc_order id="start-db-before-backup" first="database" then="backup"/>
   <rsc_order id="promote-db-then-app" first="database" first-action="promote"
     then="myApp" then-action="start"/>
</constraints>
```

Colocarea unei resurse obişnuite (sau a unui grup) cu o resursă multi-state înseamnă că poate rula pe orice maşină cu o copie activă a resursei multi-state care se alfă în starea specificată (**Master** sau **Slave**). În exemplu, clusterul va alege o locaţie în funcţie de unde rulează baza de date în mod curent ca **Master**, şi dacă sunt mai multe instanţe de **Master** va lua în considerare şi preferinţele proprii de locaţie ale **myApp** când va decide care locaţie să aleagă.

Colocarea alături de clone obişnuite şi alte resurse multi-state este posibilă de asemenea. În astfel de cazuri, setul de locaţii permise pentru clona **rsc** este (după filtrarea rolului) limitat la nodurile pe care resursa multi-state **with-rsc** există (sau va exista) în rolul specificat. Alocarea este atunci efectuată în mod normal.

10.3.7. Multi-state Stickiness

To achieve a stable allocation pattern, multi-state resources are slightly sticky by default. If no value for **`resource-stickiness`** is provided, the multi-state resource will use a value of 1. Being a small value, it causes minimal disturbance to the score calculations of other resources but is enough to prevent Pacemaker from needlessly moving copies around the cluster.

10.3.8. Care Instanţă a Resursei este Promovată

During the start operation, most Resource Agent scripts should call the **`crm_master`** utility. This tool automatically detects both the resource and host and should be used to set a preference for being promoted. Based on this, **`master-max`**, and **`master-node-max`**, the instance(s) with the highest preference will be promoted.

Cealaltă alternativă să se creeze o restricţie de locaţie care să indice care noduri sunt cele mai preferate ca masteri.

Exemplu 10.9. Specificând manual care nod ar trebui să fie promovat

```
<rsc_location id="master-location" rsc="myMasterRsc">
    <rule id="master-rule" score="100" role="Master">
      <expression id="master-exp" attribute="#uname" operation="eq" value="node1"/>
    </rule>
</rsc_location>
```

10.3.9. Multi-state Resource Agent Requirements

Since multi-state resources are an extension of cloned resources, all the requirements of Clones are also requirements of multi-state resources. Additionally, multi-state resources require two extra actions: **`demote`** and **`promote`**; these actions are responsible for changing the state of the resource. Like **`start`** and **`stop`**, they should return **`OCF_SUCCESS`** if they completed successfully or a relevant error code if they did not.

Stările pot însemna orice doriţi, dar atunci când resursa este pornită, trebuie să ajungă în modul numit **`Slave`**. De acolo clusterul va decide mai apoi care instanţe să promoveze într-un **`Master`**.

În plus faţă de cerinţele Clonelor pentru acţiunile de monitorizare, agenţii trebuie să raporteze *corespunzător* starea în care sunt. Clusterul se bazează pe agent să-şi raporteze statusul (incluzând rolul) în mod corespunzător şi nu indică agentului în care rol crede acesta în mod curent că ar trebui să se afle.

Tabel 10.8. Implicaţiile rolurilor codurilor returnate de OCF

Cod de Monitorizare Returnat	Descriere
OCF_NOT_RUNNING	Stopped
OCF_SUCCESS	Running (Slave)
OCF_RUNNING_MASTER	Running (Master)
OCF_FAILED_MASTER	Failed (Master)
Altul	Eşuat (Slave)

10.3.10. Multi-state Notifications

Ca şi în cazul clonelor, suportarea notificărilor necesită acţiunea **`notify`** să fie implementată. Odată ce este suportată, acţiunii notify îi vor fi trimise un număr de variabile suplimentare care, când sunt

combinate cu un context suplimentar, pot fi folosite pentru a calcula starea curentă a clusterului şi ceea ce urmează să i se întâmple.

Tabel 10.9. Environment variables supplied with Master notify actions [1]

Variabilă	Descriere
OCF_RESKEY_CRM_meta_notify_type	Allowed values: **pre**, **post**
OCF_RESKEY_CRM_meta_notify_operation	Allowed values: **start**, **stop**
OCF_RESKEY_CRM_meta_notify_active_resource	Resources the that are running
OCF_RESKEY_CRM_meta_notify_inactive_resource	Resources the that are not running
OCF_RESKEY_CRM_meta_notify_master_resource	Resources that are running in **Master** mode
OCF_RESKEY_CRM_meta_notify_slave_resource	Resources that are running in **Slave** mode
OCF_RESKEY_CRM_meta_notify_start_resource	Resources to be started
OCF_RESKEY_CRM_meta_notify_stop_resource	Resursele care vor fi oprite
OCF_RESKEY_CRM_meta_notify_promote_resource	Resources to be promoted
OCF_RESKEY_CRM_meta_notify_demote_resource	Resources to be demoted
OCF_RESKEY_CRM_meta_notify_start_uname	Nodes on which resources will be started
OCF_RESKEY_CRM_meta_notify_stop_uname	Nodes on which resources will be stopped
OCF_RESKEY_CRM_meta_notify_promote_uname	Nodes on which resources will be promote
OCF_RESKEY_CRM_meta_notify_demote_uname	Nodes on which resources will be demoted
OCF_RESKEY_CRM_meta_notify_active_uname	Nodes on which resources are running
OCF_RESKEY_CRM_meta_notify_inactive_uname	Nodes on which resources are not running
OCF_RESKEY_CRM_meta_notify_master_uname	Nodes on which resources are running in **Master** mode
OCF_RESKEY_CRM_meta_notify_slave_uname	Nodes on which resources are running in **Slave** mode

[1] Emphasized variables are specific to **Master** resources and all behave in the same manner as described for Clone resources.

10.3.11. Multi-state - Proper Interpretation of Notification Environment Variables

Pre-notificare (degradează)

- **Active** resources: **$OCF_RESKEY_CRM_meta_notify_active_resource**
- **Master** resources: **$OCF_RESKEY_CRM_meta_notify_master_resource**
- **Slave** resources: **$OCF_RESKEY_CRM_meta_notify_slave_resource**
- Inactive resources: **$OCF_RESKEY_CRM_meta_notify_inactive_resource**

- Resources to be started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be promoted: **$OCF_RESKEY_CRM_meta_notify_promote_resource**
- Resources to be demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources to be stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**

Post-notificare (degradează) / Pre-notificare (oprire)

- **Active** resources: **$OCF_RESKEY_CRM_meta_notify_active_resource**
- Resurse **Master**:
  - **$OCF_RESKEY_CRM_meta_notify_master_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- **Slave** resources: **$OCF_RESKEY_CRM_meta_notify_slave_resource**
- Inactive resources: **$OCF_RESKEY_CRM_meta_notify_inactive_resource**
- Resources to be started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be promoted: **$OCF_RESKEY_CRM_meta_notify_promote_resource**
- Resources to be demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources to be stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resources that were demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**

Post-notificare (oprire) / Pre-notificare (pornire)

- Resursele **Active**:
  - **$OCF_RESKEY_CRM_meta_notify_active_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resurse **Master**:
  - **$OCF_RESKEY_CRM_meta_notify_master_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resursele **Slave**:
  - **$OCF_RESKEY_CRM_meta_notify_slave_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resurse inactive:
  - **$OCF_RESKEY_CRM_meta_notify_inactive_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resources to be started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be promoted: **$OCF_RESKEY_CRM_meta_notify_promote_resource**

- Resources to be demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources to be stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resources that were demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources that were stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**

Post-notificare (pornire) / Pre-notificare (promovează)

- Resursele **Active**:
  - **$OCF_RESKEY_CRM_meta_notify_active_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resurse **Master**:
  - **$OCF_RESKEY_CRM_meta_notify_master_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resursele **Slave**:
  - **$OCF_RESKEY_CRM_meta_notify_slave_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resurse inactive:
  - **$OCF_RESKEY_CRM_meta_notify_inactive_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be promoted: **$OCF_RESKEY_CRM_meta_notify_promote_resource**
- Resources to be demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources to be stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resources that were started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources that were demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources that were stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**

Post-notificare (promovează)

- Resursele **Active**:
  - **$OCF_RESKEY_CRM_meta_notify_active_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**

  - plus **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resurse **Master**:
  - **$OCF_RESKEY_CRM_meta_notify_master_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_demote_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_promote_resource**
- Resursele **Slave**:
  - **$OCF_RESKEY_CRM_meta_notify_slave_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_start_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_promote_resource**
- Resurse inactive:
  - **$OCF_RESKEY_CRM_meta_notify_inactive_resource**
  - plus **$OCF_RESKEY_CRM_meta_notify_stop_resource**
  - minus **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources to be promoted: **$OCF_RESKEY_CRM_meta_notify_promote_resource**
- Resources to be demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources to be stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**
- Resources that were started: **$OCF_RESKEY_CRM_meta_notify_start_resource**
- Resources that were promoted: **$OCF_RESKEY_CRM_meta_notify_promote_resource**
- Resources that were demoted: **$OCF_RESKEY_CRM_meta_notify_demote_resource**
- Resources that were stopped: **$OCF_RESKEY_CRM_meta_notify_stop_resource**

Utilization and Placement Strategy

Cuprins

11.1. Background

Pacemaker decides where to place a resource according to the resource allocation scores on every node. The resource will be allocated to the node where the resource has the highest score. If the resource allocation scores on all the nodes are equal, by the **`default`** placement strategy, Pacemaker will choose a node with the least number of allocated resources for balancing the load. If the number of resources on each node is equal, the first eligible node listed in cib will be chosen to run the resource.

Though resources are different. They may consume different amounts of the capacities of the nodes. Actually, we cannot ideally balance the load just according to the number of resources allocated to a node. Besides, if resources are placed such that their combined requirements exceed the provided capacity, they may fail to start completely or run with degraded performance.

To take these into account, Pacemaker allows you to specify the following configurations:

1. The **`capacity`** a certain **`node provides`**.
2. The **`capacity`** a certain **`resource requires`**.
3. An overall **`strategy`** for placement of resources.

11.2. Utilization attributes

To configure the capacity a node provides and the resource's requirements, use **`utilization`** attributes. You can name the **`utilization`** attributes according to your preferences and define as many **`name/value`** pairs as your configuration needs. However, the attribute's values must be **`integers`**.

First, specify the capacities the nodes provide:

```
<node id="node1" type="normal" uname="node1">
  <utilization id="node1-utilization">
    <nvpair id="node1-utilization-cpu" name="cpu" value="2"/>
    <nvpair id="node1-utilization-memory" name="memory" value="2048"/>
  </utilization>
</node>
<node id="node2" type="normal" uname="node2">
```

```
  <utilization id="node2-utilization">
    <nvpair id="node2-utilization-cpu" name="cpu" value="4"/>
    <nvpair id="node2-utilization-memory" name="memory" value="4096"/>
  </utilization>
</node>
```

Then, specify the capacities the resources require:

```
<primitive id="rsc-small" class="ocf" provider="pacemaker" type="Dummy">
  <utilization id="rsc-small-utilization">
    <nvpair id="rsc-small-utilization-cpu" name="cpu" value="1"/>
    <nvpair id="rsc-small-utilization-memory" name="memory" value="1024"/>
  </utilization>
</primitive>
<primitive id="rsc-medium" class="ocf" provider="pacemaker" type="Dummy">
  <utilization id="rsc-medium-utilization">
    <nvpair id="rsc-medium-utilization-cpu" name="cpu" value="2"/>
    <nvpair id="rsc-medium-utilization-memory" name="memory" value="2048"/>
  </utilization>
</primitive>
<primitive id="rsc-large" class="ocf" provider="pacemaker" type="Dummy">
  <utilization id="rsc-large-utilization">
    <nvpair id="rsc-large-utilization-cpu" name="cpu" value="3"/>
    <nvpair id="rsc-large-utilization-memory" name="memory" value="3072"/>
  </utilization>
</primitive>
```

A node is considered eligible for a resource if it has sufficient free capacity to satisfy the resource's requirements. The nature of the required or provided capacities is completely irrelevant for Pacemaker, it just makes sure that all capacity requirements of a resource are satisfied before placing a resource to a node.

11.3. Placement Strategy

After you have configured the capacities your nodes provide and the capacities your resources require, you need to set the **`placement-strategy`** in the global cluster options, otherwise the capacity configurations have **`no effect`**.

Four values are available for the **`placement-strategy`**:

`default`
: Utilization values are not taken into account at all, per default. Resources are allocated according to allocation scores. If scores are equal, resources are evenly distributed across nodes.

`utilization`
: Utilization values are taken into account when deciding whether a node is considered eligible if it has sufficient free capacity to satisfy the resource's requirements. However, load-balancing is still done based on the number of resources allocated to a node.

`balanced`
: Utilization values are taken into account when deciding whether a node is eligible to serve a resource; an attempt is made to spread the resources evenly, optimizing resource performance.

`minimal`
: Utilization values are taken into account when deciding whether a node is eligible to serve a resource; an attempt is made to concentrate the resources on as few nodes as possible, thereby enabling possible power savings on the remaining nodes.

Set **`placement-strategy`** with **`crm_attribute`**:

```
# crm_attribute --attr-name placement-strategy --attr-value balanced
```

Now Pacemaker will ensure the load from your resources will be distributed evenly throughout the cluster - without the need for convoluted sets of colocation constraints.

11.4. Allocation Details

11.4.1. Which node is preferred to be chosen to get consumed first on allocating resources?

- The node that is most healthy (which has the highest node weight) gets consumed first.
- If their weights are equal:
  - If **`placement-strategy="default|utilization"`**, the node that has the least number of allocated resources gets consumed first.
    - If their numbers of allocated resources are equal, the first eligible node listed in cib gets consumed first.
  - If **`placement-strategy="balanced"`**, the node that has more free capacity gets consumed first.
    - If the free capacities of the nodes are equal, the node that has the least number of allocated resources gets consumed first.
      - If their numbers of allocated resources are equal, the first eligible node listed in cib gets consumed first.
  - If **`placement-strategy="minimal"`**, the first eligible node listed in cib gets consumed first.

11.4.1.1. Which node has more free capacity?

This will be quite clear if we only define one type of **`capacity`**. While if we define multiple types of **`capacity`**, for example:

- If **`nodeA`** has more free **`cpus`**, **`nodeB`** has more free **`memory`**, their free capacities are equal.
- If **`nodeA`** has more free **`cpus`**, while **`nodeB`** has more free **`memory`** and **`storage`**, **`nodeB`** has more free capacity.

11.4.2. Which resource is preferred to be chosen to get assigned first?

- The resource that has the highest priority gets allocated first.
- If their priorities are equal, check if they are already running. The resource that has the highest score on the node where it's running gets allocated first (to prevent resource shuffling).
- If the scores above are equal or they are not running, the resource has the highest score on the preferred node gets allocated first.
- If the scores above are equal, the first runnable resource listed in cib gets allocated first.

11.5. Limitations

This type of problem Pacemaker is dealing with here is known as the *knapsack problem*[1] and falls into the *NP-complete*[2] category of computer science problems - which is fancy way of saying "it takes a really long time to solve".

Clearly in a HA cluster, it's not acceptable to spend minutes, let alone hours or days, finding an optional solution while services remain unavailable.

So instead of trying to solve the problem completely, Pacemaker uses a *best effort* algorithm for determining which node should host a particular service. This means it arrives at a solution much faster than traditional linear programming algorithms, but by doing so at the price of leaving some services stopped.

In the contrived example above:

- **`rsc-small`** would be allocated to **`node1`**
- **`rsc-medium`** would be allocated to **`node2`**
- **`rsc-large`** would remain inactive

Which is not ideal.

11.6. Strategies for Dealing with the Limitations

- Ensure you have sufficient physical capacity. It might sounds obvious, but if the physical capacity of your nodes is (close to) maxed out by the cluster under normal conditions, then failover isn't going to go well. Even without the Utilization feature, you'll start hitting timeouts and getting secondary failures'.
- Build some buffer into the capabilities advertised by the nodes. Advertise slightly more resources than we physically have on the (usually valid) assumption that a resource will not use 100% of the configured number of cpu/memory/etc **`all`** the time. This practice is also known as *over commit*.
- Specify resource priorities. If the cluster is going to sacrifice services, it should be the ones you care (comparatively) about the least. Ensure that resource priorities are properly set so that your most important resources are scheduled first.

[1] http://en.wikipedia.org/wiki/Knapsack_problem
[2] http://en.wikipedia.org/wiki/NP-complete

Resource Templates

Cuprins

12.1. Abstract

If you want to create lots of resources with similar configurations, defining a resource template simplifies the task. Once defined, it can be referenced in primitives or in certain types of constraints.

12.2. Configuring Resources with Templates

The primitives referencing the template will inherit all meta attributes, instance attributes, utilization attributes and operations defined in the template. And you can define specific attributes and operations for any of the primitives. If any of these are defined in both the template and the primitive, the values defined in the primitive will take precedence over the ones defined in the template.

Hence, resource templates help to reduce the amount of configuration work. If any changes are needed, they can be done to the template definition and will take effect globally in all resource definitions referencing that template.

Resource templates have a similar syntax like primitives. For example:

```
<template id="vm-template" class="ocf" provider="heartbeat" type="Xen">
  <meta_attributes id="vm-template-meta_attributes">
    <nvpair id="vm-template-meta_attributes-allow-migrate" name="allow-migrate" value="true"/
>
  </meta_attributes>
  <utilization id="vm-template-utilization">
    <nvpair id="vm-template-utilization-memory" name="memory" value="512"/>
  </utilization>
  <operations>
    <op id="vm-template-monitor-15s" interval="15s" name="monitor" timeout="60s"/>
    <op id="vm-template-start-0" interval="0" name="start" timeout="60s"/>
  </operations>
</template>
```

Once you defined the new resource template, you can use it in primitives:

```
<primitive id="vm1" template="vm-template">
  <instance_attributes id="vm1-instance_attributes">
    <nvpair id="vm1-instance_attributes-name" name="name" value="vm1"/>
    <nvpair id="vm1-instance_attributes-xmfile" name="xmfile" value="/etc/xen/shared-vm/vm1"/
>
  </instance_attributes>
</primitive>
```

The new primitive **`vm1`** is going to inherit everything from the **`vm-template`**. For example, the equivalent of the above two would be:

```
<primitive id="vm1" class="ocf" provider="heartbeat" type="Xen">
  <meta_attributes id="vm-template-meta_attributes">
```

```
    <nvpair id="vm-template-meta_attributes-allow-migrate" name="allow-migrate" value="true"/
>
  </meta_attributes>
  <utilization id="vm-template-utilization">
    <nvpair id="vm-template-utilization-memory" name="memory" value="512"/>
  </utilization>
  <operations>
    <op id="vm-template-monitor-15s" interval="15s" name="monitor" timeout="60s"/>
    <op id="vm-template-start-0" interval="0" name="start" timeout="60s"/>
  </operations>
  <instance_attributes id="vm1-instance_attributes">
    <nvpair id="vm1-instance_attributes-name" name="name" value="vm1"/>
    <nvpair id="vm1-instance_attributes-xmfile" name="xmfile" value="/etc/xen/shared-vm/vm1"/
>
  </instance_attributes>
</primitive>
```

If you want to overwrite some attributes or operations, add them to the particular primitive's definition.

For instance, the following new primitive **vm2** has special attribute values. Its **monitor** operation has a longer **timeout** and **interval**, and the primitive has an additional **stop** operation.

```
<primitive id="vm2" template="vm-template">
  <meta_attributes id="vm2-meta_attributes">
    <nvpair id="vm2-meta_attributes-allow-migrate" name="allow-migrate" value="false"/>
  </meta_attributes>
  <utilization id="vm2-utilization">
    <nvpair id="vm2-utilization-memory" name="memory" value="1024"/>
  </utilization>
  <instance_attributes id="vm2-instance_attributes">
    <nvpair id="vm2-instance_attributes-name" name="name" value="vm2"/>
    <nvpair id="vm2-instance_attributes-xmfile" name="xmfile" value="/etc/xen/shared-vm/vm2"/
>
  </instance_attributes>
  <operations>
    <op id="vm2-monitor-30s" interval="30s" name="monitor" timeout="120s"/>
    <op id="vm2-stop-0" interval="0" name="stop" timeout="60s"/>
  </operations>
</primitive>
```

The following command shows the resulting definition of a resource:

```
# crm_resource --query-xml --resource vm2
```

The following command shows its raw definition in cib:

```
# crm_resource --query-xml-raw --resource vm2
```

12.3. Referencing Templates in Constraints

A resource template can be referenced in the following types of constraints:

- **order** constraints
- **colocation** constraints,
- **rsc_ticket** constraints (for multi-site clusters).

Resource templates referenced in constraints stand for all primitives which are derived from that template. This means, the constraint applies to all primitive resources referencing the resource

template. Referencing resource templates in constraints is an alternative to resource sets and can simplify the cluster configuration considerably.

For example:

```
<rsc_colocation id="vm-template-colo-base-rsc" rsc="vm-template" rsc-role="Started" with-
rsc="base-rsc" score="INFINITY"/>
```

is the equivalent of the following constraint configuration:

```
<rsc_colocation id="vm-colo-base-rsc" score="INFINITY">
  <resource_set id="vm-colo-base-rsc-0" sequential="false" role="Started">
    <resource_ref id="vm1"/>
    <resource_ref id="vm2"/>
  </resource_set>
  <resource_set id="vm-colo-base-rsc-1">
    <resource_ref id="base-rsc"/>
  </resource_set>
</rsc_colocation>
```

Notă

In a colocation constraint, only one template may be referenced from either **`rsc`** or **`with-rsc`**, and the other reference must be a regular resource.

Resource templates can also be referenced in resource sets.

For example:

```
<rsc_order id="order1" score="INFINITY">
  <resource_set id="order1-0">
    <resource_ref id="base-rsc"/>
    <resource_ref id="vm-template"/>
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

is the equivalent of the following constraint configuration:

```
<rsc_order id="order1" score="INFINITY">
  <resource_set id="order1-0">
    <resource_ref id="base-rsc"/>
    <resource_ref id="vm1"/>
    <resource_ref id="vm2"/>
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

If the resources referencing the template can run in parallel:

```
<rsc_order id="order2" score="INFINITY">
  <resource_set id="order2-0">
    <resource_ref id="base-rsc"/>
  </resource_set>
  <resource_set id="order2-1" sequential="false">
```

```
    <resource_ref id="vm-template"/>
  </resource_set>
  <resource_set id="order2-2">
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

is the equivalent of the following constraint configuration:

```
<rsc_order id="order2" score="INFINITY">
  <resource_set id="order2-0">
    <resource_ref id="base-rsc"/>
  </resource_set>
  <resource_set id="order2-1" sequential="false">
    <resource_ref id="vm1"/>
    <resource_ref id="vm2"/>
  </resource_set>
  <resource_set id="order2-2">
    <resource_ref id="top-rsc"/>
  </resource_set>
</rsc_order>
```

Configure STONITH

Cuprins

13.1. What Is STONITH

STONITH is an acronym for Shoot-The-Other-Node-In-The-Head and it protects your data from being corrupted by rogue nodes or concurrent access.

Just because a node is unresponsive, this doesn't mean it isn't accessing your data. The only way to be 100% sure that your data is safe, is to use STONITH so we can be certain that the node is truly offline, before allowing the data to be accessed from another node.

STONITH mai are un rol pe care îl joacă în cazul în care un serviciu de pe cluster nu poate fi oprit. În acest caz, clusterul foloseşte STONITH pentru a forţa întregul nod offline, astfel făcând să fie sigură pornirea serviciului în altă parte.

13.2. Ce Dispozitiv STONITH Ar Trebui Să Folosiţi

It is crucial that the STONITH device can allow the cluster to differentiate between a node failure and a network one.

The biggest mistake people make in choosing a STONITH device is to use remote power switch (such as many on-board IMPI controllers) that shares power with the node it controls. In such cases, the cluster cannot be sure if the node is really offline, or active and suffering from a network fault.

În mod similar, orice dispozitiv care se bazează pe maşină să fie activă (cum ar fi "dispozitivele" bazate pe SSH folosite în timpul testării) sunt nepotrivite.

13.3. Configurarea STONITH

1. Find the correct driver: **`stonith_admin --list-installed`**

2. Since every device is different, the parameters needed to configure it will vary. To find out the parameters associated with the device, run: **`stonith_admin --metadata --agent type`**

   ```
   The output should be XML formatted text containing additional
   parameter descriptions. We will endevor to make the output more
   friendly in a later version.
   ```

3. Enter the shell crm Create an editable copy of the existing configuration **`cib new stonith`** Create a fencing resource containing a primitive resource with a class of stonith, a type of type and a parameter for each of the values returned in step 2: **`configure primitive …`**

4. If the device does not know how to fence nodes based on their uname, you may also need to set the special **`pcmk_host_map`** parameter. See **`man stonithd`** for details.

5. If the device does not support the list command, you may also need to set the special **pcmk_host_list** and/or **pcmk_host_check** parameters. See **man stonithd** for details.

6. If the device does not expect the victim to be specified with the port parameter, you may also need to set the special **pcmk_host_argument** parameter. See **man stonithd** for details.

7. Upload it into the CIB from the shell: **cib commit stonith**

8. Once the stonith resource is running, you can test it by executing: **stonith_admin --reboot nodename**. Although you might want to stop the cluster on that machine first.

13.4. Exemplu

Assuming we have an chassis containing four nodes and an IPMI device active on 10.0.0.1, then we would chose the fence_ipmilan driver in step 2 and obtain the following list of parameters

Obţinerea unei liste de Parametri STONITH

```
# stonith_admin --metadata -a fence_ipmilan
```

```
<?xml version="1.0" ?>
<resource-agent name="fence_ipmilan" shortdesc="Fence agent for IPMI over LAN">
<longdesc>
fence_ipmilan is an I/O Fencing agent which can be used with machines controlled by IPMI.
 This agent calls support software using ipmitool (http://ipmitool.sf.net/).

To use fence_ipmilan with HP iLO 3 you have to enable lanplus option (lanplus / -P) and
 increase wait after operation to 4 seconds (power_wait=4 / -T 4)</longdesc>
<parameters>
        <parameter name="auth" unique="1">
                <getopt mixed="-A" />
                <content type="string" />
                <shortdesc>IPMI Lan Auth type (md5, password, or none)</shortdesc>
        </parameter>
        <parameter name="ipaddr" unique="1">
                <getopt mixed="-a" />
                <content type="string" />
                <shortdesc>IPMI Lan IP to talk to</shortdesc>
        </parameter>
        <parameter name="passwd" unique="1">
                <getopt mixed="-p" />
                <content type="string" />
                <shortdesc>Password (if required) to control power on IPMI device</shortdesc>
        </parameter>
        <parameter name="passwd_script" unique="1">
                <getopt mixed="-S" />
                <content type="string" />
                <shortdesc>Script to retrieve password (if required)</shortdesc>
        </parameter>
        <parameter name="lanplus" unique="1">
                <getopt mixed="-P" />
                <content type="boolean" />
                <shortdesc>Use Lanplus</shortdesc>
        </parameter>
        <parameter name="login" unique="1">
                <getopt mixed="-l" />
                <content type="string" />
                <shortdesc>Username/Login (if required) to control power on IPMI device</
shortdesc>
        </parameter>
        <parameter name="action" unique="1">
```

```
                <getopt mixed="-o" />
                <content type="string" default="reboot"/>
                <shortdesc>Operation to perform. Valid operations: on, off, reboot, status,
 list, diag, monitor or metadata</shortdesc>
        </parameter>
        <parameter name="timeout" unique="1">
                <getopt mixed="-t" />
                <content type="string" />
                <shortdesc>Timeout (sec) for IPMI operation</shortdesc>
        </parameter>
        <parameter name="cipher" unique="1">
                <getopt mixed="-C" />
                <content type="string" />
                <shortdesc>Ciphersuite to use (same as ipmitool -C parameter)</shortdesc>
        </parameter>
        <parameter name="method" unique="1">
                <getopt mixed="-M" />
                <content type="string" default="onoff"/>
                <shortdesc>Method to fence (onoff or cycle)</shortdesc>
        </parameter>
        <parameter name="power_wait" unique="1">
                <getopt mixed="-T" />
                <content type="string" default="2"/>
                <shortdesc>Wait X seconds after on/off operation</shortdesc>
        </parameter>
        <parameter name="delay" unique="1">
                <getopt mixed="-f" />
                <content type="string" />
                <shortdesc>Wait X seconds before fencing is started</shortdesc>
        </parameter>
        <parameter name="verbose" unique="1">
                <getopt mixed="-v" />
                <content type="boolean" />
                <shortdesc>Verbose mode</shortdesc>
        </parameter>
</parameters>
<actions>
        <action name="on" />
        <action name="off" />
        <action name="reboot" />
        <action name="status" />
        <action name="diag" />
        <action name="list" />
        <action name="monitor" />
        <action name="metadata" />
</actions>
</resource-agent>
```

from which we would create a STONITH resource fragment that might look like this

Exemplu de Resursă STONITH

```
# crm crm(live)# cib new stonith
INFO: stonith shadow CIB created
crm(stonith)# configure primitive impi-fencing stonith::fence_ipmilan \
 params pcmk_host_list="pcmk-1 pcmk-2" ipaddr=10.0.0.1 login=testuser passwd=abc123 \
 op monitor interval="60s"
```

And finally, since we disabled it earlier, we need to re-enable STONITH. At this point we should have the following configuration.

Now push the configuration into the cluster.

```
crm(stonith)# configure property stonith-enabled="true"
```

```
crm(stonith)# configure shownode pcmk-1
node pcmk-2
primitive WebData ocf:linbit:drbd \
    params drbd_resource="wwwdata" \
    op monitor interval="60s"
primitive WebFS ocf:heartbeat:Filesystem \
    params device="/dev/drbd/by-res/wwwdata" directory="/var/www/html" fstype="gfs2"
primitive WebSite ocf:heartbeat:apache \
    params configfile="/etc/httpd/conf/httpd.conf" \
    op monitor interval="1min"
primitive ClusterIP ocf:heartbeat:IPaddr2 \
    params ip="192.168.122.101" cidr_netmask="32" clusterip_hash="sourceip" \
    op monitor interval="30s"primitive ipmi-fencing
 stonith::fence_ipmilan \ params pcmk_host_list="pcmk-1
 pcmk-2" ipaddr=10.0.0.1 login=testuser passwd=abc123 \ op monitor interval="60s"ms
 WebDataClone WebData \
    meta master-max="2" master-node-max="1" clone-max="2" clone-node-max="1" notify="true"
clone WebFSClone WebFS
clone WebIP ClusterIP \
    meta globally-unique="true" clone-max="2" clone-node-max="2"
clone WebSiteClone WebSite
colocation WebSite-with-WebFS inf: WebSiteClone WebFSClone
colocation fs_on_drbd inf: WebFSClone WebDataClone:Master
colocation website-with-ip inf: WebSiteClone WebIP
order WebFS-after-WebData inf: WebDataClone:promote WebFSClone:start
order WebSite-after-WebFS inf: WebFSClone WebSiteClone
order apache-after-ip inf: WebIP WebSiteClone
property $id="cib-bootstrap-options" \
    dc-version="1.1.5-bdd89e69ba545404d02445be1f3d72e6a203ba2f" \
    cluster-infrastructure="openais" \
    expected-quorum-votes="2" \
    stonith-enabled="true" \
    no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
    resource-stickiness="100"
crm(stonith)# cib commit stonithINFO: commited 'stonith' shadow CIB to the cluster
crm(stonith)# quit
bye
```

Status - Aici sunt dragoni

Cuprins

Most users never need to understand the contents of the status section and can be happy with the output from **crm_mon**.

However for those with a curious inclination, this section attempts to provide an overview of its contents.

14.1. Node Status

In addition to the cluster's configuration, the CIB holds an up-to-date representation of each cluster node in the status section.

Exemplu 14.1. O intrare iniţială de status pentru un nod sănătos numit **cl-virt-1**

```
 <node_state id="cl-virt-1" uname="cl-virt-2" ha="active" in_ccm="true" crmd="online"
join="member" expected="member" crm-debug-origin="do_update_resource">
  <transient_attributes id="cl-virt-1"/>
  <lrm id="cl-virt-1"/>
 </node_state>
```

Users are highly recommended *not to modify* any part of a node's state *directly*. The cluster will periodically regenerate the entire section from authoritative sources. So any changes should be done with the tools for those subsystems.

Tabel 14.1. Surse Autoritative pentru Informaţia de Stare

Dataset	Sursă Autoritativă
node_state fields	crmd
transient_attributes tag	attrd
lrm tag	lrmd

The fields used in the **node_state** objects are named as they are largely for historical reasons and are rooted in Pacemaker's origins as the Heartbeat resource manager.

They have remained unchanged to preserve compatibility with older versions.

Tabel 14.2. Câmpuri de Status ale Nodului

Câmp	Descriere
id	Unique identifier for the node. Corosync based clusters use the uname of the machine, Heartbeat clusters use a human-readable (but annoying) UUID.
uname	The node's machine name (output from **uname -n**).

Câmp	Descriere
ha	Flag specifying whether the cluster software is active on the node. Allowed values: **active**, **dead**.
in_ccm	Flag for cluster membership; allowed values: **true**, **false**.
crmd	Flag: is the crmd process active on the node? One of **online**, **offline**.
join	Flag saying whether the node participates in hosting resources. Possible values: **down**, **pending**, **member**, **banned**.
expected	Expected value for **join**.
crm-debug-origin	Diagnostic indicator: the origin of the most recent change(s).

Clusterul foloseşte aceste câmpuri ca să determine dacă, la nivel de nod, nodul este sănătos sau este într-o stare defectuoasă şi trebuie evacuat forţat.

14.2. Atribute Tranziente ale Nodului

Precum *atributele nodului* obişnuite, perechile nume/valoare listate aici ajută de asemenea la descrierea nodului. Totuşi acestea sunt uitate de către cluster atunci când nodul trece în mod offline. Acest lucru poate fi util, de exemplu, când vreţi un nod să se afle în mod standby (să nu poată rula resurse) până la următorul reboot.

În plus faţă de orice valori setează administratorul, clusterul va stoca de asemenea informaţii despre resursele eşuate aici.

Exemplu 14.2. Exemplu de atribute tranziente de nod pentru nodul "cl-virt-1"

```
 <transient_attributes id="cl-virt-1">
   <instance_attributes id="status-cl-virt-1">
      <nvpair id="status-cl-virt-1-pingd" name="pingd" value="3"/>
      <nvpair id="status-cl-virt-1-probe_complete" name="probe_complete" value="true"/>
      <nvpair id="status-cl-virt-1-fail-count-pingd:0" name="fail-count-pingd:0"
value="1"/>
      <nvpair id="status-cl-virt-1-last-failure-pingd:0" name="last-failure-pingd:0"
value="1239009742"/>
   </instance_attributes>
 </transient_attributes>
```

In the above example, we can see that the **pingd:0** resource has failed once, at **Mon Apr 6 11:22:22 2009**. [1] We also see that the node is connected to three "pingd" peers and that all known resources have been checked for on this machine (**probe_complete**).

14.3. Operation History

A node's resource history is held in the **lrm_resources** tag (a child of the **lrm** tag). The information stored here includes enough information for the cluster to stop the resource safely if it is removed from the **configuration** section. Specifically the resource's **id**, **class**, **type** and **provider** are stored.

[1] You can use the standard **date** command to print a human readable of any seconds-since-epoch value: # **date -d @<parameter>number</parameter>**

Exemplu 14.3. O înregistrare a resursei apcstonith

```
<lrm_resource id="apcstonith" type="apcmastersnmp" class="stonith"/>
```

Suplimentar, stocăm ultimul job pentru fiecare combinaţie de **resource, action** şi **interval**. Concatenarea valorilor din această tuplă este folosită pentru a crea id-ul obiectului **lrm_rsc_op**.

Tabel 14.3. Contents of an **lrm_rsc_op** job

Câmp	Descriere
id	Identifier for the job constructed from the resource's **id**, **operation** and **interval**.
call-id	The job's ticket number. Used as a sort key to determine the order in which the jobs were executed.
operation	Acţiunea cu care a fost invocat agentul de resursă.
interval	Frecvenţa, în milisecunde, la care va fi repetată operaţiunea. 0 indică un job unicat.
op-status	The job's status. Generally this will be either 0 (done) or -1 (pending). Rarely used in favor of **rc-code**.
rc-code	The job's result. Refer to *Secțiune B.4, „OCF Return Codes”* for details on what the values here mean and how they are interpreted.
last-run	Indicator de diagnostic. Ora/data locală a maşinii, în secunde de la epoch, la care job-ul a fost executat.
last-rc-change	Indicator de diagnostic. Ora/data locală a maşinii, în secunde de la epocă, la care job-ul a returnat prima oară valoarea **rc-code**.
exec-time	Indicator de diagnostic. Timpul, în milisecunde, pentru care a rulat job-ul
queue-time	Indicator de diagnostic. Timpul, în secunde, pentru care a fost pus job-ul în coada de aşteptare în LRMd
crm_feature_set	

Câmp	Descriere
	Versiunea la care se conformează această descriere a job-ului. Folosită când se procesează **op-digest**.
transition-key	A concatenation of the job's graph action number, the graph number, the expected result and the UUID of the crmd instance that scheduled it. This is used to construct **transition-magic** (below).
transition-magic	A concatenation of the job's **op-status**, **rc-code** and **transition-key**. Guaranteed to be unique for the life of the cluster (which ensures it is part of CIB update notifications) and contains all the information needed for the crmd to correctly analyze and process the completed job. Most importantly, the decomposed elements tell the crmd if the job entry was expected and whether it failed.
op-digest	Un hash MD5 reprezentând parametrii trimişi job-ului. Folosit pentru a detecta schimbările în configuraţie şi a reporni resursele dacă este necesar.
crm-debug-origin	Indicator de diagnostic. Originea valorilor curente.

14.3.1. Exemplu Simplu

Exemplu 14.4. O operaţiune de monitorizare (determina starea curentă a resursei apcstonith)

```
<lrm_resource id="apcstonith" type="apcmastersnmp" class="stonith">
  <lrm_rsc_op id="apcstonith_monitor_0" operation="monitor" call-id="2"
    rc-code="7" op-status="0" interval="0"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    op-digest="2e3da9274d3550dc6526fb24bfcbcba0"
    transition-key="22:2:7:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    transition-magic="0:7;22:2:7:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    last-run="1239008085" last-rc-change="1239008085" exec-time="10" queue-time="0"/>
</lrm_resource>
```

In the above example, the job is a non-recurring monitor operation often referred to as a "probe" for the **apcstonith** resource.

The cluster schedules probes for every configured resource on when a new node starts, in order to determine the resource's current state before it takes any further action.

From the **transition-key**, we can see that this was the 22nd action of the 2nd graph produced by this instance of the crmd (2668bbeb-06d5-40f9-936d-24cb7f87006a).

The third field of the **transition-key** contains a 7, this indicates that the job expects to find the resource inactive.

By looking at the **rc-code** property, we see that this was the case.

Cum acela este singurul job înregistrat pentru acest nod putem concluziona că clusterul a pornit resursa în altă parte.

14.3.2. Exemplu Complex de Istoric al Resurselor

Exemplu 14.5. Istoricul resurselor unei clone pingd cu job-uri multiple

```
<lrm_resource id="pingd:0" type="pingd" class="ocf" provider="pacemaker">
  <lrm_rsc_op id="pingd:0_monitor_30000" operation="monitor" call-id="34"
    rc-code="0" op-status="0" interval="30000"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    transition-key="10:11:0:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239009741" last-rc-change="1239009741" exec-time="10" queue-time="0"/>
  <lrm_rsc_op id="pingd:0_stop_0" operation="stop"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1" call-id="32"
    rc-code="0" op-status="0" interval="0"
    transition-key="11:11:0:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239009741" last-rc-change="1239009741" exec-time="10" queue-time="0"/>
  <lrm_rsc_op id="pingd:0_start_0" operation="start" call-id="33"
    rc-code="0" op-status="0" interval="0"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    transition-key="31:11:0:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239009741" last-rc-change="1239009741" exec-time="10" queue-time="0" />
  <lrm_rsc_op id="pingd:0_monitor_0" operation="monitor" call-id="3"
    rc-code="0" op-status="0" interval="0"
    crm-debug-origin="do_update_resource" crm_feature_set="3.0.1"
    transition-key="23:2:7:2668bbeb-06d5-40f9-936d-24cb7f87006a"
    ...
    last-run="1239008085" last-rc-change="1239008085" exec-time="20" queue-time="0"/>
  </lrm_resource>
```

When more than one job record exists, it is important to first sort them by **`call-id`** before interpreting them.

Once sorted, the above example can be summarized as:

1. O operaţiune non-recurentă de monitorizare returnează 7 (nu rulează), cu un **`call-id`** de 3
2. O operaţiune de oprire returnează 0 (succes), cu un **`call-id`** de 32
3. O operaţiune de pornire returnează 0 (succes), cu un **`call-id`** de 33
4. Un monitor recurent returnează 0 (succes), cu un **`call-id`** de 34

The cluster processes each job record to build up a picture of the resource's state. After the first and second entries, it is considered stopped and after the third it considered active.

Based on the last operation, we can tell that the resource is currently active.

Additionally, from the presence of a **`stop`** operation with a lower **`call-id`** than that of the **`start`** operation, we can conclude that the resource has been restarted. Specifically this occurred as part of actions 11 and 31 of transition 11 from the crmd instance with the key **`2668bbeb…`**. This information can be helpful for locating the relevant section of the logs when looking for the source of a failure.

Multi-Site Clusters and Tickets

Cuprins

15.1. Abstract

Apart from local clusters, Pacemaker also supports multi-site clusters. That means you can have multiple, geographically dispersed sites with a local cluster each. Failover between these clusters can be coordinated by a higher level entity, the so-called **`CTR (Cluster Ticket Registry)`**.

15.2. Challenges for Multi-Site Clusters

Typically, multi-site environments are too far apart to support synchronous communication between the sites and synchronous data replication. That leads to the following challenges:

- How to make sure that a cluster site is up and running?
- How to make sure that resources are only started once?
- How to make sure that quorum can be reached between the different sites and a split brain scenario can be avoided?
- How to manage failover between the sites?
- How to deal with high latency in case of resources that need to be stopped?

In the following sections, learn how to meet these challenges.

15.3. Conceptual Overview

Multi-site clusters can be considered as “overlay” clusters where each cluster site corresponds to a cluster node in a traditional cluster. The overlay cluster can be managed by a **`CTR (Cluster Ticket Registry)`** mechanism. It guarantees that the cluster resources will be highly available across different cluster sites. This is achieved by using so-called **`tickets`** that are treated as failover domain between cluster sites, in case a site should be down.

The following list explains the individual components and mechanisms that were introduced for multi-site clusters in more detail.

15.3.1. Components and Concepts

15.3.1.1. Ticket

"Tickets" are, essentially, cluster-wide attributes. A ticket grants the right to run certain resources on a specific cluster site. Resources can be bound to a certain ticket by **`rsc_ticket`** dependencies. Only if the ticket is available at a site, the respective resources are started. Vice versa, if the ticket is revoked, the resources depending on that ticket need to be stopped.

The ticket thus is similar to a *site quorum*; i.e., the permission to manage/own resources associated with that site.

(One can also think of the current **`have-quorum`** flag as a special, cluster-wide ticket that is granted in case of node majority.)

These tickets can be granted/revoked either manually by administrators (which could be the default for the classic enterprise clusters), or via an automated **`CTR`** mechanism described further below.

A ticket can only be owned by one site at a time. Initially, none of the sites has a ticket. Each ticket must be granted once by the cluster administrator.

The presence or absence of tickets for a site is stored in the CIB as a cluster status. With regards to a certain ticket, there are only two states for a site: **`true`** (the site has the ticket) or **`false`** (the site does not have the ticket). The absence of a certain ticket (during the initial state of the multi-site cluster) is also reflected by the value **`false`**.

15.3.1.2. Dead Man Dependency

A site can only activate the resources safely if it can be sure that the other site has deactivated them. However after a ticket is revoked, it can take a long time until all resources depending on that ticket are stopped "cleanly", especially in case of cascaded resources. To cut that process short, the concept of a **`Dead Man Dependency`** was introduced:

- If the ticket is revoked from a site, the nodes that are hosting dependent resources are fenced. This considerably speeds up the recovery process of the cluster and makes sure that resources can be migrated more quickly.

This can be configured by specifying a **`loss-policy="fence"`** in **`rsc_ticket`** constraints.

15.3.1.3. CTR (Cluster Ticket Registry)

This is for those scenarios where the tickets management is supposed to be automatic (instead of the administrator revoking the ticket somewhere, waiting for everything to stop, and then granting it on the desired site).

A **`CTR`** is a network daemon that handles granting, revoking, and timing out "tickets". The participating clusters would run the daemons that would connect to each other, exchange information on their connectivity details, and vote on which site gets which ticket(s).

A ticket would only be granted to a site once they can be sure that it has been relinquished by the previous owner, which would need to be implemented via a timer in most scenarios. If a site loses connection to its peers, its tickets time out and recovery occurs. After the connection timeout plus the recovery timeout has passed, the other sites are allowed to re-acquire the ticket and start the resources again.

This can also be thought of as a "quorum server", except that it is not a single quorum ticket, but several.

15.3.1.4. Configuration Replication

As usual, the CIB is synchronized within each cluster, but it is not synchronized across cluster sites of a multi-site cluster. You have to configure the resources that will be highly available across the multi-site cluster for every site accordingly.

15.4. Configuring Ticket Dependencies

The **rsc_ticket** constraint lets you specify the resources depending on a certain ticket. Together with the constraint, you can set a **loss-policy** that defines what should happen to the respective resources if the ticket is revoked.

The attribute **loss-policy** can have the following values:

fence
: Fence the nodes that are running the relevant resources.

stop
: Stop the relevant resources.

freeze
: Do nothing to the relevant resources.

demote
: Demote relevant resources that are running in master mode to slave mode.

An example to configure a **rsc_ticket** constraint:

```
<rsc_ticket id="rsc1-req-ticketA" rsc="rsc1" ticket="ticketA" loss-policy="fence"/>
```

This creates a constraint with the ID **rsc1-req-ticketA**. It defines that the resource **rsc1** depends on **ticketA** and that the node running the resource should be fenced in case **ticketA** is revoked.

If resource **rsc1** was a multi-state resource that can run in master or slave mode, you may want to configure that only **rsc1's** master mode depends on **ticketA**. With the following configuration, **rsc1** will be demoted to slave mode if **ticketA** is revoked:

```
<rsc_ticket id="rsc1-req-ticketA" rsc="rsc1" rsc-role="Master" ticket="ticketA" loss-
policy="demote"/>
```

You can create more **rsc_ticket** constraints to let multiple resources depend on the same ticket.

rsc_ticket also supports resource sets. So one can easily list all the resources in one **rsc_ticket** constraint. For example:

```
      <rsc_ticket id="resources-dep-ticketA" ticket="ticketA" loss-policy="fence">
        <resource_set id="resources-dep-ticketA-0" role="Started">
          <resource_ref id="rsc1"/>
          <resource_ref id="group1"/>
          <resource_ref id="clone1"/>
        </resource_set>
        <resource_set id="resources-dep-ticketA-1" role="Master">
          <resource_ref id="ms1"/>
        </resource_set>
      </rsc_ticket>
```

In the example, there are two resource sets for listing the resources with different **roles** in one **rsc_ticket** constraint. There's no dependency between the two resource sets. And there's no

dependency among the resources within a resource set. Each of the resources just depends on **ticketA**.

Referencing resource templates in **rsc_ticket** constraints, and even referencing them within resource sets, is also supported.

If you want other resources to depend on further tickets, create as many constraints as necessary with **rsc_ticket**.

15.5. Managing Multi-Site Clusters

15.5.1. Granting and Revoking Tickets Manually

You can grant tickets to sites or revoke them from sites manually. Though if you want to re-distribute a ticket, you should wait for the dependent resources to cleanly stop at the previous site before you grant the ticket to another desired site.

Use the **crm_ticket** command line tool to grant and revoke tickets.

To grant a ticket to this site:

```
# crm_ticket --ticket ticketA --grant
```

To revoke a ticket from this site:

```
# crm_ticket --ticket ticketA --revoke
```

Important

If you are managing tickets manually. Use the **crm_ticket** command with great care as they cannot help verify if the same ticket is already granted elsewhere.

15.5.2. Granting and Revoking Tickets via a Cluster Ticket Registry

15.5.2.1. Booth

Booth is an implementation of **Cluster Ticket Registry** or so-called **Cluster Ticket Manager**.

Booth is the instance managing the ticket distribution and thus, the failover process between the sites of a multi-site cluster. Each of the participating clusters and arbitrators runs a service, the boothd. It connects to the booth daemons running at the other sites and exchanges connectivity details. Once a ticket is granted to a site, the booth mechanism will manage the ticket automatically: If the site which holds the ticket is out of service, the booth daemons will vote which of the other sites will get the ticket. To protect against brief connection failures, sites that lose the vote (either explicitly or implicitly by being disconnected from the voting body) need to relinquish the ticket after a time-out. Thus, it is made sure that a ticket will only be re-distributed after it has been relinquished by the previous site. The resources that depend on that ticket will fail over to the new site holding the ticket. The nodes that have run the resources before will be treated according to the **loss-policy** you set within the **rsc_ticket** constraint.

Before the booth can manage a certain ticket within the multi-site cluster, you initially need to grant it to a site manually via **`booth client`** command. After you have initially granted a ticket to a site, the booth mechanism will take over and manage the ticket automatically.

Important

The **`booth client`** command line tool can be used to grant, list, or revoke tickets. The **`booth client`** commands work on any machine where the booth daemon is running.

If you are managing tickets via **`Booth`**, only use **`booth client`** for manual intervention instead of **`crm_ticket`**. That can make sure the same ticket will only be owned by one cluster site at a time.

Booth includes an implementation of *Paxos*[1] and *Paxos Lease* algorithm, which guarantees the distributed consensus among different cluster sites.

Notă

`Arbitrator`

Each site runs one booth instance that is responsible for communicating with the other sites. If you have a setup with an even number of sites, you need an additional instance to reach consensus about decisions such as failover of resources across sites. In this case, add one or more arbitrators running at additional sites. Arbitrators are single machines that run a booth instance in a special mode. As all booth instances communicate with each other, arbitrators help to make more reliable decisions about granting or revoking tickets.

An arbitrator is especially important for a two-site scenario: For example, if site **A** can no longer communicate with site **B**, there are two possible causes for that:

- **A** network failure between **A** and **B**.
- Site **B** is down.

However, if site **C** (the arbitrator) can still communicate with site **B**, site **B** must still be up and running.

15.5.2.1.1. Requirements

- All clusters that will be part of the multi-site cluster must be based on Pacemaker.
- Booth must be installed on all cluster nodes and on all arbitrators that will be part of the multi-site cluster.

The most common scenario is probably a multi-site cluster with two sites and a single arbitrator on a third site. However, technically, there are no limitations with regards to the number of sites and the number of arbitrators involved.

[1] http://en.wikipedia.org/wiki/Paxos_algorithm

Nodes belonging to the same cluster site should be synchronized via NTP. However, time synchronization is not required between the individual cluster sites.

15.5.3. General Management of Tickets

Display the information of tickets:

```
# crm_ticket --info
```

Or you can monitor them with:

```
# crm_mon --tickets
```

Display the rsc_ticket constraints that apply to a ticket:

```
# crm_ticket --ticket ticketA --constraints
```

When you want to do maintenance or manual switch-over of a ticket, the ticket could be revoked from the site for any reason, which would trigger the loss-policies. If **`loss-policy="fence"`**, the dependent resources could not be gracefully stopped/demoted, and even, other unrelated resources could be impacted.

The proper way is making the ticket **`standby`** first with:

```
# crm_ticket --ticket ticketA --standby
```

Then the dependent resources will be stopped or demoted gracefully without triggering the loss-policies.

If you have finished the maintenance and want to activate the ticket again, you can run:

```
# crm_ticket --ticket ticketA --activate
```

15.6. For more information

`Multi-site Clusters`*http://doc.opensuse.org/products/draft/SLE-HA/SLE-ha-guide_sd_draft/cha.ha.geo.html*

`Booth`*https://github.com/ClusterLabs/booth*

Anexa A. FAQ

A.1. Istoric

Î: De ce este Proiectul Numit Pacemaker?

R: În primul rând, motivul pentru care nu este denumit CRM este datorită abundenţei de [1] care sunt asociaţi în mod obişnuit cu acele trei litere.

Numele de Pacemaker a provenit de la [2], un bun prieten de-al meu, şi a fost folosit iniţial de către un GUI Java pentru care am creat prototipul în prima parte a anului 2007. Alte angajamente au împiedicat progresul semnificativ al GUI-ului şi când a venit momentul să alegem un nume pentru acest proiect, Lars a sugerat că ar fi o potrivire şi mai bună pentru un CRM independent.

Ideea provine din analogia dintre rolul acestui software şi acela al micului dispozitiv care menţine inima umană pompând. Pacemaker monitorizează clusterul şi intervine când este necesar pentru a asigura operarea fluentă a serviciilor pe care le furnizează.

Au existat un număr de alte nume (şi acronime) aruncate de colo, colo, dar este suficient să spun că "Pacemaker" a fost cel mai bun

Î: De ce a fost Creat Proiectul Pacemaker?

R: Decizia a fost luată de a crea un produs secundar din CRM prin a avea proiectul propriu după lansarea Heartbeat 2.1.3 pentru a

- suporta ambele stive de cluster, Corosync şi Heartbeat, în mod egal
- decupla ciclurile de lansare ale celor două proiecte aflate la stadii foarte diferite ale ciclului vieţii acestora
- adopta graniţe mai clare legate de pachete, tinzând către
- interfeţe mai bune şi mai stabile

A.2. Setup

Î: Care Straturi de Mesagerie sunt Suportate?

R:

- Corosync (*http://www.corosync.org/*)
- Heartbeat (*http://linux-ha.org/*)

Î: Pot Alege care Strat de Mesagerie să îl Folosesc la Momentul Rulării?

R: Da. CRM-ul va detecta în mod automat cine l-a pornit şi se va comporta în concordanţă.

Î: Pot Avea un Cluster Mixt Heartbeat-Corosync?

R: Nu.

Î: Care Strat de Mesagerie ar trebui să îl aleg?

[1] *termeni* [http://en.wikipedia.org/wiki/CRM]
[2] *Kham* [http://khamsouk.souvanlasy.com/]

R: Acest lucru este dscutat în *Anexa D, Instalare*.

Î: De Unde Pot Obţine Pachete Pre-Compilate?

R: Pachete oficiale pentru majoritatea distribuţiilor majore bazate pe .rpm sunt disponibile de pe WebSite-ul ClusterLabs[3].

Pentru pachete Debian, compilarea din sursă şi detalii asupra folosirii repositoarelor de mai sus, vedeți pagina noastră de instalare[4].

Î: Care Versiuni de Pacemaker sunt Suportate?

R: Vă rugam să consultaţi pagina de Releases[5] pentru o listă la zi a versiunilor suportate în mod direct de către proiect.

Când căutaţi asistenţă, vă rugăm să vă asiguraţi că aveţi una din versiunile acestea.

[3] *http://www.clusterlabs.org/rpm*
[4] *http://clusterlabs.org/wiki/Install*
[5] *http://clusterlabs.org/wiki/Releases*

Anexa B. Mai Multe Despre Agenţii de Resursă OCF

Cuprins

B.1. Locaţia Scripturilor Personalizate

OCF Resource Agents are found in */usr/lib/ocf/resource.d/**provider***.

When creating your own agents, you are encouraged to create a new directory under */usr/lib/ocf/resource.d/* so that they are not confused with (or overwritten by) the agents shipped with Heartbeat.

So, for example, if you chose the provider name of bigCorp and wanted a new resource named bigApp, you would create a script called */usr/lib/ocf/resource.d/bigCorp/bigApp* and define a resource:

```
<primitive id="custom-app" class="ocf" provider="bigCorp" type="bigApp"/>
```

B.2. Acţiuni

Toţi Agenţii de Resursă OCF sunt obligaţi să implementeze următoarele acţiuni

Tabel B.1. Acţiuni Necesare pentru Agenţii OCF

Acţiune	Descriere	Instrucţiuni
start	Porneşte resursa	Return 0 on success and an appropriate error code otherwise. Must not report success until the resource is fully active.
stop	Opreşte resursa	Return 0 on success and an appropriate error code otherwise. Must not report success until the resource is fully stopped.
monitor	Check the resource's state	Exit 0 if the resource is running, 7 if it is stopped, and anything else if it is failed. NOTĂ: Scriptul de monitorizare ar trebui să testeze starea resursei numai pe maşina locală.
meta-data	Descrie resursa	Provide information about this resource as an XML snippet. Exit with 0. NOTE: This is **not** performed as root.
validate-all	Verifică dacă parametrii furnizaţi sunt corecţi	Exit with 0 if parameters are valid, 2 if not valid, 6 if resource is not configured.

Cerinţe suplimentare (care nu sunt parte din specificaţia OCF) sunt plasate pe agenţi care vor fi folosiţi pentru concepte avansate cum ar fi *clone* şi resurse *multi-state*.

Tabel B.2. Acţiuni Opţionale pentru Agenţi OCF

Acţiune	Descriere	Instrucţiuni
`promote`	Promovează instanţa locală a unei resurse multi-state la starea master/primară	Return 0 on success
`demote`	Retrogradează instanţa locală a unei resurse multi-state la starea slave/secundară	Return 0 on success
`notify`	Folosit de către cluster pentru a trimite agentului notificări pre şi post eveniment spunându-i resursei ceea ce se întâmplă sau ce tocmai s-a întâmplat	Must not fail. Must exit with 0

O acţiune specificată în specificaţiile OCF nu este folosită de către cluster în mod curent

- **`recover`** - o variantă a acţiunii **`start`**, aceasta ar trebui să recupereze o resursă local.

Remember to use **`ocf-tester`** to verify that your new agent complies with the OCF standard properly.

B.3. Cum sunt Interpretate Codurile de Ieşire OCF?

Primul lucru pe care îl face clusterul este să verifice codul de ieşire faţă de rezultatul aşteptat. Dacă rezultatul nu se potriveşte cu valoarea aşteptată, atunci este considerat că operaţiunea a eşuat şi acţiunea de recuperare este iniţiată.

Sunt trei tipuri de recuperare în caz de eşec:

Tabel B.3. Tipuri de recuperare realizate de către cluster

Tip	Descriere	Acţiunea Luată de către Cluster
`soft`	O eroare tranzientă a avut loc	Restart the resource or move it to a new location
`hard`	O eroare non-tranzientă s-a produs care ar putea fi specifică nodului curent	Move the resource elsewhere and prevent it from being retried on the current node
`fatal`	O eroare non-tranzientă care va fi comună pe toate nodurile clusterului (ex.: o configuraţie greşită a fost specificată)	Stop the resource and prevent it from being started on any cluster node

Plecând de la presupunerea că o acţiune este considerată că ar fi eşuat, următorul tabel evidenţiază diferitele coduri de ieşire OCF şi tipul de recuperare pe care o va iniţia clusterul când acest cod este primit.

B.4. OCF Return Codes

Tabel B.4. Codurile de Ieşire OCF şi Cum Sunt Ele Gestionate

RC	Alias OCF	Descriere	RT
`0`	**`OCF_SUCCESS`**	Success. The command completed successfully. This is the expected result for all start, stop, promote and demote commands.	**`soft`**

RC	Alias OCF	Descriere	RT
1	**OCF_ERR_GENERIC**	Generic "there was a problem" error code.	**soft**
2	**OCF_ERR_ARGS**	The resource's configuration is not valid on this machine. Eg. refers to a location/ tool not found on the node.	**hard**
3	**OCF_ERR_UNIMPLEMENTED**	The requested action is not implemented.	**hard**
4	**OCF_ERR_PERM**	The resource agent does not have sufficient privileges to complete the task.	**hard**
5	**OCF_ERR_INSTALLED**	The tools required by the resource are not installed on this machine.	**hard**
6	**OCF_ERR_CONFIGURED**	The resource's configuration is invalid. Eg. required parameters are missing.	**fatal**
7	**OCF_NOT_RUNNING**	The resource is safely stopped. The cluster will not attempt to stop a resource that returns this for any action.	**N/A**
8	**OCF_RUNNING_MASTER**	The resource is running in **Master** mode.	**soft**
9	**OCF_FAILED_MASTER**	The resource is in **Master** mode but has failed. The resource will be demoted, stopped and then started (and possibly promoted) again.	**soft**
other	**NA**	Custom error code.	**soft**

Although counterintuitive, even actions that return 0 (aka. **OCF_SUCCESS**) can be considered to have failed.

B.5. Excepţii

- Acţiunile de monitorizare nerecurente (probele) care găsesc o resursă activă (sau în starea Master) nu vor rezulta într-o acţiune de recuperare decât dacă este găsită activă în altă parte
- The recovery action taken when a resource is found active more than once is determined by the *multiple-active* property of the resource
- Recurring actions that return **OCF_ERR_UNIMPLEMENTED** do not cause any type of recovery

Anexa C. Ce S-a Schimbat în 1.0

Cuprins

C.1. Nou

- Intervale de aşteptare înainte de a declara un eşec. Vedeţi *Secţiune 9.3.2, „Mutarea Resurselor Datorită Eşecului”*
- O nouă secţiune pentru valorile implicite ale resurselor şi operaţiunilor. Vedeţi *Secţiune 5.5, „Setarea de Valori Implicite Globale pentru Opţiunile Clusterului”* şi *Secţiune 5.7.2, „Setarea de Valori Implicite Globale pentru Operaţiuni”*
- Utilitar pentru realizarea de modificări ale configuraţiei offline. Vedeţi *Secţiune 2.6, „Realizând Modificări de Configurare într-un Sandbox”*
- **`Rules, instance_attributes, meta_atributes`** şi seturile de operaţiuni pot fi definite o dată iar apoi pot fi referenţiate în mai multe locuri. Vedeţi *Secţiune 9.4, „Refolosirea Regulilor, Opţiunilor şi a Setului de Operaţiuni”*
- CIB-ul acum acceptă operaţiuni de creare/modificare/ştergere bazate pe tipul XPath. Vedeţi ajutorul textual pentru **`cibadmin`**.
- Restricţii de ordonare şi colocare multi-dimensională. Vedeţi *Secţiune 6.5, „Ordonarea Seturilor de Resurse”* şi *Secţiune 6.9, „Colocarea Seturilor de Resurse”*
- Posibilitatea de conectare la CIB de pe maşini care nu fac parte din cluster. Vedeţi *Secţiune 9.1, „Conectarea de pe o Maşină la Distanţă”*
- Permite declanşarea de acţiuni recurente la intervale cunoscute de timp. Vedeţi *Secţiune 9.2, „Specificând Când Acţiunile Recurente sunt Efectuate”*

C.2. Modificat

- Sintaxă
  - Toate opţiunile şi resursele cluster-ului folosesc acum cratime (-) în loc de underscore (_)
  - **`master_slave`** a fost redenumit în **`master`**
  - Tag-ul **`attributes`** a fost scos
  - Câmpul operaţiunii **`pre-req`** fost redenumit în **`requires`**
  - Toate operaţiunile trebuie să aibe un **`interval`**, cele de **`start`**/**`stop`** trebuie să îl aibe setat la zero
- Opţiunea **`stonith-enabled`** are acum valoarea implicită setată pe true.
- Clusterul va refuza să pornească resurse dacă **`stonith-enabled`** este true (sau nu este setat) şi nici o resursă STONITH nu a fost definită

- Atributele restricţiilor de colocare şi ordonare au fost redenumite pentru o mai bună claritate. Vedeţi *Secţiune 6.3, „Specificând Ordinea în care Resursele ar Trebui să Pornească/Oprească”* şi *Secţiune 6.4, „Plasarea Resurselor Relativă la alte Resurse”*
- **`resource-failure-stickiness`** a fost înlocuită de **`migration-threshold`**. Vedeţi *Secţiune 9.3.2, „Mutarea Resurselor Datorită Eşecului”*
- Argumentele pentru utilitarele folosite în linia de comandă au fost aduse la o formă consistentă
- Switched to *RelaxNG* schema validation and *libxml2* parser
  - câmpurile id sunt acum ID-uri XML care au următoarele limitări:
    - id’s cannot contain colons (:)
    - id’s cannot begin with a number
    - id’s must be globally unique (not just unique for that tag)
  - Some fields (such as those in constraints that refer to resources) are IDREFs.

    This means that they must reference existing resources or objects in order for the configuration to be valid. Removing an object which is referenced elsewhere will therefore fail.
  - The CIB representation, from which a MD5 digest is calculated to verify CIBs on the nodes, has changed.

    This means that every CIB update will require a full refresh on any upgraded nodes until the cluster is fully upgraded to 1.0. This will result in significant performance degradation and it is therefore highly inadvisable to run a mixed 1.0/0.6 cluster for any longer than absolutely necessary.
- Ping node information no longer needs to be added to *ha.cf*.

  Simply include the lists of hosts in your ping resource(s).

C.3. Scoase

- Sintaxă
  - Nu mai este posibil să fie setate meta opţiunile resursei ca attribute de nivel înalt. Folosiţi meta atributele în schimb.
  - Valorile implicite ale resurselor si operaţiunilor nu mai sunt citite din **`crm_config`**. Vedeţi *Secţiune 5.5, „Setarea de Valori Implicite Globale pentru Opţiunile Clusterului”* şi *Secţiune 5.7.2, „Setarea de Valori Implicite Globale pentru Operaţiuni”* în schimb.

Anexa D. Instalare

Cuprins

Avertisment

The following text may no longer be accurate in some places.

D.1. Choosing a Cluster Stack

Ultimately the choice of cluster stack is a personal decision that must be made in the context of you or your company's needs and strategic direction. Pacemaker currently functions equally well with both stacks.

În continuare sunt câţiva factori care ar putea influenţa decizia

- SUSE/Novell, Red Hat şi Oracle îşi pun la contribuţie greutatea colectivă pentru a susţine stiva de cluster Corosync.
- Folosirea Corosync oferă aplicaţiilor dvs. acces la următoarele servicii de cluster adiţionale
  - serviciu de blocare distribuit
  - serviciu de sincronie virtuală extinsă
  - serviciu de grup de procese închise de cluster
- Este probabil ca Pacemaker, la un moment dat în viitor, să utilizeze o parte din aceste servicii adiţionale care nu sunt furnizate de Heartbeat

D.2. Activarea Pacemaker

D.2.1. Pentru Corosync

The Corosync configuration is normally located in */etc/corosync/corosync.conf* and an example for a machine with an address of **`1.2.3.4`** in a cluster communicating on port 1234 (without peer authentication and message encryption) is shown below.

Un exemplu de fişier de configurare al Corosync

```
totem {
    version: 2
```

```
      secauth: off
      threads: 0
      interface {
          ringnumber: 0
          bindnetaddr: 1.2.3.4
          mcastaddr: 239.255.1.1
          mcastport: 1234
      }
  }
  logging {
      fileline: off
      to_syslog: yes
      syslog_facility: daemon
  }
  amf {
      mode: disabled
  }
```

Partea de loguri ar trebui să fie evidentă în cea mai mare parte şi secţiunea amf se referă la Availability Management Framework şi nu este acoperit în acest document.

The interesting part of the configuration is the totem section. This is where we define how the node can communicate with the rest of the cluster and what protocol version and options (including encryption [1]) it should use. Beginners are encouraged to use the values shown and modify the interface section based on their network.

Este totodată posibilă configurarea Corosync pentru un mediu bazat pe IPV6. Pur şi simplu configuraţi **`bindnetaddr`** şi **`mcastaddr`** cu echivalentele IPV6 ale acestora. ex.

Exemple de opţiuni pentru un mediu IPV6

```
  bindnetaddr: fec0::1:a800:4ff:fe00:20
  mcastaddr: ff05::1
```

Pentru a îi spune Corosync-ului să folosească Pacemaker ca şi manager al clusterului, adăugaţi următorul fragment la o configuraţie funcţională de Corosync şi reporniţi clusterul.

Fragment de configurare pentru a activa Pacemaker sub Corosync

```
aisexec {
    user:  root
    group: root
}
service {
    name: pacemaker
    ver: 0
}
```

The cluster needs to be run as root so that its child processes (the **`lrmd`** in particular) have sufficient privileges to perform the actions requested of it. After all, a cluster manager that can't add an IP address or start apache is of little use.

A doua directivă este cea care instruieşte de fapt clusterul să ruleze Pacemaker.

[1] Please consult the Corosync website (*http://www.corosync.org/*) and documentation for details on enabling encryption and peer authentication for the cluster.

D.2.2. Pentru Heartbeat

Add the following to a functional *ha.cf* configuration file and restart Heartbeat:

Fragment de configurare pentru a activa Pacemaker sub Heartbeat

```
crm respawn
```

Anexa E. Actualizarea Soft-ului de Cluster

Cuprins

E.1. Compatibilitatea Versiunii

Când lansăm versiuni noi avem grijă că suntem compatibili invers cu versiunile anterioare. În timp ce veţi putea oricând să actualizaţi de la versiunea x la x+1, pentru a putea să continuăm producţia de soft de înaltă calitate ar putea fi necesar în mod ocazional să renunţăm la compatibilitatea cu versiunile mai vechi.

Întotdeauna va exista o cale de actualizare de la oricare produs lansat în seria-2 la oricare alt produs lansat în aceeaşi serie-2.

Sunt trei alternative în scopul actualizării soft-ului de cluster

- Oprirea Completă a Clusterului
- Secvenţial (nod după nod)
- Deconectează şi Reataşează

Fiecare metodă are avantaje şi dezavantaje, unele dintre acestea sunt listate în tabelul de mai jos şi ar trebui să o alegeţi pe aceea cel mai apropiată de nevoile voastre.

Tabel E.1. Sumar al Metodologiilor de Actualizare

Tip	Disponibil între toate versiunile de soft	Indisponibilitate Serviciului pe Durata Actualizării	Recuperarea Serviciului pe Durata Actualizării	Exersează Logica/ Configuraţia de Failover	Allows change of cluster stack type [1]
Shutdown	da	întotdeauna	N/A	nu	da
Rolling	nu	întotdeauna	da	da	nu
Reattach	da	doar din cauza eşecului	nu	nu	da

[1] For example, switching from Heartbeat to Corosync. Consult the Heartbeat or Corosync documentation to see if upgrading them to a newer version is also supported.

E.2. Oprirea Completă a Clusterului

În acest scenariu se închid toate nodurile şi resursele clusterului şi se actualizează toate nodurile înainte de a reporni clusterul.

E.2.1. Procedură

1. Pe fiecare nod:
   a. Închideţi stiva de cluster (Heartbeat sau Corosync)
   b. Actualizaţi soft-ul Pacemaker. Acest lucru poate include actualizarea stivei de cluster şi/sau a sistemului de operare de bază.
   c. Check the configuration manually or with the **`crm_verify`** tool if available.
2. Pe fiecare nod:
   a. Porniţi stiva de cluster. Aceasta poate fi oricare dintre Corosync sau Heartbeat şi nu trebuie să fie aceeaşi ca stiva anterioară de cluster.

E.3. Secvenţial (nod după nod)

În acest scenariu fiecare nod este scos din cluster, actualizat şi apoi adus înapoi online până ce toate nodurile rulează pe cea mai recentă versiune.

Important

Această metodă este în prezent nefuncţională între Pacemaker 0.6.x şi 1.0.x

Measures have been put into place to ensure rolling upgrades always work for versions after 1.0.0. Please try one of the other upgrade strategies. Detach/Reattach is a particularly good option for most people.

E.3.1. Procedură

On each node: . Shutdown the cluster stack (Heartbeat or Corosync) . Upgrade the Pacemaker software. This may also include upgrading the cluster stack and/or the underlying operating system. .. On the first node, check the configuration manually or with the **`crm_verify`** tool if available. .. Start the cluster stack.

+ This must be the same type of cluster stack (Corosync or Heartbeat) that the rest of the cluster is using. Upgrading Corosync/Heartbeat may also be possible, please consult the documentation for those projects to see if the two versions will be compatible.

+ .. Repeat for each node in the cluster.

E.3.2. Compatibilitatea Versiunii

Tabel E.2. Tabel cu Compatibilitatea Versiunilor

Versiunea care este Instalată	Cea mai Veche Versiune Compatibilă
Pacemaker 1.0.x	Pacemaker 1.0.0

Versiunea care este Instalată	Cea mai Veche Versiune Compatibilă
Pacemaker 0.7.x	Pacemaker 0.6 sau Heartbeat 2.1.3
Pacemaker 0.6.x	Heartbeat 2.0.8
Heartbeat 2.1.3 (sau mai mică)	Heartbeat 2.0.4
Heartbeat 2.0.4 (sau mai mică)	Heartbeat 2.0.0
Heartbeat 2.0.0	Niciuna. Folosiţi o strategie de actualizare alternativă.

E.3.3. Trecerea Graniţelor de Compatibilitate

Actualizările secvenţiale care trec de graniţele compatibilităţii trebuie efectuate în paşi multipli. De exemplu, pentru a efectua o actualizare secvenţială de la Heartbeat 2.0.1 la Pacemaker 0.6.6 trebuie să:

1. Efectueaze o actualizare secvenţială de la Heartbeat 2.0.1 la Heartbeat 2.0.4
2. Efectueaze o actualizare secvenţială de la Heartbeat 2.0.4 la Heartbeat 2.1.3
3. Efectueaze o actualizare secvenţială de la Heartbeat 2.1.3 la Pacemaker 0.6.6

E.4. Deconectează şi Reataşează

O variantă de închidere completă a clusterului, dar resursele sunt lăsate active şi re-detectate când clusterul este repornit.

E.4.1. Procedură

1. Tell the cluster to stop managing services.

   This is required to allow the services to remain active after the cluster shuts down.

   ```
   # crm_attribute -t crm_config -n is-managed-default -v false
   ```

2. Pentru orice resursă care are o valoare pentru **`is-managed`**, vă rugăm să vă asiguraţi că este setată pe **`false`** (astfel încât clusterul să nu o oprească)

   ```
   # crm_resource -t primitive -r $rsc_id -p is-managed -v false
   ```

3. Pe fiecare nod:

   a. Închideţi stiva de cluster (Heartbeat sau Corosync)

   b. Actualizaţi stiva de cluster - Acest lucru poate include actualizarea sistemului de operare de bază.

4. Check the configuration manually or with the **`crm_verify`** tool if available.

5. Pe fiecare nod:

   a. Start the cluster stack.

      This can be either Corosync or Heartbeat and does not need to be the same as the previous cluster stack.

6. Verificaţi că, clusterul a re-detectat toate resursele în mod corect

7. Permiteţi clusterului să reia gestionarea resurselor

```
# crm_attribute -t crm_config -n is-managed-default -v true
```

8. Pentru orice resursă care are o valoare pentru **`is-managed`** resetaţi-o, dacă doriţi, pe **`true`** (astfel încât clusterul să poată recupera serviciul dacă acesta eşuează)

```
# crm_resource -t primitive -r $rsc_id -p is-managed -v true
```

E.4.2. Menţiuni

Important

Întotdeauna verificaţi configuraţia existentă şi că este in continuare compatibilă cu versiunea pe care o instalaţi înainte de a porni clusterul.

Notă

Cea mai veche versiune de CRM care suportă acest tip de actualizare a fost Heartbeat 2.0.4

Anexa F. Actualizarea Configuraţiei de la 0.6

Cuprins

F.1. Pregătire

Download the latest *DTD*[1] and ensure your configuration validates.

F.2. Realizaţi actualizarea

F.2.1. Actualizaţi software-ul

Consultaţi anexa: *Anexa E, Actualizarea Soft-ului de Cluster*

F.2.2. Actualizaţi Configuraţia

Cum XML-ul nu este cel mai prietenos dintre limbaje, este obişnuit pentru administratorii de cluster să fi scriptat unele dintre activităţile acestora. În astfel de cazuri, este probabil ca acele scripturi să nu funcţioneze cu noua sintaxă 1.0.

Pentru a suporta astfel de medii, este chiar posibilă continuarea folosirii sintaxei vechi de 0.6.

Partea nefastă însă, este că nu toate funcţionalităţile noi vor fi disponibile şi este un impact de performanţă din moment ce clusterul trebuie să execute o actualizare non-persistentă a configuraţiei înainte de fiecare tranziţie. Deci în timp ce folosirea sintaxei vechi este posibilă, nu este recomandată folosirea acesteia pe termen nelimitat.

Chiar dacă doriţi să continuaţi folosirea sintaxei vechi, este recomandat să urmaţi procedura de actualizare pentru a vă asigura că clusterul este capabil să folosească configuraţia existentă (din moment ce va efectua în mare parte aceeaşi sarcină intern).

1. Create a shadow copy to work with

```
# crm_shadow --create upgrade06
```

2. Verify the configuration is valid

[1] http://hg.clusterlabs.org/pacemaker/stable-1.0/file-raw/tip/xml/crm.dtd

```
# crm_verify --live-check
```

3. Reparaţi orice erori sau avertismente
4. Realizaţi actualizarea

```
# cibadmin --upgrade
```

5. Dacă acest pas eşuează, sunt trei posibilităţi principale
   a. Configuraţia nu a fost validă de la început - mergeţi înapoi la pasul 2
   b. The transformation failed - report a bug or *email the project*[2]
   c. The transformation was successful but produced an invalid result [3]

      If the result of the transformation is invalid, you may see a number of errors from the validation library. If these are not helpful, visit *http://clusterlabs.org/wiki/Validation_FAQ* and/or try the procedure described below under *Secţiune F.2.3, „Actualizarea Manuală a Configuraţiei”*
6. Verificaţi modificările

```
# crm_shadow --diff
```

   Dacă la acest punct există orice legat de actualizare ce doriţi să reglaţi fin (de exemplu, să schimbaţi unele din ID-urile automate) acum este momentul să realizaţi acest lucru. Din moment ce configuraţia ascunsă nu este folosită de către cluster, este neprimejdios să editaţi fişierul manual:

```
# crm_shadow --edit
```

   This will open the configuration in your favorite editor (whichever is specified by the standard **$EDITOR** environment variable)
7. Previzualizaţi cum va reacţiona clusterul

   Testaţi ce va face clusterul când încărcaţi noua configuraţie

```
# crm_simulate --live-check --save-dotfile upgrade06.dot -S
# graphviz upgrade06.dot
```

   Verify that either no resource actions will occur or that you are happy with any that are scheduled. If the output contains actions you do not expect (possibly due to changes to the score calculations), you may need to make further manual changes. See *Secţiune 2.7, „Testarea Modificărilor Voastre de Configurare”* for further details on how to interpret the output of **crm_simulate**
8. Încărcaţi modificările

[2] mailto:pacemaker@oss.clusterlabs.org?subject=Transformation%20failed%20during%20upgrade
[3] The most common reason is ID values being repeated or invalid. Pacemaker 1.0 is much stricter regarding this type of validation.

```
# crm_shadow --commit upgrade06 --force
```

Dacă acest pas eşuează, ceva cu adevărat ciudat s-a întâmplat. Ar trebui să raportaţi bug-ul.

F.2.3. Actualizarea Manuală a Configuraţiei

It is also possible to perform the configuration upgrade steps manually. To do this

Locate the *upgrade06.xsl* conversion script or download the latest version from *Git*[4]

1. Convert the XML blob:

```
# xsltproc /path/to/upgrade06.xsl config06.xml > config10.xml
```

2. Locate the *pacemaker.rng* script.

3. Check the XML validity:

```
# xmllint --relaxng /path/to/pacemaker.rng config10.xml
```

Avantajul acestei metode este acela că poate fi efectuată fară ca şi clusterul să funcţioneze şi orice erori de validare ar trebui să fie cu caracter mai informativ (în ciuda faptului că sunt generate de aceeaşi bibliotecă!) din moment ce includ numerele liniilor.

[4] https://github.com/ClusterLabs/pacemaker/tree/master/xml/upgrade06.xsl

Anexa G. Este acest script de init compatibil LSB?

The relevant part of *LSB spec*[1] includes a description of all the return codes listed here.

Sub presupunerea că **`some_service`** este configurat corect şi nu este activ în momentul de faţă, următoarea secvenţă vă poate ajuta să determinaţi dacă este compatibil LSB:

1. Porneşte (oprit):

```
# /etc/init.d/some_service start ; echo "result: $?"
```

   a. A pornit serviciul?

   b. A printat comanda rezultatul de ieşire: 0 (suplimentar faţă de rezultatul contextual normal)?

2. Status (rulând):

```
# /etc/init.d/some_service status ; echo "result: $?"
```

   a. A acceptat scriptul comanda?

   b. A indicat scriptul dacă serviciul rula?

   c. A printat comanda rezultatul de ieşire: 0 (suplimentar faţă de rezultatul contextual normal)?

3. Porneşte (rulând):

```
# /etc/init.d/some_service start ; echo "result: $?"
```

   a. Serviciul încă rulează?

   b. A printat comanda rezultatul de ieşire: 0 (suplimentar faţă de rezultatul contextual normal)?

4. Opreşte (rulând):

```
# /etc/init.d/some_service stop ; echo "result: $?"
```

   a. A fost oprit serviciul?

   b. A printat comanda rezultatul de ieşire: 0 (suplimentar faţă de rezultatul contextual normal)?

5. Status (oprit):

```
# /etc/init.d/some_service status ; echo "result: $?"
```

   a. A acceptat scriptul comanda?

   b. A indicat scriptul faptul că serviciul nu rula?

[1] http://refspecs.freestandards.org/LSB_3.1.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html

c. A printat comanda rezultatul de ieşire: 3 (suplimentar faţă de rezultatul contextual normal)?

6. Opreşte (oprit):

```
# /etc/init.d/some_service stop ; echo "result: $?"
```

a. Este serviciul în continuare oprit?

b. A printat comanda rezultatul de ieşire: 0 (suplimentar faţă de rezultatul contextual normal)?

7. Status (eşuat):

Acest pas nu este gata pentru a fi testat şi se bazează pe inspecţia manuală a scriptului.

Scriptul poate folosi unul din codurile de eroare (altul decât 3) listate în specificaţia LSB pentru a indica faptul că este activ dar eşuat. Acesta îi spune clusterului ca înainte de a muta resursa pe alt nod, trebuie să o oprească mai întâi pe cel existent.

Dacă răspunsul la oricare din întrebările de mai sus este nu, atunci scriptul nu este compatibil LSB. Opţiunile disponibile în acel moment sunt fie de a repara scriptul fie de a scrie un agent OCF bazat pe scriptul existent.

Anexa H. Exemple de Configurare

Cuprins

H.1. Empty

Exemplu H.1. O Configuraţie Goală

```
<cib admin_epoch="0" epoch="0" num_updates="0" have-quorum="false">
  <configuration>
    <crm_config/>
    <nodes/>
    <resources/>
    <constraints/>
  </configuration>
  <status/>
</cib>
```

H.2. Simple

Exemplu H.2. Simple Configuration - 2 nodes, some cluster options and a resource

```
<cib admin_epoch="0" epoch="1" num_updates="0" have-quorum="false"
    validate-with="pacemaker-1.0">
  <configuration>
    <crm_config>
      <nvpair id="option-1" name="symmetric-cluster" value="true"/>
      <nvpair id="option-2" name="no-quorum-policy" value="stop"/>
    </crm_config>
    <op_defaults>
      <nvpair id="op-default-1" name="timeout" value="30s"/>
    </op_defaults>
    <rsc_defaults>
      <nvpair id="rsc-default-1" name="resource-stickiness" value="100"/>
      <nvpair id="rsc-default-2" name="migration-threshold" value="10"/>
    </rsc_defaults>
    <nodes>
      <node id="xxx" uname="c001n01" type="normal"/>
      <node id="yyy" uname="c001n02" type="normal"/>
    </nodes>
    <resources>
      <primitive id="myAddr" class="ocf" provider="heartbeat" type="IPaddr">
        <operations>
          <op id="myAddr-monitor" name="monitor" interval="300s"/>
        </operations>
        <instance_attributes>
          <nvpair name="ip" value="10.0.200.30"/>
        </instance_attributes>
      </primitive>
    </resources>
    <constraints>
      <rsc_location id="myAddr-prefer" rsc="myAddr" node="c001n01" score="INFINITY"/>
```

```
    </constraints>
  </configuration>
  <status/>
</cib>
```

În acest exemplu, avem o resursă (o adresă IP) pe care o verificăm la fiecare cinci minute şi va rula pe host-ul **`c001n01`** până fie resursa eşuează de 10 ori fie host-ul este închis.

H.3. Advanced Configuration

Exemplu H.3. Advanced configuration - groups and clones with stonith

```
<cib admin_epoch="0" epoch="1" num_updates="0" have-quorum="false"
    validate-with="pacemaker-1.0">
  <configuration>
    <crm_config>
      <nvpair id="option-1" name="symmetric-cluster" value="true"/>
      <nvpair id="option-2" name="no-quorum-policy" value="stop"/>
      <nvpair id="option-3" name="stonith-enabled" value="true"/>
    </crm_config>
    <op_defaults>
      <nvpair id="op-default-1" name="timeout" value="30s"/>
    </op_defaults>
    <rsc_defaults>
      <nvpair id="rsc-default-1" name="resource-stickiness" value="100"/>
      <nvpair id="rsc-default-2" name="migration-threshold" value="10"/>
    </rsc_defaults>
    <nodes>
      <node id="xxx" uname="c001n01" type="normal"/>
      <node id="yyy" uname="c001n02" type="normal"/>
      <node id="zzz" uname="c001n03" type="normal"/>
    </nodes>
    <resources>
      <primitive id="myAddr" class="ocf" provider="heartbeat" type="IPaddr">
        <operations>
          <op id="myAddr-monitor" name="monitor" interval="300s"/>
        </operations>
        <instance_attributes>
          <nvpair name="ip" value="10.0.200.30"/>
        </instance_attributes>
      </primitive>
      <group id="myGroup">
        <primitive id="database" class="lsb" type="oracle">
          <operations>
            <op id="database-monitor" name="monitor" interval="300s"/>
          </operations>
        </primitive>
        <primitive id="webserver" class="lsb" type="apache">
          <operations>
            <op id="webserver-monitor" name="monitor" interval="300s"/>
          </operations>
        </primitive>
      </group>
      <clone id="STONITH">
        <meta_attributes id="stonith-options">
          <nvpair id="stonith-option-1" name="globally-unique" value="false"/>
        </meta_attributes>
        <primitive id="stonithclone" class="stonith" type="external/ssh">
          <operations>
            <op id="stonith-op-mon" name="monitor" interval="5s"/>
          </operations>
          <instance_attributes id="stonith-attrs">
```

```
            <nvpair id="stonith-attr-1" name="hostlist" value="c001n01,c001n02"/>
          </instance_attributes>
        </primitive>
      </clone>
    </resources>
    <constraints>
      <rsc_location id="myAddr-prefer" rsc="myAddr" node="c001n01"
        score="INFINITY"/>
      <rsc_colocation id="group-with-ip" rsc="myGroup" with-rsc="myAddr"
        score="INFINITY"/>
    </constraints>
  </configuration>
  <status/>
</cib>
```

Anexa I. Documentaţie Adiţională

- Site-ul Proiectului *http://www.corosync.org/*
- Documentaţia Proiectului *http://www.clusterlabs.org/wiki/Documentation*
- A comprehensive guide to cluster commands has been written by Novell[1]
- Configuraţia Heartbeat: *http://www.linux-ha.org/*
- Configuraţia Corosync: *http://www.corosync.org/*

[1] *http://www.suse.com/documentation/sle_ha/book_sleha/data/book_sleha.html*

Anexa J. Istoricul Reviziilor

Versiune 1-1 **19 Oct 2009** **Andrew Beekhof** ***andrew@beekhof.net***

Import din Pages.app

Versiune 2-1 **26 Oct 2009** **Andrew Beekhof** ***andrew@beekhof.net***

Curăţarea şi reformatarea xml-ului pentru docbook terminată

Versiune 3-1 **Tue Nov 12 2009** **Andrew Beekhof** ***andrew@beekhof.net***

Împărţirea cărţii în capitole şi trecerea validării
Re-organize book for use with *Publican*[1]

Versiune 4-1 **Mon Oct 8 2012** **Andrew Beekhof** ***andrew@beekhof.net***

Converted to *asciidoc*[2] (which is converted to docbook for use with *Publican*[3])

[1] https://fedorahosted.org/publican/
[2] http://www.methods.co.nz/asciidoc
[3] https://fedorahosted.org/publican/

Index

Simboluri

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y