Geo Clustering Guide

SUSE Linux Enterprise High Availability Extension 15

Geo Clustering Guide

SUSE Linux Enterprise High Availability Extension 15

by Tanja Roth and Thomas Schraitle

This document covers the setup options and parameters for Geo clusters and their components, such as booth ticket manager, the specific Csync2 setup, and the configuration of the required cluster resources (and how to transfer them to other sites in case of changes). Learn how to monitor and manage Geo clusters from command line or with the Hawk2 Web interface.

Publication Date: 07/11/2018

SUSE LLC
10 Canal Park Drive
Suite 200
Cambridge MA 02141
USA
https://www.suse.com/documentation

Contents

1 Challenges for Geo Clusters

Typically, Geo environments are too far apart to support synchronous communication between the sites. 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 keep the CIB up to date on all nodes and sites?
- 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 with SUSE Linux Enterprise High Availability Extension.

2 Conceptual Overview

Geo clusters based on SUSE® Linux Enterprise High Availability Extension can be considered “overlay” clusters where each cluster site corresponds to a cluster node in a traditional cluster. The overlay cluster is managed by the booth cluster ticket manager (in the following called booth).

Each of the parties involved in a Geo cluster runs a service, the `boothd`. It connects to the booth daemons running at the other sites and exchanges connectivity details. For making cluster resources highly available across sites, booth relies on cluster objects called tickets. A ticket grants the right to run certain resources on a specific cluster site. Booth guarantees that every ticket is granted to no more than one site at a time.

If the communication between two booth instances breaks down, it might be because of a network breakdown between the cluster sites *or* because of an outage of one cluster site. In this case, you need an additional instance (a third cluster site or an `arbitrator`) to reach consensus about decisions (such as failover of resources across sites). Arbitrators are single machines (outside of the clusters) that run a booth instance in a special mode. Each Geo cluster can have one or multiple arbitrators.

The most common scenario probably is a Geo cluster with two sites and a single arbitrator on a third site. This requires three booth instances.

It is also possible to run a two-site Geo cluster *without* an arbitrator. In this case, a Geo cluster administrator needs to manually manage the tickets. If a ticket should be granted to more than one site at the same time, booth displays a warning.

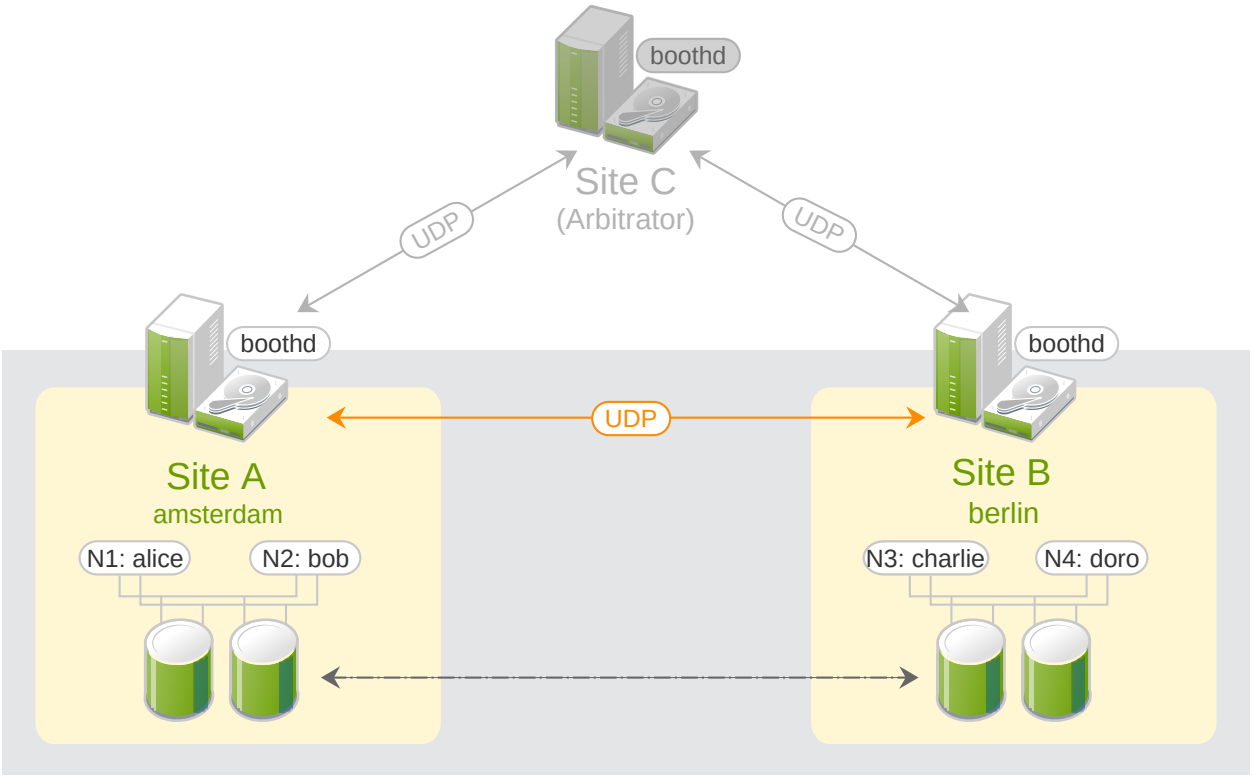

FIGURE 2.1: TWO-SITE CLUSTER—2X2 NODES + ARBITRATOR (OPTIONAL)

The following list explains the components and mechanisms for Geo clusters in more detail.

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, it is useful to have 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. Arbitrators cannot hold any 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.

Booth Cluster Ticket Manager

Booth is the instance managing the ticket distribution, and thus, the failover process between the sites of a Geo 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. After a ticket has been granted to a site, the booth mechanism can manage the ticket automatically: If the site that 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 redistributed after it has been relinquished by the previous site. See also *Dead Man Dependency (`loss-policy="fence"`)*.

For a Geo cluster with two sites and arbitrator, you need 3 booth instances: one instance per site plus the instance running on the arbitrator.

Note: Limited Number of Booth Instances

The upper limit is (currently) 16 booth instances.

Dead Man Dependency (`loss-policy="fence"`)

After a ticket is revoked, it can take a long time until all resources depending on that ticket are stopped, especially in case of cascaded resources. To cut that process short, the cluster administrator can configure a `loss-policy` (together with the ticket dependencies) for the case that a ticket gets revoked from a site. If the loss-policy is set to `fence`, the nodes that are hosting dependent resources are fenced.

Warning: Potential Loss of Data

On the one hand, `loss-policy="fence"` considerably speeds up the recovery process of the cluster and makes sure that resources can be migrated more quickly.

On the other hand, it can lead to loss of all unwritten data, such as:

- Data lying on shared storage.
- Data in a replicating database (for example, MariaDB or PostgreSQL) or on a replicating device (DRBD), where the data has not yet reached the other site because of a slow network link.

Ticket

A ticket grants the right to run certain resources on a specific cluster site. 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. After that, tickets are managed by the booth for automatic failover of resources. But administrators may also intervene and grant or revoke tickets manually.

After a ticket is administratively revoked, it is not managed by booth anymore. For booth to start managing the ticket again, the ticket must be again granted to a site.

Resources can be bound to a certain ticket by dependencies. Only if the defined ticket is available at a site, the respective resources are started. Vice versa, if the ticket is removed, the resources depending on that ticket are automatically stopped.

The presence or absence of tickets for a site is stored in the CIB as a cluster status. With regard 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 Geo cluster) is not treated differently from the situation after the ticket has been revoked. Both are reflected by the value `false`.

A ticket within an overlay cluster is similar to a resource in a traditional cluster. But in contrast to traditional clusters, tickets are the only type of resource in an overlay cluster. They are primitive resources that do not need to be configured or cloned.

Ticket Failover

After you have initially granted an automatic ticket to a site, booth will manage this ticket automatically. If the site holding a ticket should be out of service, the ticket is automatically revoked after the expiry time. If the remaining sites have quorum, the ticket will be granted to another site (fail over). The resources that depend on that ticket fail over to the new site that holds the ticket. The `loss-policy` (which is defined within the constraint) specifies what happens to the nodes that have run the resources before.

If automatic failover in case of a split brain scenario is *not* required, administrators can also grant manual tickets to the healthy site. How to manage tickets from command line is described in *Section 8.1*.

3 Requirements

SOFTWARE REQUIREMENTS

- All machines (cluster nodes and arbitrators) that will be part of the cluster need at least the following modules and extensions:
  - Base System Module 15
  - Server Applications Module 15
  - SUSE Linux Enterprise High Availability Extension 15
- When installing the machines, select `HA GEO Node` as `system role`. This leads to the installation of a minimal system where the packages from the pattern `Geo Clustering for High Availability (ha_geo)` are installed by default.

NETWORK REQUIREMENTS

- The virtual IPs to be used for each cluster site must be accessible across the Geo cluster.
- The sites must be reachable on one UDP and TCP port per booth instance. That means any firewalls or IPsec tunnels in between must be configured accordingly.
- Other setup decisions may require to open more ports (for example, for DRBD or database replication).

OTHER REQUIREMENTS AND RECOMMENDATIONS

- All cluster nodes on all sites should synchronize to an NTP server outside the cluster. For more information, see the *Administration Guide* for SUSE Linux Enterprise Server 15, available at http://www.suse.com/documentation/sles. Refer to the chapter *Time Synchronization with NTP*.
  If nodes are not synchronized, log files and cluster reports are very hard to analyze.
- Use an *uneven* number of sites in your Geo cluster. In case the network connection breaks down, this makes sure that there still is a majority of sites (to avoid a split brain scenario). In case you have an even number of cluster sites, use an arbitrator for handling automatic failover of tickets. If you do not use an arbitrator, you need to handle ticket failover manually.
- The cluster on each site has a meaningful name, for example: `amsterdam` and `berlin`.

The cluster names for each site are defined in the respective `/etc/corosync/corosync.conf` files:

```
totem {
    [...]
    cluster_name: amsterdam
    }
```

Change the name with following crmsh command:

```
root # crm cluster rename NEW_NAME
```

Stop and start the `pacemaker` service for the changes to take effect:

```
root # systemctl stop pacemaker
root # systemctl start pacemaker
```

4 Setting Up the Booth Services

This chapter describes the setup and configuration options for booth, how to synchronize the booth configuration to all sites and arbitrators, how to enable and start the booth services, and how to reconfigure booth while its services are running.

4.1 Booth Configuration and Setup Options

The default booth configuration is `/etc/booth/booth.conf`. This file must be the same on all sites of your Geo cluster, including the arbitrator or arbitrators. To keep the booth configuration synchronous across all sites and arbitrators, use Csync2, as described in *Section 4.5, "Synchronizing the Booth Configuration to All Sites and Arbitrators"*.

Note: Ownership of `/etc/booth` and Files

The directory `/etc/booth` and all files therein need to belong to the user `hacluster` and the group `haclient`. Whenever you copy a new file from this directory, use the option `-p` for the **`cp`** command to preserve the ownership. Alternatively, when you create a new file, set the user and group afterward with **`chown`** `hacluster:haclient` *`FILE`*.

For setups including multiple Geo clusters, it is possible to "share" the same arbitrator (as of SUSE Linux Enterprise High Availability Extension 12). By providing several booth configuration files, you can start multiple booth instances on the same arbitrator, with each booth instance running on a different port. That way, you can use *one* machine to serve as arbitrator for *different* Geo clusters. For details on how to configure booth for multiple Geo clusters, refer to *Section 4.4, "Using a Multi-Tenant Booth Setup"*.

To prevent malicious parties from disrupting the booth service, you can configure authentication for talking to booth, based on a shared key. For details, see ❺ in *Example 4.1, "A Booth Configuration File"*. All hosts that communicate with various booth servers need this key. Therefore make sure to include the key file in the Csync2 configuration or to synchronize it manually across all parties.

4.2 Automatic versus Manual Tickets

A ticket grants the right to run certain resources on a specific cluster site. Two types of tickets are supported:

- Automatic tickets are controlled by the `boothd` daemon.
- Manual tickets are managed by the cluster administrator only.

Automatic and manual tickets have the following properties:

- **Automatic and manual tickets can be defined together.** You can define and use both automatic and manual tickets within the same Geo cluster.
- **Manual ticket management remains manual.** The automatic ticket management is not applied to manually controlled tickets. Manual tickets do not require any quorum elections, cannot fail over automatically, and do not have an expiry time.
- **Manual tickets will not be moved automatically.** Tickets which were manually granted to a site will remain there until they are manually revoked. Even if a site goes offline, the ticket will not be moved to another site. This behavior ensures that the services that depend on a ticket remain on a particular site and are not moved to another site.
- **Same commands for managing both types of tickets.** The manual tickets are managed by the same commands as automatic tickets (`grant` or `revoke`, for example).
- **Arbitrators are not needed if only manual tickets are used.** If you configure only manual tickets in a Geo cluster, arbitrators are not necessary, because manual ticket management does not require quorum decisions.

To configure tickets, use the `/etc/booth/booth.conf` configuration file (see *Section 4.3, "Using the Default Booth Setup"* for further information).

4.3 Using the Default Booth Setup

If you have set up your basic Geo cluster with the `ha-cluster-bootstrap` scripts as described in the Geo Clustering Quick Start, the scripts have created a default booth configuration on all sites with a minimal set of parameters. To extend or fine-tune the minimal booth configuration, have a look at *Example 4.1* or at the examples in *Section 4.4, "Using a Multi-Tenant Booth Setup"*.

To add or change parameters needed for booth, either edit the booth configuration files manually or use the YaST *Geo Cluster* module. To access the YaST module, start it from command line with **`yast2 geo-cluster`** (or start YaST and select *High Availability* › *Geo Cluster*).

EXAMPLE 4.1: A BOOTH CONFIGURATION FILE

```
transport = UDP (1)
port = 9929 (2)
arbitrator = 192.168.203.100 (3)
site =  192.168.201.100 (4)
site =  192.168.202.100 (4)
authfile = /etc/booth/authkey (5)
ticket = "ticket-nfs" (6)
     mode = MANUAL (7)
ticket = "ticketA" (6)
     expire = 600 (8)
     timeout = 10 (9)
     retries = 5 (10)
     renewal-freq = 30 (11)
     before-acquire-handler (12) = /etc/booth/ticket-A (13) db-1 (14)
     acquire-after = 60 (15)
ticket = "ticketB" (6)
     expire = 600 (8)
     timeout = 10 (9)
     retries = 5 (10)
     renewal-freq = 30 (11)
     before-acquire-handler (12) = /etc/booth/ticket-B (13) db-8 (14)
     acquire-after = 60 (15)
```

(1) The transport protocol used for communication between the sites. Only UDP is supported, but other transport layers will follow in the future. Currently, this parameter can therefore be omitted.

(2) The port to be used for communication between the booth instances at each site. When not using the default port (`9929`), choose a port that is not already used for different services. Make sure to open the port in the nodes' and arbitrators' firewalls. The booth clients use TCP to communicate with the `boothd`. Booth will always bind and listen to both UDP and TCP ports.

(3) The IP address of the machine to use as arbitrator. Add an entry for each arbitrator you use in your Geo cluster setup.

4 The IP address used for the `boothd` on a site. Add an entry for each site you use in your Geo cluster setup. Make sure to insert the correct virtual IP addresses (`IPaddr2`) for each site, otherwise the booth mechanism will not work correctly. Booth works with both IPv4 and IPv6 addresses.
If you have set up booth with the `ha-cluster-bootstrap` scripts, the virtual IPs you have specified during setup have been written to the booth configuration already (and have been added to the cluster configuration, too). To set up the cluster resources manually, see *Section 6.2, "Configuring a Resource Group for* `boothd`*"*.

5 Optional parameter. Enables booth authentication for clients and servers on the basis of a shared key. This parameter specifies the path to the key file.

KEY REQUIREMENTS

- The key can be either binary or text.
  If it is text, the following characters are ignored: leading and trailing white space, new lines.
- The key must be between 8 and 64 characters long.
- The key must belong to the user `hacluster` and the group `haclient`.
- The key must be readable only by the file owner.

6 The tickets to be managed by booth or a cluster administrator. For each ticket, add a `ticket` entry. For example, the ticket `ticket-nfs` specified here can be used for failover of NFS and DRBD as explained in http://www.suse.com/documentation/suse-best-practices/sbp-drbd/data/sbp-drbd.html.

7 Optional parameter. Defines the ticket mode. By default, all tickets are managed by booth. To define tickets which are managed by the administrator (*manual tickets*), set the `mode` parameter to `MANUAL` or `manual`.
Manual tickets do not have `expire`, `renewal-freq`, and `retries` parameters.

8 Optional parameter. Defines the ticket's expiry time in seconds. A site that has been granted a ticket will renew the ticket regularly. If booth does not receive any information about renewal of the ticket within the defined expiry time, the ticket will be revoked and granted to another site. If no expiry time is specified, the ticket will expire after `600` seconds by default. The parameter should not be set to a value less than 120 seconds. The default value set by the `ha-cluster-init` scripts is `600`.

9 Optional parameter. Defines a timeout period in seconds. After that time, booth will resend packets if it did not receive a reply within this period. The timeout defined should be long enough to allow packets to reach other booth members (all arbitrators and sites).

10 Optional parameter. Defines how many times booth retries sending packets before giving up waiting for confirmation by other sites. Values smaller than `3` are invalid and will prevent booth from starting.

11 Optional parameter. Sets the ticket renewal frequency period. Ticket renewal occurs every half expiry time by default. If the network reliability is often reduced over prolonged periods, it is advisable to renew more often. Before every renewal the `before-acquire-handler` is run.

12 Optional parameter. It supports one or more scripts. To use more than one script, each script can be responsible for different checks, like cluster state, data center connectivity, environment health sensors, and more. Store all scripts in the directory `/etc/booth.d/`*`TICKET_NAME`* and make sure they have the correct ownership (user `hacluster` and group `haclient`). Assign the directory name as a value to the parameter `before-acquire-handler`.

The scripts in this directory are executed in alphabetical order. All scripts will be called before `boothd` tries to acquire or renew a ticket. For the ticket to be granted or renewed, *all* scripts must succeed. The semantics are the same as for a single script: On exit code other than `0`, `boothd` relinquishes the ticket.

13 The `/usr/share/booth/service-runnable` script is included in the product as an example. To use it, link it into the respective "ticket" directory:

```
root # ln -s /usr/share/booth/service-runnable /etc/booth.d/TICKET_NAME
```

Assume that the `/etc/booth.d`*`TICKET_NAME`* directory contains the **`service-runnable`** script. This simple script is based on **`crm_simulate`**. It can be used to test whether a particular cluster resource *can* be run on the current cluster site. That means, it checks if the cluster is healthy enough to run the resource (all resource dependencies are fulfilled, the cluster partition has quorum, no dirty nodes, etc.). For example, if a service in the dependency-chain has a failcount of `INFINITY` on all available nodes, the service cannot be run on that site. In that case, it is of no use to claim the ticket.

14 The resource to be tested by the `before-acquire-handler` (in this case, by the `service-runnable` script). You need to reference the resource that is protected by the respective ticket. In this example, resource `db-1` is protected by `ticketA` whereas `db-8` is protected by `ticketB`. The resource for DRBD (`ms_drbd_nfs`) is protected by the ticket `ticket-nfs`.

15 Optional parameter. After a ticket is lost, booth will wait this time in addition before acquiring the ticket. This is to allow for the site that lost the ticket to relinquish the resources, by either stopping them or fencing a node. A typical delay might be `60` seconds, but ultimately it depends on the protected resources and the fencing configuration. The default value is `0`.

If you are unsure how long stopping or demoting the resources or fencing a node may take (depending on the `loss-policy`), use this parameter to prevent resources from running on two sites at the same time.

4.3.1 Manually Editing The Booth Configuration File

1. Log in to a cluster node as `root` or equivalent.
2. If `/etc/booth/booth.conf` does not exist yet, copy the example booth configuration file `/etc/booth/booth.conf.example` to `/etc/booth/booth.conf`:

```
root # cp -p /etc/booth/booth.conf.example /etc/booth/booth.conf
```

3. Edit `/etc/booth/booth.conf` according to *Example 4.1, "A Booth Configuration File"*.
4. Verify your changes and save the file.
5. On all cluster nodes and arbitrators, open the port in the firewall that you have configured for booth. See *Example 4.1, "A Booth Configuration File"*, position 2.

4.3.2 Setting Up Booth with YaST

1. Log in to a cluster node as `root` or equivalent.
2. Start the YaST *Geo Cluster* module.

3. Choose to *Edit* an existing booth configuration file or click *Add* to create a new booth configuration file:

   a. In the screen that appears configure the following parameters:

      - **Configuration File.** A name for the booth configuration file. YaST suggests `booth` by default. This results in the booth configuration being written to `/etc/booth/booth.conf`. Only change this value if you need to set up multiple booth instances for different Geo clusters as described in *Section 4.4, "Using a Multi-Tenant Booth Setup"*.

      - **Transport.** The transport protocol used for communication between the sites. Only UDP is supported, but other transport layers will follow in the future. See also *Example 4.1, "A Booth Configuration File"*, position ❶.

      - **Port.** The port to be used for communication between the booth instances at each site. See also *Example 4.1, "A Booth Configuration File"*, position ❷.

      - **Arbitrator.** The IP address of the machine to use as arbitrator. See also *Example 4.1, "A Booth Configuration File"*, position ❸.
        To specify an *Arbitrator*, click *Add*. In the dialog that opens, enter the IP address of your arbitrator and click *OK*.

      - **Site.** The IP address used for the `boothd` on a site. See also *Example 4.1, "A Booth Configuration File"*, position ❹.
        To specify a *Site* of your Geo cluster, click *Add*. In the dialog that opens, enter the IP address of one site and click *OK*.

      - **Ticket.** The tickets to be managed by booth or a cluster administrator. See also *Example 4.1, "A Booth Configuration File"*, position ❻.

To specify a *Ticket*, click *Add*. In the dialog that opens, enter a unique *Ticket* name. If you need to define multiple tickets with the same parameters and values, save configuration effort by creating a "ticket template" that specifies the default parameters and values for all tickets. To do so, use `__default__` as *Ticket* name.

- **Authentication.** To enable authentication for booth, click *Authentication* and in the dialog that opens, activate *Enable Security Auth*. If you already have an existing key, specify the path and file name in *Authentication file*. To generate a key file for a new Geo cluster, click *Generate Authentication Key File*. The key will be created and written to the location specified in *Authentication file*.
  Additionally, you can specify optional parameters for your ticket. For an overview, see *Example 4.1, "A Booth Configuration File"*, positions 7 to 15.
  Click *OK* to confirm your changes.

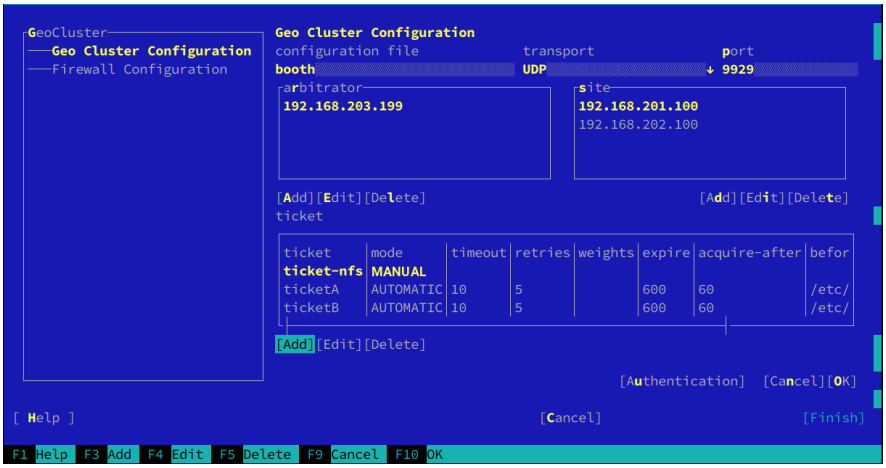

FIGURE 4.1: EXAMPLE TICKET DEPENDENCY

b. Click *OK* to close the current booth configuration screen. YaST shows the name of the booth configuration file that you have defined.

4. Before closing the YaST module, switch to the *Firewall Configuration* category.

5. To open the port you have configured for booth, enable *Open Port in Firewall*.

Important: Firewall Setting for Local Machine Only

The firewall setting is only applied to the current machine. It will open the UDP/TCP ports for all ports that have been specified in `/etc/booth/booth.conf` or any other booth configuration files (see *Section 4.4, "Using a Multi-Tenant Booth Setup"*).

Make sure to open the respective ports on all other cluster nodes and arbitrators of your Geo cluster setup, too. Do so either manually or by synchronizing the following files with Csync2:

- `/usr/lib/firewalld`
- `/usr/lib/firewalld/services/booth.xml`

6. Click *Finish* to confirm all settings and close the YaST module. Depending on the *NAME* of the *Configuration File* specified in *Step 3.a*, the configuration is written to `/etc/booth/NAME.conf`.

4.4 Using a Multi-Tenant Booth Setup

For setups including multiple Geo clusters, it is possible to "share" the same arbitrator (as of SUSE Linux Enterprise High Availability Extension 12). By providing several booth configuration files, you can start multiple booth instances on the same arbitrator, with each booth instance running on a different port. That way, you can use *one* machine to serve as arbitrator for *different* Geo clusters.

Let us assume you have two Geo clusters, one in EMEA (Europe, the Middle East and Africa), and one in the Asia-Pacific region (APAC).

To use the same arbitrator for both Geo clusters, create two configuration files in the `/etc/booth` directory: `/etc/booth/emea.conf` and `/etc/booth/apac.conf`. Both must minimally differ in the following parameters:

- The port used for the communication of the booth instances.
- The sites belonging to the different Geo clusters that the arbitrator is used for.

EXAMPLE 4.2: `/etc/booth/apac.conf`

```
transport = UDP (1)
port = 9133 (2)
arbitrator = 192.168.203.100 (3)
site = 192.168.2.254 (4)
site = 192.168.1.112 (4)
authfile = /etc/booth/authkey-apac (5)
ticket ="tkt-db-apac-intern" (6)
     timeout = 10
     retries = 5
     renewal-freq = 60
     before-acquire-handler (12) = /usr/share/booth/service-runnable (13) db-apac-intern (14)
ticket = "tkt-db-apac-cust" (6)
     timeout = 10
     retries = 5
     renewal-freq = 60
     before-acquire-handler = /usr/share/booth/service-runnable db-apac-cust
```

EXAMPLE 4.3: `/etc/booth/emea.conf`

```
transport = UDP (1)
port = 9150 (2)
arbitrator = 192.168.203.100 (3)
site = 192.168.201.100 (4)
site = 192.168.202.100 (4)
authfile = /etc/booth/authkey-emea (5)
ticket = "tkt-sap-crm" (6)
     expire = 900
     renewal-freq = 60
     before-acquire-handler (12) = /usr/share/booth/service-runnable (13) sap-crm (14)
ticket = "tkt-sap-prod" (6)
     expire = 600
     renewal-freq = 60
     before-acquire-handler = /usr/share/booth/service-runnable sap-prod
```

(1) The transport protocol used for communication between the sites. Only UDP is supported, but other transport layers will follow in the future. Currently, this parameter can therefore be omitted.

(2) The port to be used for communication between the booth instances at each site. The configuration files use different ports to allow for start of multiple booth instances on the same arbitrator.

(3) The IP address of the machine to use as arbitrator. In the examples above, we use the same arbitrator for different Geo clusters.

4 The IP address used for the `boothd` on a site. The sites defined in both booth configuration files are different, because they belong to two different Geo clusters.

5 Optional parameter. Enables booth authentication for clients and servers on the basis of a shared key. This parameter specifies the path to the key file. Use different key files for different tenants.

KEY REQUIREMENTS

- The key can be either binary or text.
  If it is text, the following characters are ignored: leading and trailing white space, new lines.
- The key must be between 8 and 64 characters long.
- The key must belong to the user `hacluster` and the group `haclient`.
- The key must be readable only by the file owner.

6 The tickets to be managed by booth or a cluster administrator. Theoretically the same ticket names can be defined in different booth configuration files—the tickets will not interfere because they are part of different Geo clusters that are managed by different booth instances. However, (for better overview) we advise to use distinct ticket names for each Geo cluster as shown in the examples above.

12 Optional parameter. If set, the specified command will be called before `boothd` tries to acquire or renew a ticket. On exit code other than `0`, `boothd` relinquishes the ticket.

13 The `service-runnable` script referenced here is included in the product as an example. It is a simple script based on **`crm_simulate`**. It can be used to test whether a particular cluster resource *can* be run on the current cluster site. That means, it checks if the cluster is healthy enough to run the resource (all resource dependencies are fulfilled, the cluster partition has quorum, no dirty nodes, etc.). For example, if a service in the dependency-chain has a failcount of `INFINITY` on all available nodes, the service cannot be run on that site. In that case, it is of no use to claim the ticket.

14 The resource to be tested by the `before-acquire-handler` (in this case, by the `service-runnable` script). You need to reference the resource that is protected by the respective ticket.

PROCEDURE 4.1: USING THE SAME ARBITRATOR FOR DIFFERENT GEO CLUSTERS

1. Create different booth configuration files in `/etc/booth` as shown in *Example 4.2, "/etc/booth/apac.conf"* and *Example 4.3, "/etc/booth/emea.conf"*. Do so either manually or with YaST, as outlined in *Section 4.3.2, "Setting Up Booth with YaST"*.
2. On the arbitrator, open the ports that are defined in any of the booth configuration files in `/etc/booth`.
3. On the nodes belonging to the individual Geo clusters that the arbitrator is used for, open the port that is used for the respective booth instance.
4. Synchronize the respective booth configuration files across all cluster nodes and arbitrators that use the same booth configuration. For details, see *Section 4.5, "Synchronizing the Booth Configuration to All Sites and Arbitrators"*.
5. On the arbitrator, start the individual booth instances as described in *Starting the Booth Services on Arbitrators* for multi-tenancy setups.
6. On the individual Geo clusters, start the booth service as described in *Starting the Booth Services on Cluster Sites*.

4.5 Synchronizing the Booth Configuration to All Sites and Arbitrators

Note: Use the Same Booth Configuration On All Sites and Arbitrators

To make booth work correctly, all cluster nodes and arbitrators within one Geo cluster must use the same booth configuration.

You can use Csync2 to synchronize the booth configuration. For details, see *Section 5.1, "Csync2 Setup for Geo Clusters"* and *Section 5.2, "Synchronizing Changes with Csync2"*.

In case of any booth configuration changes, make sure to update the configuration files accordingly on all parties and to restart the booth services as described in *Section 4.7, "Reconfiguring Booth While Running"*.

4.6 Enabling and Starting the Booth Services

Starting the Booth Services on Cluster Sites

The booth service for each cluster site is managed by the booth resource group (that has either been configured automatically if you used the `ha-cluster-init` scripts for Geo cluster setup, or manually as described in *Section 6.2, "Configuring a Resource Group for* `boothd`*"*). To start one instance of the booth service per site, start the respective booth resource group on each cluster site.

Starting the Booth Services on Arbitrators

Starting with SUSE Linux Enterprise 12, booth arbitrators are managed with systemd. The unit file is named `booth@.service`. The `@` denotes the possibility to run the service with a parameter, which is in this case the name of the configuration file.

To *enable* the booth service on an arbitrator, use the following command:

```
root # systemctl enable booth@booth
```

After the service has been enabled from command line, YaST Services Manager can be used to manage the service (as long as the service is not disabled). In that case, it will disappear from the service list in YaST the next time systemd is restarted.

The command to *start* the booth service depends on your booth setup:

- If you are using the default setup as described in *Section 4.3*, only `/etc/booth/booth.conf` is configured. In that case, log in to each arbitrator and use the following command:

  ```
  root # systemctl start booth@booth
  ```

- If you are running booth in multi-tenancy mode as described in *Section 4.4*, you have configured multiple booth configuration files in `/etc/booth`. To start the services for the individual booth instances, use **`systemctl start booth@`***`NAME`*, where *`NAME`* stands for the name of the respective configuration file `/etc/booth/`*`NAME`*`.conf`.

  For example, if you have the booth configuration files `/etc/booth/emea.conf` and `/etc/booth/apac.conf`, log in to your arbitrator and execute the following commands:

  ```
  root # systemctl start booth@emea
  root # systemctl start booth@apac
  ```

This starts the booth service in arbitrator mode. It can communicate with all other booth daemons but in contrast to the booth daemons running on the cluster sites, it cannot be granted a ticket. Booth arbitrators take part in elections only. Otherwise, they are dormant.

4.7 Reconfiguring Booth While Running

In case you need to change the booth configuration while the booth services are already running, proceed as follows:

1. Adjust the booth configuration files as desired.
2. Synchronize the updated booth configuration files to all cluster nodes and arbitrators that are part of your Geo cluster. For details, see *Chapter 5, Synchronizing Configuration Files Across All Sites and Arbitrators*.
3. Restart the booth services on the arbitrators and cluster sites as described in *Section 4.6, "Enabling and Starting the Booth Services"*. This does not have any effect on tickets that have already been granted to sites.

5 Synchronizing Configuration Files Across All Sites and Arbitrators

To replicate important configuration files across all nodes in the cluster and across Geo clusters, use Csync2. Csync2 can handle any number of hosts, sorted into synchronization groups. Each synchronization group has its own list of member hosts and its include/exclude patterns that define which files should be synchronized in the synchronization group. The groups, the host names belonging to each group, and the include/exclude rules for each group are specified in the Csync2 configuration file, `/etc/csync2/csync2.cfg`.

For authentication, Csync2 uses the IP addresses and pre-shared keys within a synchronization group. You need to generate one key file for each synchronization group and copy it to all group members.

Csync2 will contact other servers via a TCP port (by default `6556`), and start remote Csync2 instances. For detailed information about Csync2, refer to http://oss.linbit.com/csync2/paper.pdf

5.1 Csync2 Setup for Geo Clusters

How to set up Csync2 for individual clusters with YaST is explained in *Book "Administration Guide", Chapter 4 "Using the YaST Cluster Module", Section 4.5 "Transferring the Configuration to All Nodes"*. However, YaST cannot handle more complex Csync2 setups, like those that are needed for Geo clusters. For the following setup, as shown in *Figure 5.1, "Example Csync2 Setup for Geo Clusters"*, configure Csync2 manually by editing the configuration files.

To adjust Csync2 for synchronizing files not only within local clusters but also across geographically dispersed sites, you need to define two synchronization groups in the Csync2 configuration:

- A global group `ha_global` (for the files that need to be synchronized globally, across all sites and arbitrators belonging to a Geo cluster).
- A group for the local cluster site `ha_local` (for the files that need to be synchronized within the local cluster).

For an overview of the multiple Csync2 configuration files for the two synchronization groups, see *Figure 5.1, "Example Csync2 Setup for Geo Clusters"*.

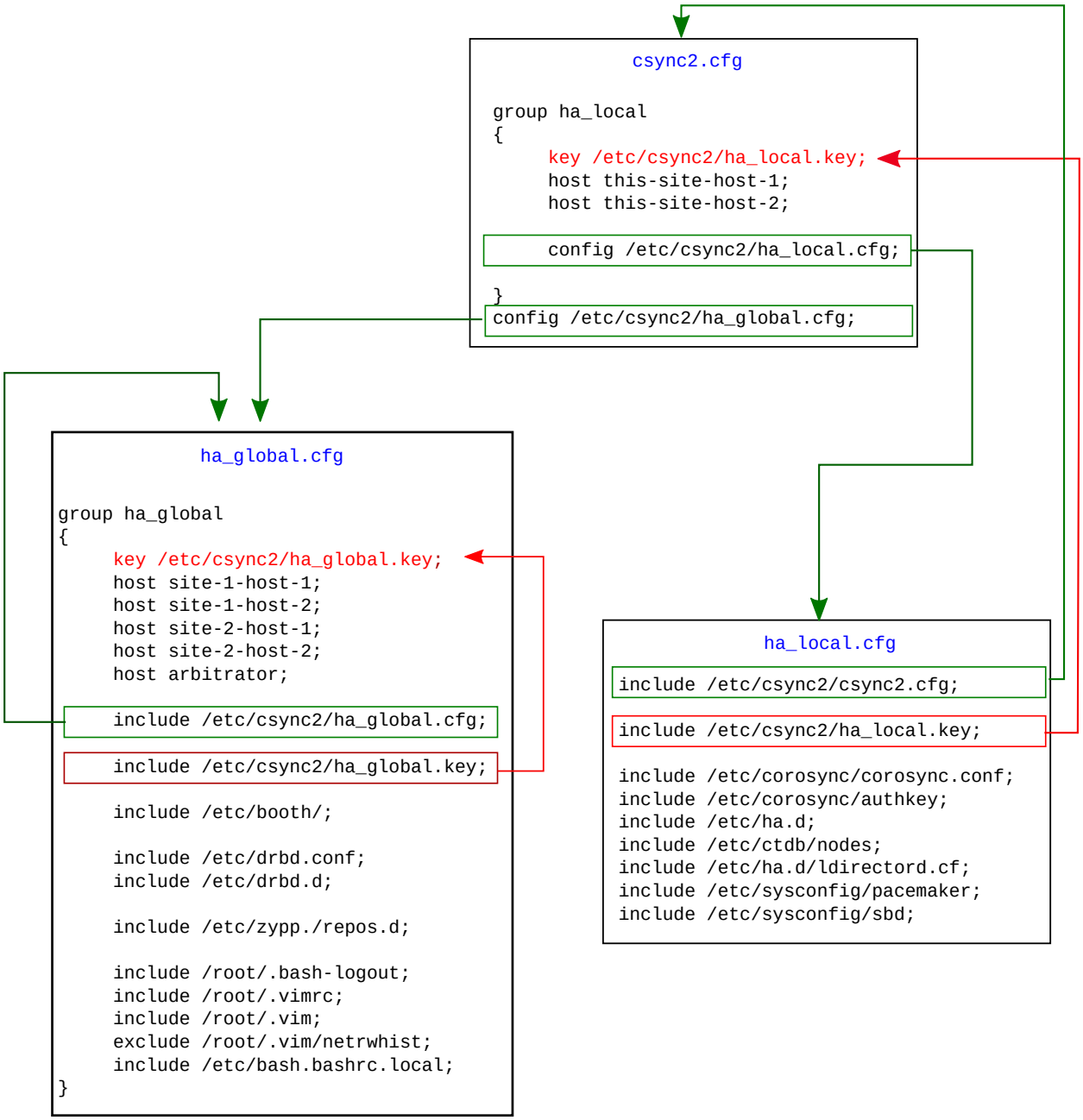

FIGURE 5.1: EXAMPLE CSYNC2 SETUP FOR GEO CLUSTERS

Authentication key files and their references are displayed in red. The names of Csync2 configuration files are displayed in blue, and their references are displayed in green. For details, refer to *Example Csync2 Setup: Configuration Files*.

EXAMPLE CSYNC2 SETUP: CONFIGURATION FILES

`/etc/csync2/csync2.cfg`

The main Csync2 configuration file. It is kept short and simple on purpose and only contains the following:

- The definition of the synchronization group `ha_local`. The group consists of two nodes (`this-site-host-1` and `this-site-host-2`) and uses `/etc/csync2/ha_local.key` for authentication. A list of files to be synchronized for this group only is defined in another Csync2 configuration file, `/etc/csync2/ha_local.cfg`. It is included with the `config` statement.
- A reference to another Csync2 configuration file, `/etc/csync2.cfg/ha_global.cfg`, included with the `config` statement.

`/etc/csync2/ha_local.cfg`

This file concerns only the local cluster. It specifies a list of files to be synchronized only within the `ha_local` synchronization group, as these files are specific per cluster. The most important ones are the following:

- `/etc/csync2/csync2.cfg`, as this file contains the list of the local cluster nodes.
- `/etc/csync2/ha_local.key`, the authentication key to be used for Csync2 synchronization within the local cluster.
- `/etc/corosync/corosync.conf`, as this file defines the communication channels between the local cluster nodes.
- `/etc/corosync/authkey`, the Corosync authentication key.

The rest of the file list depends on your specific cluster setup. The files listed in *Figure 5.1, "Example Csync2 Setup for Geo Clusters"* are only examples. If you also want to synchronize files for any site-specific applications, include them in `ha_local.cfg`, too. Even though `ha_local.cfg` is targeted at the nodes belonging to one site of your Geo cluster, the content may be identical on all sites. If you need different sets of hosts or different keys, adding extra groups may be necessary.

`/etc/csync2.cfg/ha_global.cfg`

This file defines the Csync2 synchronization group `ha_global`. The group spans *all* cluster nodes across multiple sites, including the arbitrator. As it is recommended to use a separate key for each Csync2 synchronization group, this group uses `/etc/csync2/ha_glob-`

`al.key` for authentication. The `include` statements define the list of files to be synchronized within the `ha_global` synchronization group. The most important ones are the following:

- `/etc/csync2/ha_global.cfg` and `/etc/csync2/ha_global.key` (the configuration file for the `ha_global` synchronization group and the authentication key used for synchronization within the group).
- `/etc/booth/`, the default directory holding the booth configuration. In case you are using a booth setup for multiple tenants, it contains more than one booth configuration file. If you use authentication for booth, it is useful to place the key file in this directory, too.
- `/etc/drbd.conf` and `/etc/drbd.d` (if you are using DRBD within your cluster setup). The DRBD configuration can be globally synchronized, as it derives the configuration from the host names contained in the resource configuration file.
- `/etc/zypp/repos.de`. The package repositories are likely to be the same on all cluster nodes.

The other files shown (`/etc/root/*`) are examples that may be included for reasons of convenience (to make a cluster administrator's life easier).

Note

The files `csync2.cfg` and `ha_local.key` are site-specific, which means you need to create different ones for each cluster site. The files are identical on the nodes belonging to the same cluster but different on another cluster. Each `csync2.cfg` file needs to contain a lists of hosts (cluster nodes) belonging to the site, plus a site-specific authentication key.

The arbitrator needs a `csync2.cfg` file, too. It only needs to reference `ha_global.cfg` though.

5.2 Synchronizing Changes with Csync2

To successfully synchronize the files with Csync2, the following prerequisites must be met:

- The same Csync2 configuration is available on all machines that belong to the same synchronization group.
- The Csync2 authentication key for each synchronization group must be available on all members of that group.
- Csync2 must be running on *all* nodes and the arbitrator.

Before the first Csync2 run, you therefore need to make the following preparations:

1. Log in to one machine per synchronization group and generate an authentication key for the respective group:

   ```
   root # csync2 -k NAME_OF_KEYFILE
   ```

   However, do *not* regenerate the key file on any other member of the same group.
   With regard to *Figure 5.1, "Example Csync2 Setup for Geo Clusters"*, this would result in the following key files: `/etc/csync2/ha_global.key` and one local key (`/etc/csync2/ha_local.key`) per site.

2. Copy each key file to *all* members of the respective synchronization group. With regard to *Figure 5.1, "Example Csync2 Setup for Geo Clusters"*:

   a. Copy `/etc/csync2/ha_global.key` to *all* parties (the arbitrator and all cluster nodes on all sites of your Geo cluster). The key file needs to be available on all hosts listed within the `ha_global` group that is defined in `ha_global.cfg`.

   b. Copy the local key file for each site (`/etc/csync2/ha_local.key`) to all cluster nodes belonging to the respective site of your Geo cluster.

3. Copy the site-specific `/etc/csync2/csync2.cfg` configuration file to all cluster nodes belonging to the respective site of your Geo cluster and to the arbitrator.

4. Execute the following command on all nodes and the arbitrator to make the csync2 service start automatically at boot time:

   ```
   root # systemctl enable csync2.socket
   ```

5. Execute the following command on all nodes and the arbitrator to start the service now:

```
root # systemctl start csync2.socket
```

PROCEDURE 5.1: SYNCHRONIZING FILES WITH CSYNC2

1. To initially synchronize all files once, execute the following command on the machine that you want to copy the configuration *from*:

```
root # csync2 -xv
```

This will synchronize all the files once by pushing them to the other members of the synchronization groups. If all files are synchronized successfully, Csync2 will finish with no errors.

If one or several files that are to be synchronized have been modified on other machines (not only on the current one), Csync2 will report a conflict. You will get an output similar to the one below:

```
While syncing file /etc/corosync/corosync.conf:
ERROR from peer site-2-host-1: File is also marked dirty here!
Finished with 1 errors.
```

2. If you are sure that the file version on the current machine is the "best" one, you can resolve the conflict by forcing this file and re-synchronizing:

```
root # csync2 -f /etc/corosync/corosync.conf
root # csync2 -x
```

For more information on the Csync2 options, run `csync2 -help`.

Note: Pushing Synchronization After Any Changes

Csync2 only pushes changes. It does *not* continuously synchronize files between the machines.

Each time you update files that need to be synchronized, you need to push the changes to the other machines of the same synchronization group: Run `csync2 -xv` on the machine where you did the changes. If you run the command on any of the other machines with unchanged files, nothing will happen.

6 Configuring Cluster Resources and Constraints

Apart from the resources and constraints that you need to define for your specific cluster setup, Geo clusters require additional resources and constraints as described below. You can either configure them with the crm shell (crmsh) as demonstrated in the examples below, or with Hawk2.

This chapter focuses on tasks specific to Geo clusters. For an introduction to your preferred cluster management tool and general instructions on how to configure resources and constraints with it, refer to one of the following chapters:

- *Book "Administration Guide", Chapter 7 "Configuring and Managing Cluster Resources with Hawk2"*
- *Book "Administration Guide", Chapter 8 "Configuring and Managing Cluster Resources (Command Line)"*

If you have set up your Geo cluster with the bootstrap scripts, the cluster resources needed for booth have been configured already (including a resource group for boothd). In this case, you can skip *Section 6.2* and only need to execute the remaining steps below to complete the cluster resource configuration.

If you are setting up your Geo cluster manually, you need to execute all of the following steps:

- *Section 6.1, "Configuring Ticket Dependencies of Resources"*
- *Section 6.2, "Configuring a Resource Group for* `boothd`*"*
- *Section 6.3, "Adding an Ordering Constraint"*
- *Section 6.4, "Transferring the Resource Configuration to Other Cluster Sites"*

Important: No CIB Synchronization Across Sites

The CIB is *not* automatically synchronized across cluster sites of a Geo cluster. All resources that must be highly available across the Geo cluster need to be configured for each site accordingly or need to be transferred to the other site or sites.

To simplify transfer, any resources with site-specific parameters can be configured in such a way that the parameters' values depend on the name of the cluster site where the resource is running (see also *Chapter 3, Requirements*, *Other Requirements and Recommendations*).

After you have configured the resources on one site, you can tag the resources that are needed on all cluster sites, export them from the current CIB, and import them into the CIB of another cluster site. For details, see *Section 6.4, "Transferring the Resource Configuration to Other Cluster Sites"*.

6.1 Configuring Ticket Dependencies of Resources

For Geo clusters, you can specify which resources depend on a certain ticket. Together with this special type of 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.

PROCEDURE 6.1: CONFIGURING TICKET DEPENDENCIES OF RESOURCES WITH CRMSH

1. On one of the nodes of cluster amsterdam, start a shell and log in as `root` or equivalent.
2. Enter **`crm configure`** to switch to the interactive crm shell.
3. Configure constraints that define which resources depend on a certain ticket. For example, to make a primitive resource `rsc1` depend on `ticketA`:

   ```
   crm(live)configure# rsc_ticket rsc1-req-ticketA ticketA: \
     rsc1 loss-policy="fence"
   ```

   In case `ticketA` is revoked, the node running the resource should be fenced.
4. If you want other resources to depend on further tickets, create as many constraints as necessary with **`rsc_ticket`**.
5. Review your changes with **`show`**.
6. If everything is correct, submit your changes with **`commit`** and leave the crm live configuration with **`exit`**.

The configuration is saved to the CIB.

6.2 Configuring a Resource Group for boothd

If you have set up your Geo cluster with the `ha-cluster-init` bootstrap scripts, you can skip the following procedure as the resources and the resource group for boothd have already been configured in this case.

Each site needs to run one instance of `boothd` that communicates with the other booth daemons. The daemon can be started on any node, therefore it should be configured as primitive resource. To make the `boothd` resource stay on the same node, if possible, add resource stickiness to the configuration. As each daemon needs a persistent IP address, configure another primitive with a virtual IP address. Group both primitives:

1. On one of the nodes of cluster `amsterdam`, start a shell and log in as `root` or equivalent.
2. Enter **`crm configure`** to switch to the interactive crm shell.
3. Enter the following to create both primitive resources and to add them to one group, `g-booth`:

```
crm(live)configure# primitive ip-booth ocf:heartbeat:IPaddr2 \
  params iflabel="ha" nic="eth1" cidr_netmask="24"
  params rule #cluster-name eq amsterdam ip="192.168.201.100" \
  params rule #cluster-name eq berlin ip="192.168.202.100"
crm(live)configure# primitive booth-site ocf:pacemaker:booth-site \
  meta resource-stickiness="INFINITY" \
  params config="nfs" op monitor interval="10s"
crm(live)configure# group g-booth ip-booth booth-site
```

   With this configuration, each booth daemon will be available at its individual IP address, independent of the node the daemon is running on.
4. Review your changes with **`show`**.
5. If everything is correct, submit your changes with **`commit`** and leave the crm live configuration with **`exit`**.
   The configuration is saved to the CIB.

6.3 Adding an Ordering Constraint

If a ticket has been granted to a site but all nodes of that site should fail to host the `boothd` resource group for any reason, a "split-brain" situation among the geographically dispersed sites may occur. In that case, no `boothd` instance would be available to safely manage failover of the ticket to another site. To avoid a potential concurrency violation of the ticket (the ticket is granted to multiple sites simultaneously), add an ordering constraint:

1. On one of the nodes of cluster amsterdam, start a shell and log in as `root` or equivalent.
2. Enter **`crm configure`** to switch to the interactive crm shell.
3. Create an ordering constraint, for example:

   ```
   crm(live)configure# order o-booth-before-rsc1 inf: g-booth rsc1
   ```

   It defines that `rsc1` (which depends on `ticketA`) can only be started after the `g-booth` resource group.
4. For any other resources that depend on a certain ticket, define further ordering constraints.
5. Review your changes with **`show`**.
6. If everything is correct, submit your changes with **`commit`** and leave the crm live configuration with **`exit`**.
   The configuration is saved to the CIB.

6.4 Transferring the Resource Configuration to Other Cluster Sites

After having completed or changed your resource configuration for one cluster site, transfer it to the other sites of your Geo cluster.

To simplify the transfer, you can tag any resources that are needed on all cluster sites, export them from the current CIB, and import them into the CIB of another cluster site. Tagging does not create any colocation or ordering relationship between the resources.

Procedure 6.2, "Tagging and Exporting a Resource Configuration" and *Procedure 6.3, "Importing a Tagged Resource Configuration"* give an example of how to do so. They are based on the following prerequisites:

PREREQUISITES

- You have a Geo cluster with two sites: cluster `amsterdam` and cluster `berlin`.
- The cluster names for each site are defined in the respective `/etc/corosync/corosync.conf` files:

  ```
  totem {
       [...]
       cluster_name: amsterdam
       }
  ```

  This can either be done manually (by editing `/etc/corosync/corosync.conf`) or with the YaST cluster module (by switching to the *Communication Channels* category and defining a *Cluster Name*). Afterward, stop and start the `pacemaker` service for the changes to take effect:

  ```
  root # systemctl stop pacemaker
  root # systemctl start pacemaker
  ```

- The necessary resources for booth and for all services that should be highly available across your Geo cluster have been configured in the CIB on site `amsterdam`. They will be imported to the CIB on site `berlin`.

PROCEDURE 6.2: TAGGING AND EXPORTING A RESOURCE CONFIGURATION

1. Log in to one of the nodes of cluster `amsterdam`.
2. Start the cluster with:

   ```
   root # systemctl start pacemaker
   ```

3. Enter **`crm configure`** to switch to the interactive crm shell.
4. Review the current CIB configuration:

   ```
   crm(live)configure# show
   ```

5. Mark the resources and constraints that are needed across the Geo cluster with the tag `geo_resources`:

```
crm(live)configure# tag geo_resources: \
  LIST_OF_RESOURCES_and_CONSTRAINTS_FOR_REQUIRED_SERVICES (1)\
  rsc1-req-ticketA ip-booth booth-site g-booth o-booth-before-rsc1 (2)
```

   (1) Any resources and constraints of your specific setup that you need on all sites of the Geo cluster (for example, resources for DRBD as described in http://www.suse.com/documentation/suse-best-practices/sbp-drbd/data/sbp-drbd.html).

   (2) Resources and constraints for boothd (primitives, booth resource group, ticket dependency, additional ordering constraint), see *Section 6.1* to *Section 6.3*.

6. Review your changes with **show**.

7. If the configuration is according to your wishes, submit your changes with **submit** and leave the crm live shell with **exit**.

8. Export the tagged resources and constraints to a file named `exported.cib`:

```
root # crm configure show tag:geo_resources geo_resources > exported.cib
```

   The command **crm configure show tag:** *TAGNAME* shows all resources that belong to the tag *TAGNAME*.

PROCEDURE 6.3: IMPORTING A TAGGED RESOURCE CONFIGURATION

To import the saved configuration file into the CIB of the second cluster site, proceed as follows:

1. Log in to one of the nodes of cluster `berlin`.

2. Start the cluster with:

```
root # systemctl start pacemaker
```

3. Copy the file `exported.cib` from cluster `amsterdam` to this node.

4. Import the tagged resources and constraints from the file `exported.cib` into the CIB of cluster `berlin`:

```
root # crm configure load update PATH_TO_FILE/exported.cib
```

When using the `update` parameter for the **`crm configure load`** command, crmsh tries to integrate the contents of the file into the current CIB configuration (instead of replacing the current CIB with the file contents).

5. View the updated CIB configuration with the following command:

```
root # crm configure show
```

The imported resources and constraints will appear in the CIB.

7 Setting Up IP Relocation via DNS Update

In case one site of your Geo cluster is down and a ticket failover appears, you usually need to adjust the network routing accordingly (or you need to have configured a network failover for each ticket). Depending on the kind of service that is bound to a ticket, there is an alternative solution to reconfiguring the routing: You can use dynamic DNS update and instead change the IP address for a service.

The following prerequisites must be fulfilled for this scenario:

- The service that needs to fail over is bound to a host name.
- Your DNS server must be configured for dynamic DNS updates. For information on how to do so with BIND/named, see the `named` documentation, or refer to http://www.semi-complete.com/articles/dynamic-dns-with-dhcp/. More information on how to set up DNS, including dynamic update of zone data, can be found in the SUSE Linux Enterprise *Administration Guide*, chapter *The Domain Name System*. It is available from http://www.suse.com/documentation/sles.
- The following example assumes that the DNS updates are protected by a shared key (TSIG key) for the zone to be updated. The key can be created using **`dnssec-keygen`**:

  ```
  root # dnssec-keygen -a hmac-md5 -b 128 -n USER geo-update
  ```

  For more information, see the **`dnssec-keygen`** man page or the SUSE Linux Enterprise *Administration Guide*, chapter *The Domain Name System*, section *Secure Transactions*. It is available from http://www.suse.com/documentation/sles.

Example 7.1, "Resource Configuration for Dynamic DNS Update" illustrates how to use the `ocf:heartbeat:dnsupdate` resource agent to manage the **`nsupdate`** command. The resource agent supports both IPv4 and IPv6.

EXAMPLE 7.1: RESOURCE CONFIGURATION FOR DYNAMIC DNS UPDATE

```
crm(live)configure# primitive dns-update-ip ocf:heartbeat:dnsupdate params \
  hostname="www.domain.com" ❶ ip="192.168.3.4" ❷ \
  keyfile="/etc/whereever/Kgeo-update*.key" ❸ \
  server="192.168.1.1" ❹ serverport="53" ❺
```

❶ Host name bound to the service that needs to fail over together with the ticket. The IP address of this host name needs to be updated via dynamic DNS.

2. IP address of the server hosting the service to be migrated. The IP address specified here can be under cluster control, too. This does not handle local failover, but it ensures that outside parties will be directed to the right site after a ticket failover.
3. Path to the public key file generated with **`dnssec-keygen`**.
4. IP address of the DNS server to send the updates to. If no server is provided, this defaults to the master server for the correct zone.
5. Port to use for communication with the DNS server. This option will only take effect if a DNS server is specified.

With the resource configuration above, the resource agent takes care of removing the failed Geo cluster site from the DNS record and changing the IP for a service via dynamic DNS update.

8 Managing Geo Clusters

Before booth can manage a certain ticket within the Geo cluster, you initially need to grant it to a site manually—either with the booth command line client or with Hawk2.

8.1 Managing Tickets From Command Line

Use the `booth` command line tool to grant, list, or revoke tickets as described in *Section 8.1.1, "Overview of* **booth** *Commands"*.

Warning: `crm_ticket` and `crm site ticket`

If the booth service is not running for any reasons, you can also manage tickets manually with `crm_ticket` or `crm site ticket`. Both commands are only available on cluster nodes. Use them with great care as they *cannot* verify if the same ticket is already granted elsewhere. For more information, read the man pages.

As long as booth is up and running, only use the `booth` for manual intervention.

8.1.1 Overview of `booth` Commands

The `booth` commands can be run on any machine in the cluster, not only the ones having the `boothd` running. The `booth` commands try to find the "local" cluster by looking at the booth configuration file and the locally defined IP addresses. If you do not specify a site which `booth` should connect to (using the `-s` option), it will always connect to the local site.

Listing All Tickets

```
root # booth list
ticket: ticketA, leader: none
ticket: ticketB, leader: 10.2.12.101, expires: 2014-08-13 10:28:57
```

If you do not specify a certain site with `-s`, the information about the tickets will be requested from the local booth instance.

Granting a Ticket to a Site

```
root # booth grant -s 192.168.201.100 ticketA
```

```
booth[27891]: 2014/08/13_10:21:23 info: grant request sent, waiting for the
 result ...
booth[27891]: 2014/08/13_10:21:23 info: grant succeeded!
```

In this case, `ticketA` will be granted to the site `192.168.201.100`. Without the `-s` option, booth would automatically connect to the current site (the site you are running the booth client on) and would request the **`grant`** operation.

Before granting a ticket, the command executes a sanity check. If the same ticket is already granted to another site, you are warned about that and are prompted to revoke the ticket from the current site first.

Revoking a Ticket From a Site

```
root # booth revoke ticketA
booth[27900]: 2014/08/13_10:21:23 info: revoke succeeded!
```

Booth checks to which site the ticket is currently granted and requests the **`revoke`** operation for `ticketA`. The revoke operation will be executed immediately.

The **`grant`** and (under certain circumstances), **`revoke`** operations may take a while to return a definite operation's outcome. The client waits for the result up to the ticket's `timeout` value before it gives up waiting. If the `-w` option was used, the client will wait indefinitely instead. Find the exact status in the log files or with the **`crm_ticket -L`** command.

Forcing a Grant Operation

```
root # booth grant -F ticketA
```

The result of this command depends on whether you use automatic or manual tickets.

- **Automatic Tickets.** As long as booth can make sure a ticket is granted to one site, you cannot grant the same ticket to another site, not even by using the `-F` option. However, in case of a split brain situation, booth might not be able to check if an automatic ticket is granted somewhere else. In that case, the Geo cluster administrator can override the automatic process and manually grant the ticket to the site that is still up and running. In this situation, the `-F` options tells booth *not* to wait for a response from other, unreachable sites (so ignoring the parameters `expire` and `acquire-after`, if defined for this ticket). Instead, booth will immediately grant the ticket to the specified site.
- **Manual Tickets.** When using *manual* tickets, **`booth grant -F`** makes booth grant the ticket immediately to the specified site.

Warning: Potential Loss of Data

Before using `booth grant -F`, make sure that no other site (which is online) owns the same ticket. If the same ticket is granted to multiple sites, resources depending on the ticket might start on several sites in parallel. This results in concurrency violation and potential data corruption.

As Geo cluster administrator, you need to resolve a conflict between tickets once the other site is reachable again.

In the following sections, find some examples for managing tickets in different scenarios.

8.1.2 Manually Moving an Automatic Ticket

Assuming that you want to manually move `ticketA` from site `amsterdam` (with the virtual IP `192.168.201.100`) to site `berlin` (with the virtual IP `192.168.202.100`), proceed as follows:

1. Log in to `amsterdam`.
2. Set `ticketA` to standby with the following command:

```
root # crm_ticket -t ticketA -s
```

3. Wait for any resources that depend on `ticketA` to be stopped or demoted cleanly.
4. Grant `ticketA` to the site `berlin` with:

```
root # booth grant -s 192.168.202.100 ticketA
```

5. Activate `ticketA` on site `amsterdam` with:

```
root # booth grant -a 192.168.202.100 ticketA
```

Resources depending on ticketA will automatically fail over to `amsterdam` in case `berlin` fails.

8.1.3 Moving a Manual Ticket

Assuming that you want to manually move `ticketnfs` from site `amsterdam` (with the virtual IP `192.168.201.100`) to site `berlin` (with the virtual IP `192.168.202.100`), proceed as follows:

1. Log in to `amsterdam`.
2. Set `ticket-nfs` to standby with the following command:

   ```
   root # crm_ticket -t ticket-nfs -s
   ```

3. Wait for any resources that depend on `ticket-nfs` to be stopped or demoted cleanly.
4. Grant `ticket-nfs` to the site `berlin` with:

   ```
   root # booth grant -s 192.168.202.100 -F ticket-nfs
   ```

5. Activate `ticket-nfs` on site `amsterdam` with:

   ```
   root # booth grant -a 192.168.202.100 ticketA
   ```

   Resources depending on `ticket-nfs` will automatically fail over to `amsterdam` in case `berlin` fails.

8.1.4 Failing Over a Manual Ticket

Let us assume that the (manually managed) ticket `ticket-nfs` had been granted to site `amsterdam` (with the virtual IP `192.168.201.100`. This site cannot be reached at the moment. Site `berlin` (with the virtual IP `192.168.202.100`) is still available.

If you manage to contact a local administrator on `amsterdam`, you can verify if the site is really down. In that case, the ticket on `amsterdam` cannot be revoked and the resources depending on this ticket cannot be started on `berlin` unless you manually grant the ticket `ticket-nfs` to `berlin`. To do so, proceed as follows:

1. Log in to `berlin`.
2. Grant `ticket-nfs` to site `berlin` using the `-F` option:

   ```
   root # booth grant -F ticket-nfs
   ```

   You will see a warning that the same ticket might be granted to another site, but the command will be executed.

3. Check the result with:

```
root # booth list
```

   It should show `berlin` as ticket owner for `ticket-nfs` now. All resources that depend on this ticket will be started on `berlin`.

4. Before trying to bring back `amsterdam` into the Geo cluster again, make sure to revoke `ticket-nfs` from `amsterdam`:

```
root # booth revoke -s 192.168.201.100 ticket-nfs
```

8.2 Managing Tickets with Hawk2

Tickets can be viewed in both the *Dashboard* and the *Status* view. Hawk2 displays the following ticket statuses:

- *Granted*: Tickets that are granted to the current site.
- *Elsewhere*: Tickets that are granted to another site.
- *Revoked*: Tickets that have been revoked. Additionally, Hawk2 also displays tickets as revoked if they are referenced in a ticket dependency, but have not been granted to any site yet.

Note: Granting Tickets to Current Site and Revoking Tickets

Though you can view tickets for all sites with Hawk2, any grant or revoke operations triggered by Hawk2 only apply to the current site (that you are currently connected to with Hawk2). To grant a ticket to another site of your Geo cluster, start Hawk2 on one of the cluster nodes belonging to the respective site.

You can only grant tickets that are not already given to any site.

PROCEDURE 8.1: VIEWING, GRANTING AND REVOKING TICKETS WITH HAWK2

1. Start a Web browser and log in to Hawk2.
2. In the left navigation bar, select *Monitoring* › *Status*.

Along with information about cluster nodes and resources, Hawk2 also displays a *Tickets* category. It lists the ticket status, the ticket name and when the ticket was last granted. From the *Granted* column you can manage the tickets.

3. To show further information about the ticket, along with information about the cluster sites and arbitrators, click the *Details* icon next to the ticket.

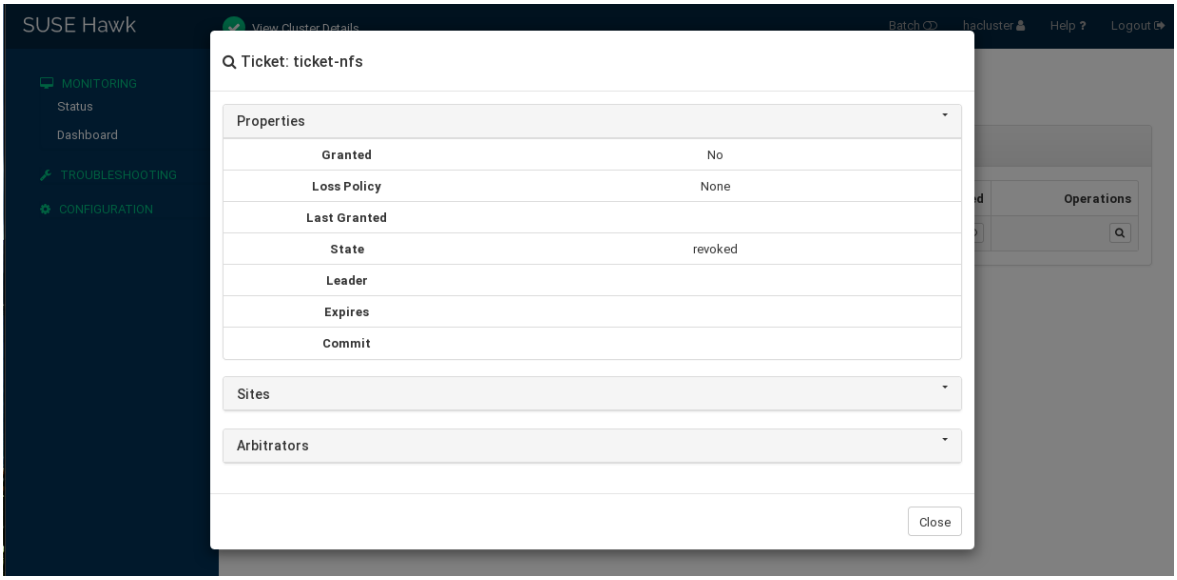

FIGURE 8.1: HAWK2—TICKET DETAILS

4. To revoke a granted ticket from the current site or to grant a ticket to the current site, click the switch in the *Granted* column next to the ticket. On clicking, it shows the available action. Confirm your choice when Hawk2 prompts for a confirmation.
If the ticket cannot be granted or revoked for any reason, Hawk2 shows an error message. If the ticket has been successfully granted or revoked, Hawk2 will update the ticket *Status*.

PROCEDURE 8.2: SIMULATING GRANTING AND REVOKING TICKETS

Hawk2's *Batch Mode* allows you to explore failure scenarios before they happen. To explore whether your resources that depend on a certain ticket behave as expected, you can also test the impact of granting or revoking tickets.

1. Start a Web browser and log in to Hawk2.
2. From the top-level row, select *Batch Mode*.
3. In the batch mode bar, click *Show* to open the *Batch Mode* window.

4. To simulate a status change of a ticket:

    a. Click *Inject* › *Ticket Event*.

    b. Select the *Ticket* you want to manipulate and select the *Action* you want to simulate.

    c. Confirm your changes. Your event is added to the queue of events listed in the *Batch Mode* dialog. Any event listed here is simulated immediately and is reflected on the *Status* screen.

    d. Close the *Batch Mode* dialog and review the simulated changes.

5. To leave the batch mode, either *Apply* or *Discard* the simulated changes.

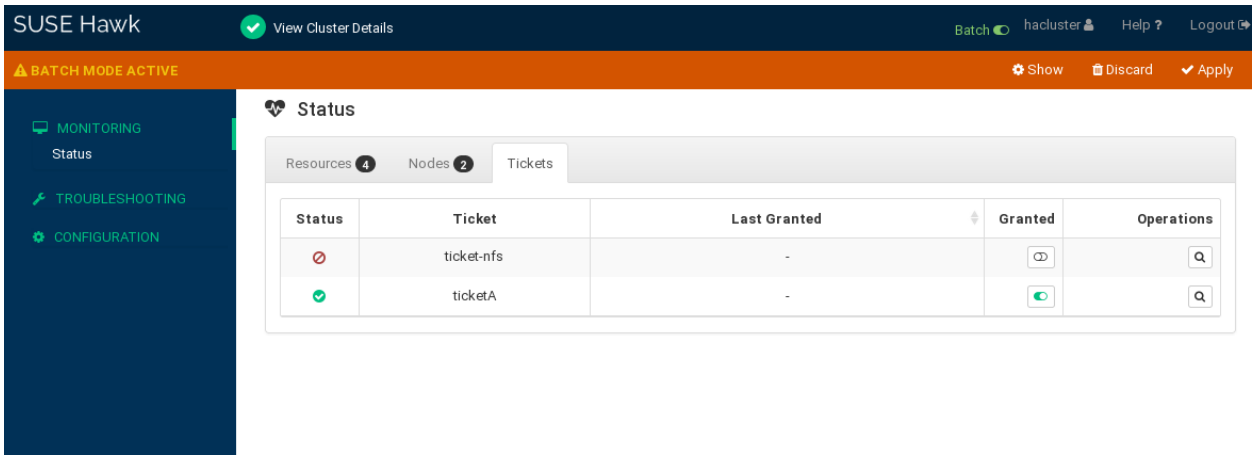

FIGURE 8.2: HAWK2 SIMULATOR—TICKETS

For more information about Hawk2's *Batch Mode* (and which other scenarios can be explored with it), refer to *Book "Administration Guide", Chapter 7 "Configuring and Managing Cluster Resources with Hawk2", Section 7.9 "Using the Batch Mode"*.

9 Troubleshooting

Booth uses the same logging mechanism as the CRM. Thus, changing the log level will also take effect on booth logging. The booth log messages also contain information about any tickets.

Both the booth log messages and the booth configuration file are included in the **`crm report`**.

In case of unexpected booth behavior or any problems, check the logging data with **`sudo journalctl -n`** or create a detailed cluster report with **`crm report`**.

In case you can access the cluster nodes on all sites (plus the arbitrators) from one single host via SSH, it is possible to collect log files from all of them within the same `crm report`. When calling **`crm report`** with the `-n` option, it gets the log files from all hosts that you specify with `-n`. (Without `-n`, it would try to obtain the list of nodes from the respective cluster). For example, to create a single `crm report` that includes the log files from two two-node clusters (`192.168.201.111`|`192.168.201.112` and `192.168.202.111`|`192.168.202.112`) plus an arbitrator (`147.2.207.14`), use the following command:

```
root #  crm report -n "147.2.207.14 192.168.201.111 192.168.201.112 \
 192.168.202.111 192.168.202.112"  -f 10:00 -t 11:00 db-incident
```

If the issue is about booth only and you know on which cluster nodes (within a site) booth is running, then specify only those two nodes plus the arbitrator.

If there is no way to access all sites from one host, run **`crm report`** individually on the arbitrator, and on the cluster nodes of the individual sites, specifying the same period of time. To collect the log files on an arbitrator, you must use the `-S` option for single node operation:

```
amsterdam # crm report -f 10:00 -t 11:00 db-incident-amsterdam
berlin # crm report -f 10:00 -t 11:00 db-incident-berlin
arbitrator # crm report -S -f 10:00 -t 11:00 db-incident-arbitrator
```

However, it is preferable to produce one single `crm report` for all machines that you need log files from.

10 Upgrading to the Latest Product Version

For instructions on how to upgrade the cluster nodes, refer to *Book "Administration Guide", Chapter 5 "Upgrading Your Cluster and Updating Software Packages"*. The chapter also describes which preparations to take before starting the upgrade process. It provides an overview of the supported upgrade paths and where to find the details for each step.

If you use an arbitrator outside of the cluster sites, upgrade the arbitrator as described in *Procedure 10.1*.

PROCEDURE 10.1: UPGRADING AN ARBITRATOR

1. Perform an upgrade to the desired target version of SUSE Linux Enterprise Server and SUSE Linux Enterprise High Availability Extension as described in *Book "Administration Guide", Chapter 5 "Upgrading Your Cluster and Updating Software Packages"*.
2. Check if you have enabled the modules and extensions mentioned in *Article "Geo Clustering Quick Start", Section 3 "Requirements"*.
3. Check if the `booth` package is installed:

   ```
   root # zypper pa | grep booth
   ```
4. If not, install it with:

   ```
   root # zypper install booth
   ```

11 For More Information

- More documentation for this product is available at http://www.suse.com/documentation/sle-ha-geo. For example, the *Geo Clustering Quick Start* guides you through the basic setup of a Geo cluster, using the Geo bootstrap scripts provided by the `ha-cluster-bootstrap` package.
- A document with detailed information how to on data replication via DRBD across Geo clusters has been published in the `SUSE Best Practices` series: http://www.suse.com/documentation/suse-best-practices/sbp-drbd/data/sbp-drbd.html.

A GNU Licenses

This appendix contains the GNU Free Documentation License version 1.2.

GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/↗.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

```
  Copyright (c) YEAR YOUR NAME.
  Permission is granted to copy,
distribute and/or modify this document
  under the terms of the GNU Free
Documentation License, Version 1.2
  or any later version published by the
Free Software Foundation;
  with no Invariant Sections, no Front-
Cover Texts, and no Back-Cover Texts.
  A copy of the license is included in
the section entitled “GNU
  Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
  with the Invariant Sections being LIST
THEIR TITLES, with the
  Front-Cover Texts being LIST, and with
the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.