Project Hedeby

Specification Reloaded

Hedeby Engineering Team

Project Hedeby: Specification Reloaded

Hedeby Engineering Team

Revision 0.4.4 - Draft

Table of Contents

List of Tables

List of Examples

Chapter 1. Overview

Common concept

Aim of the project Hedeby is a Service Domain Management system which makes it possible to manage scalable services.

This project is developed by the Sun Grid Engine Management team. As the Sun Grid Engine project, the Hedeby project has also been open sourced under SISSL license [http://hedeby.sunsource.net/license.html].

The Service Domain Manager is designed to handle very different kind of services. The main purpose is solving resource lacking of such services. Hedeby is interresting for all administrators managing huge services with an administration interface. The Service Domain Manager will be able to detect scalabilty problem and resolve them.

For the first release we (the Hedeby team) will concentrate on using Hedeby to manage the Sun Grid Engine service. In future it should be able to support other services.

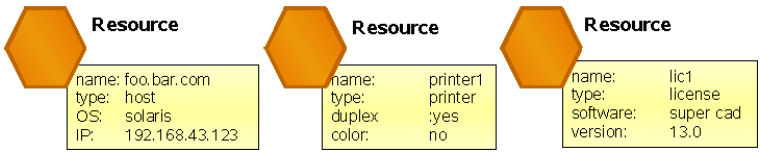
Resource

What is a resource? In the project Hedeby a resource can be nearly everything. It can be some hardware (e.g. a host or a printer) or it can be a software (a specific application or licenses). In general a resource is something a service uses to provide the service. If you give a service more resources, it can do more work in the same time.

Each resource should have a system wide unique id, the resource id to be fully usable by Hedeby. This id must identify the resource. For a host resource this can be the full qualified hostname.

A resource in Hedeby is seen as single entity. It is not wanted to share resources between services e.g. if Hedeby should manage a license which allows the usage of a software for ten user concurrently, the administrator has to add ten resources to the Hedeby system. Each license resource must have a unique id, as sharing a license may lead to violiting of license agreement and to service malfunction. Also, sharing a resource between services can lead to downgrading a service performance so ideally each resource should be assigned to a single service exclusively. See the section called “Ambiguous Resources” for details about non-unique resource ids.

A Hedeby system stores for each registered resource a set of properties. These resource properties describes the resource. Examples are the number of CPUs, memory and architecture of a host, version number of a software, number of licenses. Each resource property has a name and a value.



Resources in a Hedeby system

Static Resources

The Hedeby system distinguishes between static and dynamic resource. A static resource can not be removed from a service. The Hedeby system will never touch a static resource. When removing a service all assigned static resources will disappear from the Hedeby system.

Dynamic resources can be removed (unassigned) from service and added to another (assigned).

Ambiguous Resources

A service can use couple of resources even before the service is managed by Hedeby e.g. web server cluster (service) is running on four servers (resources). When a service becomes managed, it reports its resources to Hedeby (auto-discovery of resources) - in this case, it may happen that service reports a resource with a resource id that is already used in Hedeby system. This may signal either that a resource is shared between two services or that there is just a name collision. Either case is not wanted and we call such resource AMBIGUOUS as it is not clear which service should use the resource exclusively. The case has to be solved manually by removing one instance of the ambiguous resources from the system.

Presence of ambiguous resource does not meant that the system is not functional at all. An ambiguous resource may be fully or partially functional (depending on a service), but to avoid possible problems, system puts several constrains on an ambiguous resource:

- Ambiguous resource can not be modified
- Ambiguous resource can not be moved (from service A to service B)
- Ambiguous resource will not be considered as candidate for filling a resource request

The only operations allowed on an ambiguous resource are:

- Ambiguous resource may be reset
- Ambiguous resource may be removed from system

Resource States

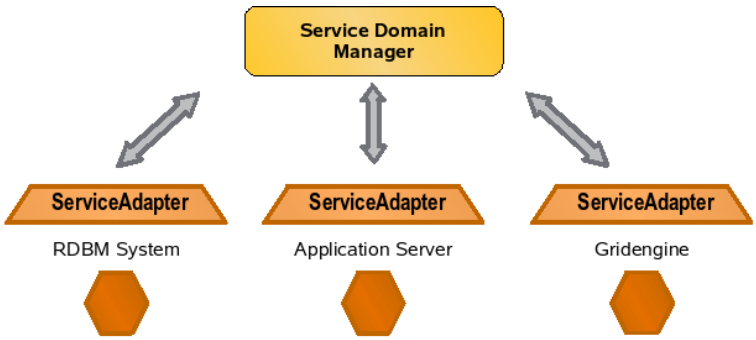
The Hedeby system knows the following resource states

UNASSIGNED	A resource is not assigned to a service. It is currently being processed by resource provider (e.g. for filling a resource request).
ASSIGNING	A resource is in proccess of being assigned to a service (e.g. service adapter performs installation of service components on a resource, but it is not finished yet).
ASSIGNED	A resource has been successfully assigned to a service. A service has full control of a resource.
UNASSIGNING	A resource is in process of being released from a service (e.g. service adapter performs uninstallation of service components on a resource, but it is not finished yet).
INPROCESS	A resource has been just added to RP (temporary, until it is assinged to another service) and there has been no info yet about a resource state (whether it has been released from service or no).
ERROR	An action on a resource produced an unrecoverable error. The resource is currently not usable and has to be reset.

Service

A service in the term of Hedeby is a piece of software. It can be a database, an application server or any other software. The only constrain is that the software has to provide a service management interface.

To make a service manageable Hedeby needs a driver for the service. Such a driver is called service adapter. The service adapter is packaged in a jar file. It has its own configuration and in the current version is runs inside a service container.



Services in a Hedeby system

Service States

From the view of a Hedeby system a service as the following states:

UNKNOWN There exists no connection to the service. Hedeby has no idea in what state the real service is. A service goes into the UNKNOWN state if the communication between the Hedeby system and the real service is interrupted.

STARTING Hedeby is currently starting the service

RUNNING Hedeby has a connection to the service. The service is observed.

SHUTDOWN The service is going down.

ERROR Hedeby can contact the service, but it is not working in the expected way. The administrator of the service has to solve the problem.

STOPPED The service has been stopped.

State changes of a service can be triggered from Hedeby system or from the service itself. The service adapter has to be smart enough to detect external service state changes (e.g. a service adapter for a Grid Engine has to catch the "qmaster goes down" event).

What really happens if a Hedeby system starts or stopps a service is a implementation detail of the service adapter (e.g does service adapter for Grid Engine shutdown qmaster?). The Hedeby system knows only the states reported by the service adapter which interprets the states of the real service.

Registered Services

The Hedeby system distinguishes between registered and unregistered services. Unregistered service are not included into the decision making process. No resources are assigned or unassigned. The service will also report no needs if it is unregistered.

Key Performance Indicators (KPI)

To make the performance measurable, each service defines a set of key performance indicators. The service adapter collections this numerical values by using the service management interface.

Example 1.1. Sample KPIs

- Number of transaction per second
- Disk usage
- Memory usage

Each PKI can have additional properties which indentifies the performance of a logical unit of the service.

Example 1.2. KPI's with properties

- Number of used rows of each database table (name of the database table is a propertiy of the KPI)
- Number of pending jobs per host (hostname is a property)
- Number of pending waiting for a specific license (name of the license is a property)

The service adapter is responsible to map the properties of the KPI's into resource properties. He has to provide some maping tables.

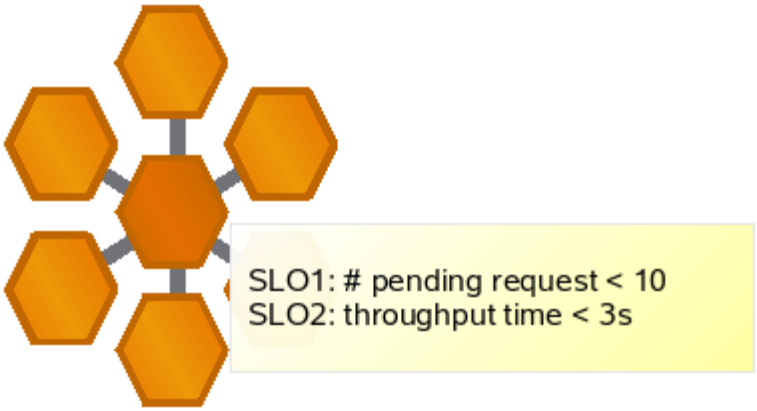
Service Level Objectives (SLO)

Hedeby allows the definition of rules which describes the current state of a service. A single rule is called service level objective. If a SLO can be fulfilled or not for a specific time. If a SLO is fulfilled we say the service is in compliance with this SLO.

With a set of SLOs the administrator of the Hedeby system defines implicit a service level agreement (SLA) for a service. If all SLOs are fullfilled the service works for a defined scenario. The SLA itself can not be defined in the Hedeby system. Only the set of SLOs.

Example 1.3. Typical SLA for a service

The following graphic shows a typical SLA definition for a service. The SLA if formulated with two SLOs. The number of pending request to the service should always be less then 10 and the throughput time of a request should be less then 3s.



SLOs of a service

Need

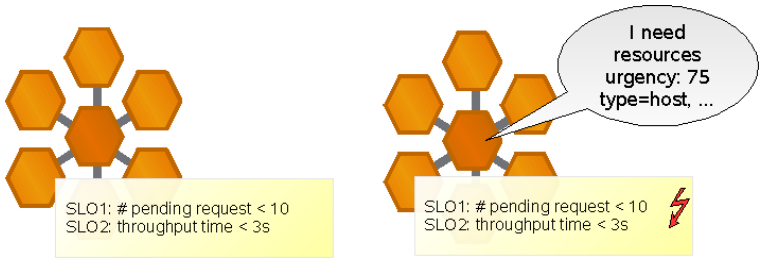
If some SLOs are not fulfilled the service has a need for additional resources. The service has to describe what kind of resources are needed by specifing resource properties. The Hedeby system (the section called “Resource Provider”) will try the solve this lack of resources by assigning new resources to the service.

A need contains the information about the needed resource (type of resource, resource properties) and a urgency. This urgency is a non-negative number (0 and above) where the higher number specifies the more urgent need.

The administrator of the Hedeby system has to define what need will be generated if SLOs are not fulfilled.

Example 1.4. Service reports a need

The following graphic shows the SLA defintion described in Example 1.3, “Typical SLA for a service ”. If one of the SLOs is not fullfilled the service will report the need the new resource of type host a needed. The urgency of the need is 75 (relatively high).



A service reports a need

The calculated urgency is only absolute for this service. Settings in the Policy Engine relativates the urgency in comparision to other services (see the section called “Policy Engine”).

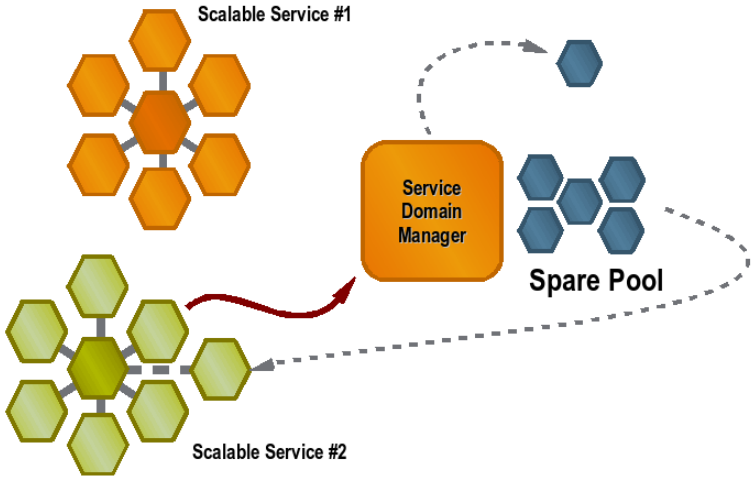
Spare Pool

Hedeby provides a special service and component in each Hedeby system. It is named the Spare Pool. This spare_pool service collects all resources which are not heavilly used by service to which are they currently assigned by sending constant request. There could be more than one spare_pool components installed.

The Spare Pool supports only one SLO. No matter how many resources are assigned to the Spare Pool the SLO is never fullfilled. The urgency of generated need of the Spare Pool is configurable by the adminstrator. It should be small enough so that no resource is assigned to the Spare Pool while other service needs them.

Example 1.5. Service gets resource from Spare Pool

In this example we have a Hedeby system with three services (including the Spare Pool). The Spare Pool contains currently six resources. Service #1 is in compliance with it's SLOs. Service #2 has a need for an additional resources. The urgency of the Spare Pool is lower then the urgency of service #2. The service domain manager is taking one resource out of the Spare Pool and is assigning it to service #2.



Role of the Spare Pool in a Hedeby system

Resource Usage

The resource usage gives the Hedeby system the information how important the resource for this service is. The usage is non-negative number (0 is also allowed). It's the resonsibility of the service to keep the usage of the resource uptodate (e.g. if KPI if the service has changed).

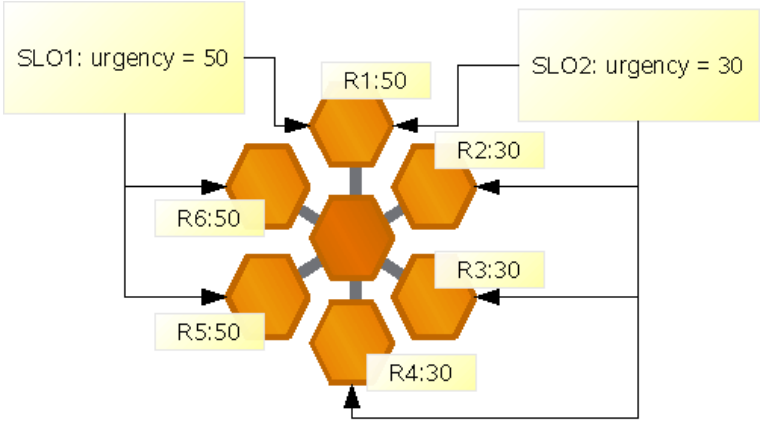
In general we can say that the usage of a resource is the maximum urgency of the SLOs which needs the resource to be fulfilled.

Example 1.6. Resource Usage

A service has six resource assigned (R1-R6). There exists two SLOs for this service. SLO1 has urgency 50 and SL2 has urgency 30.

Resource R2, R3 and R4 have a usage 30, because they are need to full fill SLO2. Resource R5 and R6 have usage 50 (urgency of SLO1).

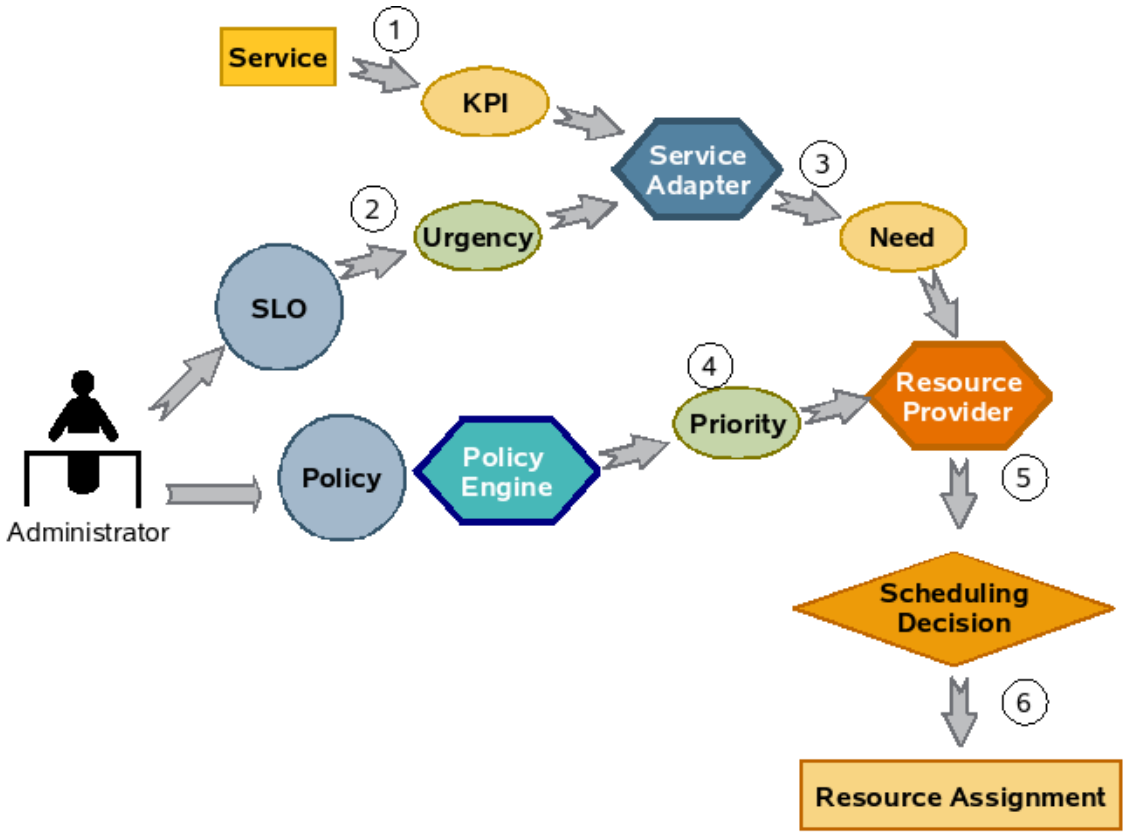
Resource R1 is need by SLO1 and SLO2. In such cases the resource will have the maximum urgency of all associated SLOs, this means R1 has usage 50 (= max(urgency of SLO1, urgency of SLO2)).



Usage of assigned resource

Resource Provider

The Resource Provider is the central component in a Hedeby system. It has the control over all services and resources. Each service adapter must inform the Resource Provider if the state of a service or a resource has changed. The Resource Provider makes the decisions whether a service gets a resource or not. The following image illustrates the decision making process:



Decision making process

1 The service reports to the service adapter it's key performance indicators (KPI).

2 The administrator defines SLOs which calculates based on the KPIs of the service the resource need for each SLO.

3 List of all needs for all SLOs are send to the resource provider.

4 The Resource Provider uses the Policy Engine to normalize the needs for the services (normalization of the needs).

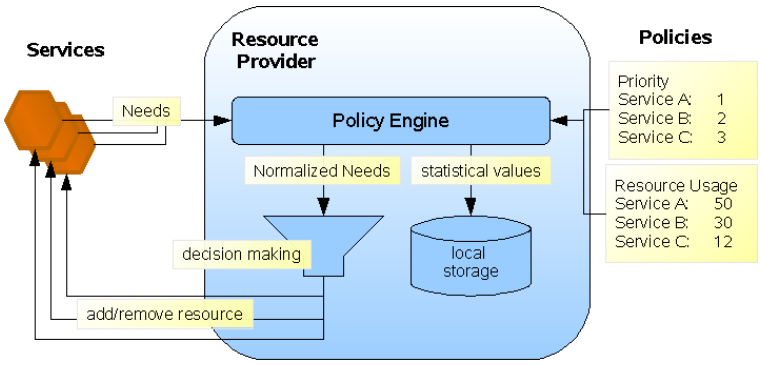
5 Based on the normalized needs the Resource Provider meets it resource assignment decisions.

6 The Resource Provider sends to the service the corresponding resource assignments/ unassignments.

At startup the Resource Provider discovers the Hedeby system. It asks all services what resources they posses and store that information in it's local storage.

Policy Engine

With the Policy Engine it is possible to define policies which influences the decisions of the Resource Provider. The Policy Engine calculates out of the need of a service a new urgency.



The Policy Engine rules the decision making process of the resource Provider

The Policy Engine has access to statistical values of the resource usage. The following information can be provided:

- Number of resources assigned to a service which match a given resource properties pattern (e.g. number of host with solaris operating system).
- Number of resources in the given state which match the given resource properties pattern (e.g. number of assigned host resources with more then 2GB memory).

Note

To make time base decisions the Policy Engine will need information about how long has a resource been assigned to service.

The Policy Engine provides a generic interface which make it possible to plug other implemenation into the Hedeby system.

Example 1.7. Example for a simple Policy Engine

A simple implementation of a Policy Engine can weight the importance of a service by given them different priorities. The policy engine multiplies the urgencies of the services reported in a need with the priority of the service and gets so the weighted needs.

Service	Priority	Number of Resources
Spare Pool	1	1
A	2	3
B	3	2

For the services the following SLOs are defined:

Service	SLO	Urgency
Spare Pool	needs always resources	1
A	need more then 3 resources	50
B	need more then 3 resources	40

The Policy Engine weights the urgencies of the reported needs by multipling the priority of the service:

Service	Urgency * Priority	Weighted Urgency
Spare Pool	1*1	1
A	50* 2	100
B	40*3	120

The police engine reports the needs with new calculated urgencies to the Resource Provider. The Resource Provider gives service B the signal that the free Resource from the Spare Pool can be assigned. After the assigment is finished service B send an event to the Resource Provider. The next scheduling run starts.

Warning

Missconfiguration of the SLOs and the policies will lead into a swinging system. We have to implement mechanisms to prevent such situations.

There is no strict definition of a policy setting - Policy Engine is open for 3rd party enhancements, therefore it does not rely on any special definition/implementation of a policy setting. An example of a policy setting can be the following rule :

Example 1.8.

<SERVICE_CONTAINER> should receive [N]% of <RESOURCE> resources

Hedeby currently embrace only a simple Policy Engine implementation which does take into account only Priority setting. Priority is value assigned to Service adapter (generally to a managed service) and is subjective importance of service (defined by an Hedeby administrator).

Decision Process in Detail

The decision process is based on an algorithm that takes into account the requirements of the service which are specified by Need and a data provided by a policy manager.

> **Note**
>
> By Resource Provider (RP) we understand an interfaces that encloses a set of managers that are responsible for whole decision making process (service manager, resource manager, request processor, order processor).

> **Note**
>
> Need is a quantified request for a resource with certain properties. One possible sample of Need: "4 resources of host type with 4GB of memory" which means that a service asks for 4 hosts with 4GB of memory. Another possiblity of Need: "1 resource of SW license type" which means that a service asks for a license (for the special SW).

The complete algorithm can basicaly be divided into solving the two cases:

- The service asks for a new resource.
- The service is giving up one of its resources.

The first case is in detail described in the following steps:

1. When a service's SLO is not met, the service (let's name it SOURCE) sends a notification to a RP that it needs a resource (ResourceRequestEvent). It is up to service to send a ResourceRequestEvent everytime it finds out the SLO is not met.
2. RP receives a ResourceRequestEvent that contains the name of the SOURCE and the list of Needs (which contains one or more Need).
3. RP enqueues the ResourceRequestEvent in the internal request queue (a request object was created from the ResourceRequestEvent).
4. RP takes the request from the queue and starts to process it.
5. If the end of the request's list of Needs is reached, go to step 6 otherwise for each need from the list of Needs in the request do the following steps:
   a. Obtain a list of resources from each service (let's name such service TARGET) that match the required resource described in Need. Let's call each such resource a CANDIDATE.

      TARGET will consider a resource as a CANDIDATE if the resource usage level is lower than the normalized urgency of the need expressed by SOURCE. Normalized urgency of the SOURCE (and of the TARGET) is calculated using the policy manager.

   b. If the list of CANDIDATE resources is not empty, continue on the next step, otherwise go to step e.
   c. Iterate over the list of CANDIDATES. If the end of the CANDIDATE list is reached go to the step e OR if the required amount of CANDIDATES was asked to be released go to step d otherwise for each CANDIDATE do the following steps:
      i. First, register an action which has to be taken once the CANDIDATE is released from TARGET. The registration is done creating the ORDER and storing it in the ORDER store. The action is an assigning of the CANDIDATE to the SOURCE (refresh: SOURCE is the service which expressed NEED).
      ii. Ask the TARGET to release the CANDIDATE (the asynchronous call to removeResource on TARGET interface will be called, and the result of the operation is NOT guaranteed). If there was problem requesting the TARGET to release the CANDIDATE, the previously created assignement ORDER is cancelled (removed).
      iii. If the required amount of CANDIDATES was asked to be released go to the next step. (Required amount is specified by quantity attribute in the Need).

d. RP filled the need (was able to ask at least the same amount of resources (CANDIDATES) to be released as it is specified by need's quantity). Remove the need from the list of needs in the request. Go to step 5.

e. RP did NOT fill the need (was not able to ask at least the same amount of CANDIDATES to be released as it is specified by need's quantity). The quantity attribute of the need is reduced by number of those CANDIDATES that RP was able to ask the related TARGET to release. Leave the need in the list of needs. Go to step 5.

6. If the list of needs in the request is not empty re-submit the request in the request queue for re-processing at later time, otherwise remove the request (it is processed).

The second case is in detail described in the folowing steps:

1. RP receives a ResourceRemovedEvent that contains the name of the service that released the resource and the snapshot of the Resource that was released. Let's name the service as a SOURCE and the resource as a RESOURCE.

2. RP looks into a REQUEST queue if there is an ORDER for the RESOURCE (the resource ID and SOURCE service (owner) is compared).

3. If there is an ORDER for a RESOURCE, process the ORDER. ORDER contains identifier of a service to which has to the RESOURCE be assigned. Let's name such service a TARGET.

   RP asks the TARGET to add a RESOURCE (the asynchronous call to addResource on TARGET interface will be called, and the result of the operation is NOT guaranteed).

   If there is a problem with executing the ORDER, the ORDER is not executed and is cancelled (removed) and the RESOURCE is assigned to the first service that is willing to accept the RESOURCE. If no such service exists, RESOURCE remains temporarily stored in RP cache and administrator has to solve the resource manually. If the ORDER is executed without any problem, all ORDERS for the same RESOURCE are removed from the resource queue (they were added based on decision that was made before the RESOURCE was added to the SOURCE).

4. If there is no ORDER for a RESOURCE, add the resource to the first available service.

Reporter

Reporter component is a log/monitoring tool for Hedeby.

The role of reporter component is to intercept and gather informations about what is going on in the system. Administrator can specify what kind of data he is interested in.Reporter is able to store informations and notifications that comes from Configuration Service, Resource Provider and all services that are installed in the system.

The reporter component is prepared to store data in ARCo data base (Grid Engines Accounting and Resourting Console). By prepared we mean that, there is a special ARCo format file created, that stores suitable for ARCo data.

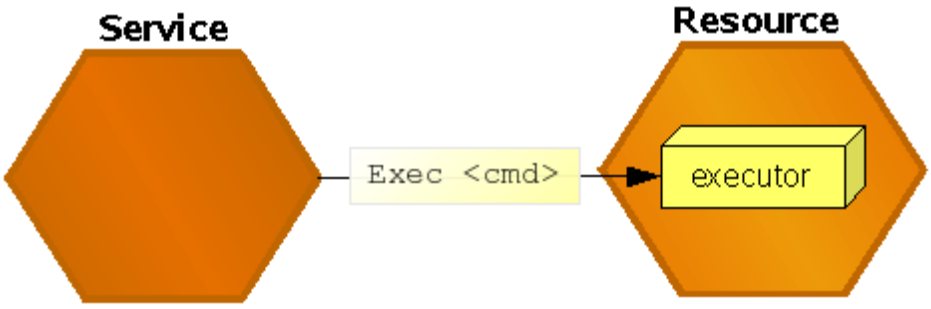
The data from ARCo file aren't so much readable for normal user, thats why Administrator can get and print out on the screen data using CLI commands. The data can be filtered using available filters. More about Reporter component you can find here: the section called “Reporter Component”

Executor

Executor is used whenever there is a need to set up (or destroy) service component on a resource that has to be a part of the service, especially in situation when there is no other way how to communicate with the resource. Once the resource is configured by executor, service adapter can use different way of communication with the resource (usually a communication channel provided by the managed service).

Executor give Hedeby the possibility to execute actions or commands on a resource. For this purpose in a Hedeby system the administrator can install on each resource an executor component. The features of executor component highly depends on the type of resource. In general the executor executes a command on a resource. For host resources user switching will be possible.

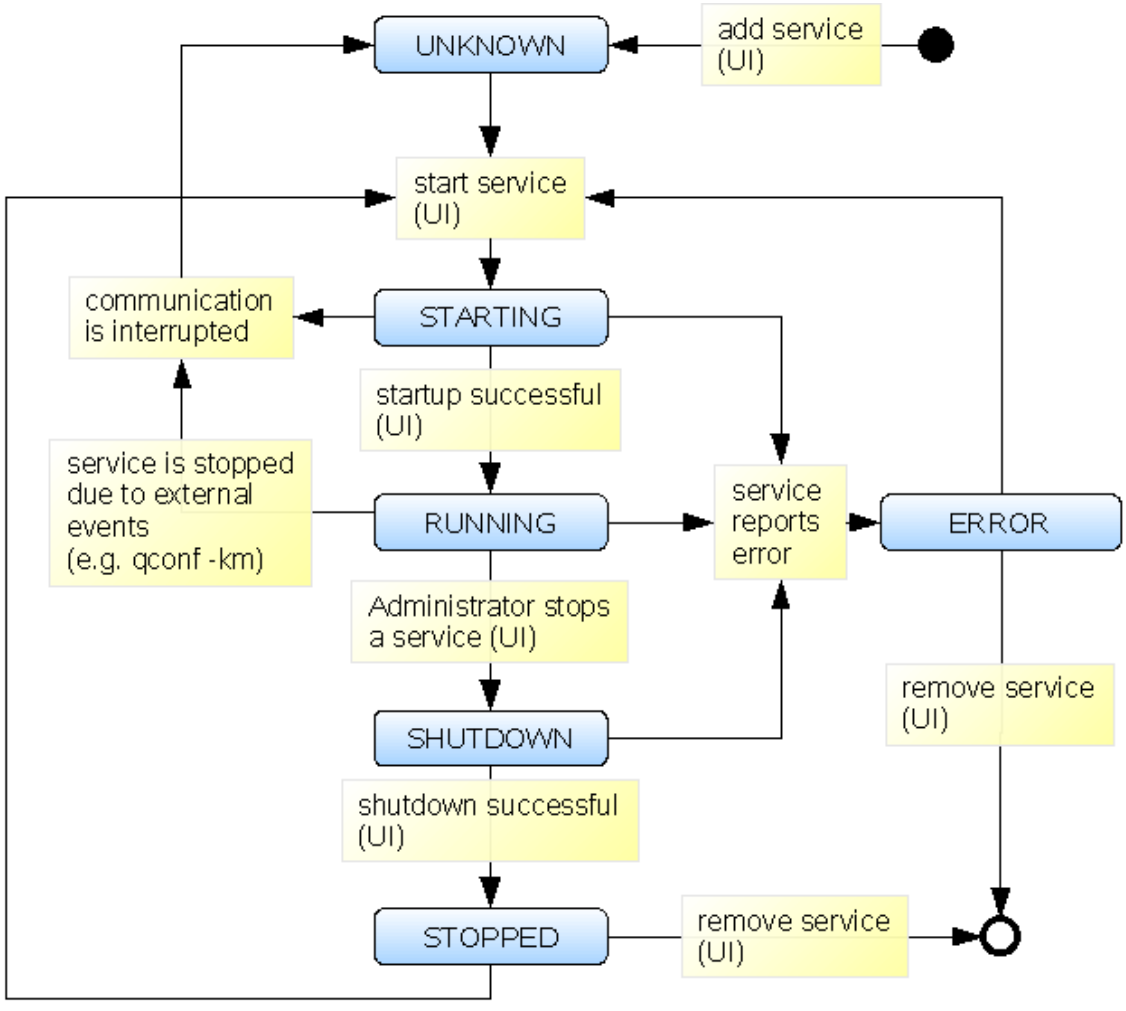
Mainly the service apdaters will use the executors for installing/uninstalling software on a resource. However the usage of executors is not restricted to the service adapters.



Principles of Operation

This section describes the basic actions or use case which can be executed on a Hedeby system.

Managing Services



Service state transistion

Add a Service

Adding a service is triggered from the UI. The adminstrator has to provide the following information.

- Name of the service, must be unique in the Hedeby system
- Type of the system (e.g. Grid Engine, RDBM system, ..)
- Connect parameter (e.g. SGE_ROOT and SGE_CELL for Grid Engine)
- Mapping of resource properties from the service conventions to the Hedeby conventions

  arch=lx26-x86 -> hardwareCpuArchitecture=x86, operatingSystemName=Linux
- Service level agreement (set of SLOs, depends highly on the type of service, different services supports different types of SLOs)
- Service specific configuration (parameters for a specific service adapter)

When adding a service the Hedeby system validates the configuration parameters. On any error the action is rejected. With a valid configuration the service adapter is instantiated. State of the service is UNKNOWN. The service is registered in the RP.

Remove a Service

Removing a service is only possible if the service is in state SHUTDOWN or ERROR. This action is triggered from the UI. The following steps are executed:

- The instance of the service adapter is removed
- The configuration of the service is removed
- The service is unregistered from the RP

Start a Service

Starting a service can be triggered from the UI. The administrator has to provide the name of the service.

- If the service is not in state UNKNOWN or SHUTDOWN the action is rejected
- The UI sends the service adapter the start_service event
- The service adapter connects to the real service
- After a successful connect the service adapter discovers the resources that are owned by the service. All resources are reported to RP. Resource that are unknown in the Hedeby system are created automatically by RP. If service adapter is not able to manage discovered resource without presence of Executor component (implementation detail of service adapter and it may vary across different service adapters) and the discovered resource is not running Executor component, the resource is marked as static.
- Finally the service adapter sets the state of the service to RUNNING and informs the RP about the state change.

Note

The service adapter does not observe the service if the service adapter is in state UNKNOWN or SHUTDOWN. If a service is started without Hedeby it has no effect on the Hedeby system.

Stop a Service

There exists two possiblilities to stop an service:

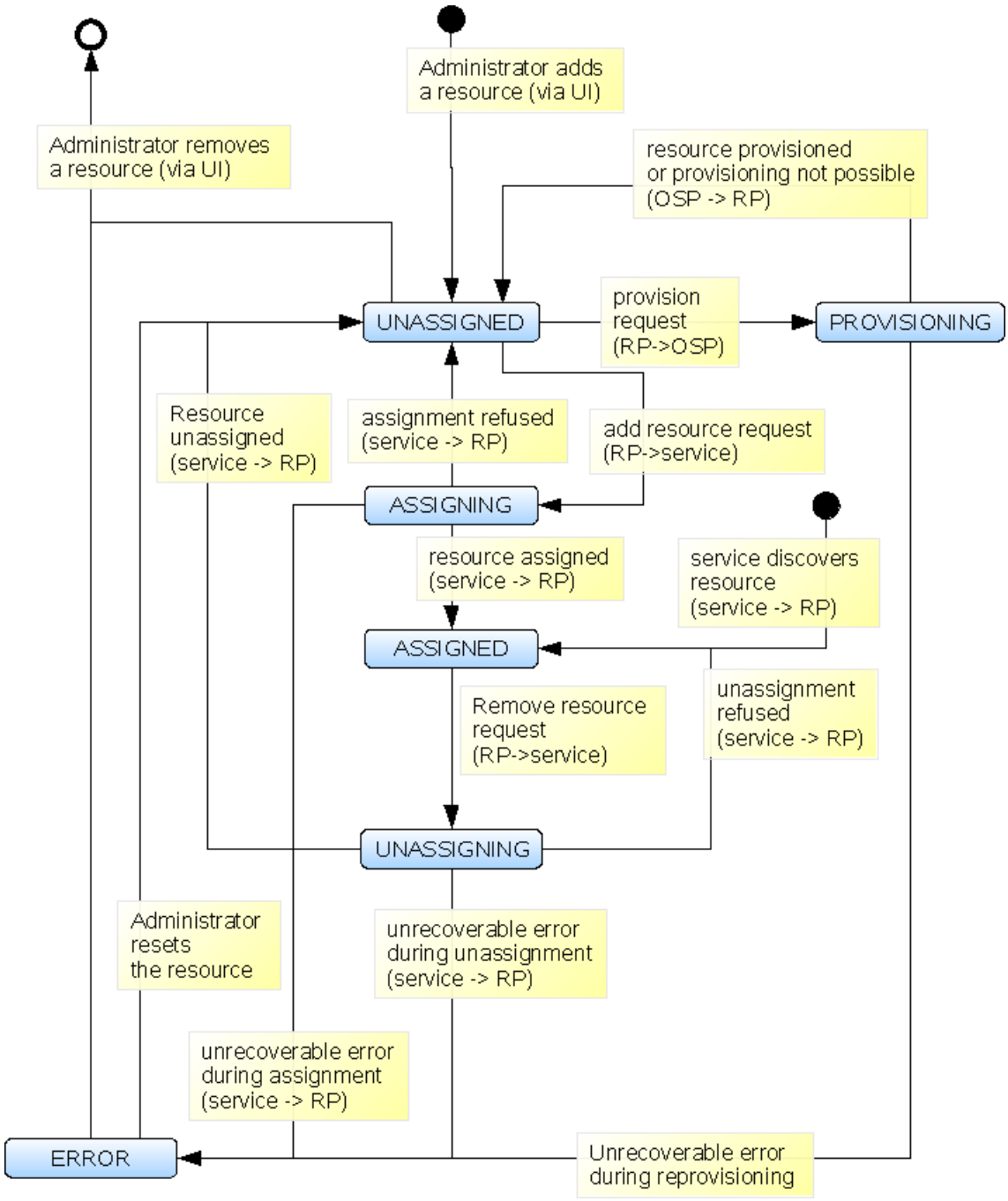
1. The administrator stop the service manually (via UI). The administrator has to specify the name of the service and the “free_resources” flag. The following steps are executed.
   - The service will go into state SHUTDOWN
   - If the “free_resources” flag is set the service adapter removes all assigned non-static resources.
   - The connection to the external service will be closed and the state of the service is set to STOPPED (event to RP).
2. The service is stopped with external tools (e.g. qconf -km). The service status is immediatly set to STOPPED and the RP is informed about the state change.

Configure a Service

Configuring a service is done via the UI. Some configuration parameters can only be changed if the service is not running (e.g. connection parameters). Other parameters can be changed dynamically (e.g. changing SLOs).

Managing Resources

The following shows all possible state transitions of resource in a Hedeby system. Each state transition requires a couple of component interactions.



Resource state transitions

Add a Resource

For adding a new resource to a Hedeby system the administrator uses the user interface. For adding a resource the administrator has to specify the following information:

- Name of the resource, must be unique within a Hedeby system.
- Type of the resource (e.g. host, printer, license)
- Additional properties for the resource (installed operating system, architecture, hardware specific properties)

When adding a resource the UI sends a corresponding request direct to the Resource Provider. The Resource Provider validates the input parameters, stores the resource in it's local storage and assigns the resource to the first service willing to accept the resource.

Resource is added automatically to the Hedeby system each time a service adapter discovers that a service uses a resource that is unknown in the Hedeby system. Such resource may be marked as static if service adapter is not able to remove the resource from service. State of a discovered resource reflects the actual resource state depending on the service adapter (for GE adapter, it may be ASSIGNED if discovered resource has execd running, or ERROR if discovered resource has not execd running).

Assign a Resource to a Service

The assignment of a Resource can be triggered in two ways:

1. The administrator uses the UI to assign a resource manually.

2. The RP finds out that a resource requires additional resources. The RP will trigger the resource assignment automatically.

For a Resource assignment the following actions are executed:

1. RP sets the state of a resource to ASSIGNING.

2. RP sends a *add_resource* request to the service (it contains the Resource properties)

3. The service checks whether Resource fulfills the requirements for this.

4. If the resource is not usable the service sends resource provider a *resource_rejected* message. The RP sets the state of the resource to UNASSIGNED. The RP marks the resource ID of the resource as not usable by the service (RP's internal storage called service blacklist).

5. If the resource is usable the service starts the necessary installation process (installation routines, depending on a service adapter). If RP received the success response message from the service (*resource_added*) it sets the state of the resource to ASSIGNED. On any unforeseen error during the installation phase the RP will set the resource in the ERROR state because the service adapter has modified the resource and the modification is undoable.

Unassign a Resource

The same as with the assignment process the unassignment can also be triggered manually (over the UI) or automatically (RP). The following actions are executed during the unassigment:

1. The RP sets the resource state to UNASSIGNING and sends the service the remove_resource request.

2. The service checks if it is possible to remove the resource.

3. If removing is not possible and the remove resource request is not forced, then the service sends the response message to RP, the state will be set to ASSIGNED.

4. If removing the resource is possible or if the remove resource request is forced the service processes the uninstall procedure. On success the RP sets the state of the resource to UNASSIGNED. On any unforeseen error the resource is treated as unusable for the Hedeby system and the state of the resource is set to ERROR.

Remove a Resource

Removing a resource is possible if the resource is owned by a service (it depends on a service adapter to check the resource state to allow/disallow the removal of resource, ideally only resource in

ASSIGNED and ERROR state can be removed). The administrator uses the UI to trigger this action. Only the name of the resource must be specified.

Administrator can remove a resource even if the resource is owned by resource provider (the resource state is not checked as this operation should be performed only if system is in inconsistent state). The administrator uses the UI to trigger this action. Only the name of the resource must be specified.

Reset a Resource

Any unforeseen error during assigment/unassigment sets a resource into ERROR state. If a resource is in ERROR state the Hedeby system treats it as unusable. If service adapter of the service that owns the ERROR resource does not support active reset of resource (automatic cleanup), the administrator must cleanup the resource manually. After the clean up the resource state can be reset manually (UI). Only the name of the resource must be specified.

GE Multi Cluster Management

The only natural candidate to prove the concept of service domain management (implemented in Hedeby) is management of Sun Grid Engine (in the next paragraphs also SGE or GE) multi cluster instance. Thanks to in-house developement of GE we were able to deliver implementation which allows simultaneous management of 2 or more GE clusters.

Components

Some of the Hedeby components (or building-blocks) posses special functionality or restrictions related to management of SGE clusters. Next paragraphs will describe in greater detail these special cases.

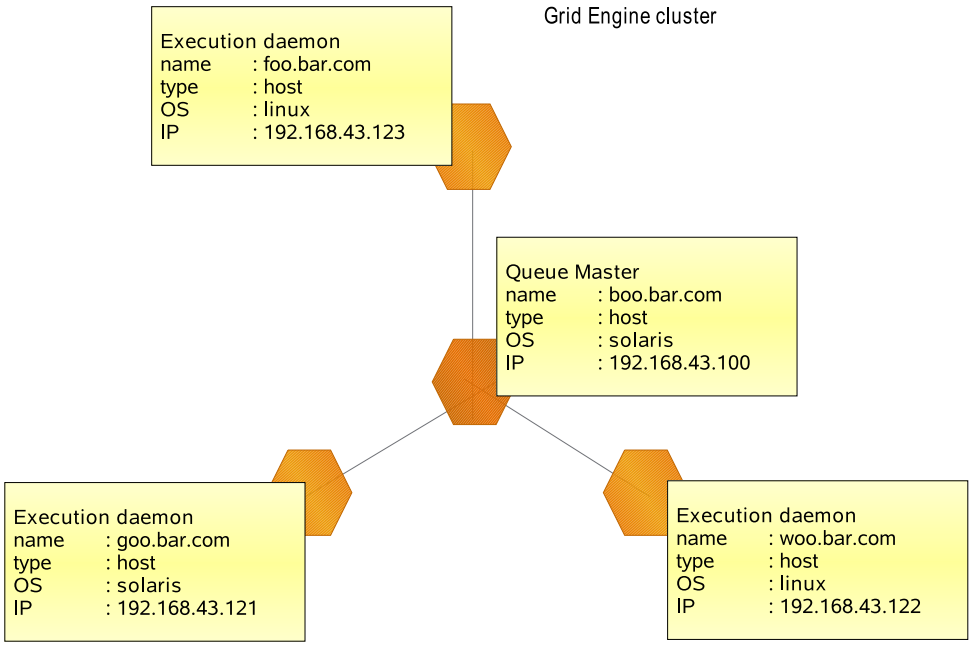
Resources

Hedeby defines resource as an aggregation of properties (each property is key/value pair which represents required attribute of the resource - for more see the section called “Resource”). Currently only one type of resources (as defined by Hedeby) is available to be used with GE that is managed by Hedeby - it is a resource which type is "HOST".

"HOST" type of resource means that resource properties are describing a host computer (machine that communicates via a network). See the section called “Resource Definition Overview” for more details about current host resource implementation.

From the GE perspective host resource could be a computer that is running an instance of GE Queue Master, an instance of GE Execution Daemon or simply any computer (node) that is part of GE cluster. Thanks to Hedeby functionality host resource for a GE managed by Hedeby can be almost any computer - if there is need (expressed by GE cluster instance) and possibility (evaluated by Agent/Executor running on a host) Hedeby can add that host to GE cluster.

The way how resources are managed (added/removed to GE cluster, daemons are started/stopped) is affected by service level objectives (SLO). Hedeby supports several SLOs (see the section called “SLO/SLA”).



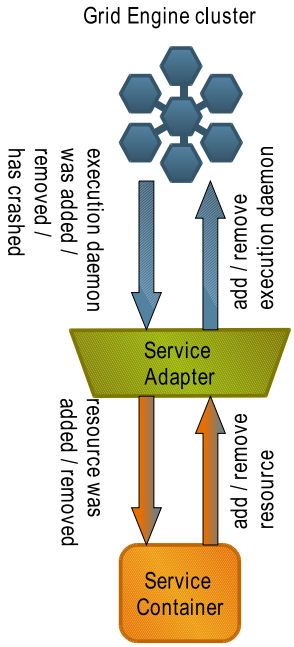
Sun GE cluster resources

GE Service Adapter

The component responsible for communication between Hedeby and managed GE is GE service adapter that acts like a bridge for communication between Resource provider and the managed GE cluster. In general, GE service adapter is kind of special driver responsible for translating "generic" management functionality needed by Hedeby to "native" functionality of a GE - for example it is able to translate "remove resource" message originating from Resource Provider to a set of GE commands, that results into :

- 1. Stopping a GE daemon (if running) on the host resource.
- 2. Removing host (node) configuration from GE.

GE service adapter is also responsible for providing information that is needed by Resource Provider . It reports cpu load values, memory consumption, number of pending jobs etc. - hence everything that helps Resource Provider to make qualified decisions to satisfy needs of managed services.



Communication between service adapter and GE

Current implementation of GE service adapter is using custom Java interface to obtain vital information from underlying GE (this API is known as JGDI).

There is no 1:1 or similar relation between GE service adapter and GE components (queue master, execution daemon, shadow daemon etc.), in fact GE service adapter is using a combination of all GE components that all provide information and functionality needed for management of a GE cluster.

GE service adapter responsibilities currently involves:

Table 1.1. GE service adapter functionality

Feature	Details
Installing a GE execution daemon on a host resource	Used when adding a new host resource to GE service. See the section called “ Executor ” and the section called “ Use cases ”.
Uninstalling a GE execution daemon from a host resource	Used when removing a host resource from GE service. See the section called “ Executor ” and the section called “ Use cases ”.
Adding a resource to GE	Only adding of a new host resource to GE is currently supported. See the section called “ Executor ” and the section called “ Use cases ”.
Removing a resource from GE	Only removing of a host resource from GE is currently supported. See the section called “ Executor ” and the section called “ Use cases ”.
Listening for a host related events dispatched from GE (host added, removed, modified)	Resource management is done in asynchronous way in GE service adapter - it does not wait until a host resource management task finishes, instead it listens for an event that is dispatched by the GE itself (when host is added to GE, removed from GE or modified) when task is finished.
Computing GE needs	GE service adapter periodically gathers data that are needed to evaluate whether configured SLOs are met or not. See the section called “SLO/SLA” for more details about SLOs.
Dispatching service related events to service adapter	All changes in the GE characteristics are reported as an events back to service adapter - this means all host related events and SLO non-compliance events.

Service adapter concepts are described in greater detail at the section called “Service”. More information about configuration is at the section called “Service Adapters, Grid Engine Adapter ”.

Resource Provider

Although Resource Provider (RP in the next paragraphs) can be considered as a component that is independent of other components, its functionality has direct impact on managed GE clusters. Regarding the GE managed by Hedeby RP is responsible for following actions:

Table 1.2. GE related Resource Provider functionality

Feature	Details
Listen for normalized SLO non-compliance events from managed GE cluster	SLO non-compliance event is dispatched by GE service adapter when manged GE cluster is in need of a resource because of any of its requirements is not met. These requirements (called service level objectives) can be: • Minimum number of nodes • Minimum number of pending jobs • Etc. The fact whether SLO is met or not met is evaluated by GE service adapter. For complete listing of supported SLOs see the section called “SLO/SLA”.
Locate resources to address reported SLO non-compliance events	If any of managed GE clusters expresses need for a resource, Resource Provider will look amongst known resources for a "free" resource. In scope of Resource Provider known resources are all resources gathered in Spare Pool and all resources assigned to other managed GE clusters (or other services). Actually, Resource Provider will look for any resource that can be taken and assigned to GE cluster that expressed need - GE cluster will just provide list of their nodes. See the section called “Resource Provider” for more details about Resource Provider and the section called “ Use cases ” for more details about satisfying the needs of GE.
Order GE to add/ remove resources	Hedeby resource management of GE cluster reflects as starting/stopping GE daemons on hosts and adding/removing hosts to GE cluster (cell). More details can be found at the section called “ GE Service Adapter ” and the section called “ Use cases ”.
Listen for error notifications from a GE cluster	If there is unexpected behaviour in the components that handle GE cluster management, an error notification is dispatched (originating from executor or service adapter). Typical case is problem in starting the GE daemons, adding new node configuration etc.

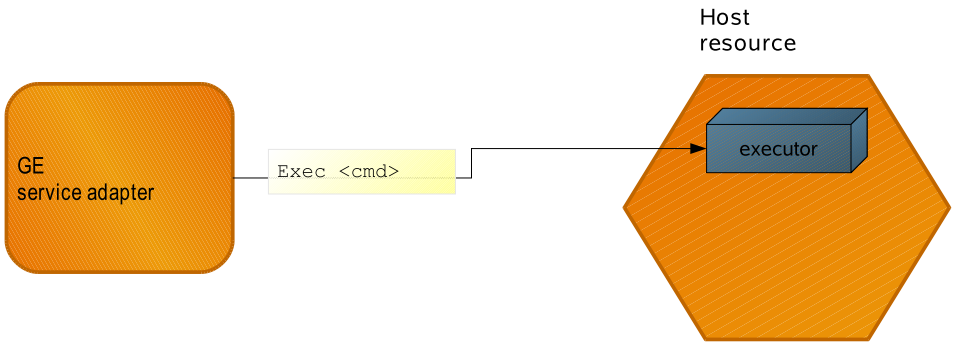
Executor

GE adapter is using executor for the following set of actions:

Table 1.3. GE related Resource Provider functionality

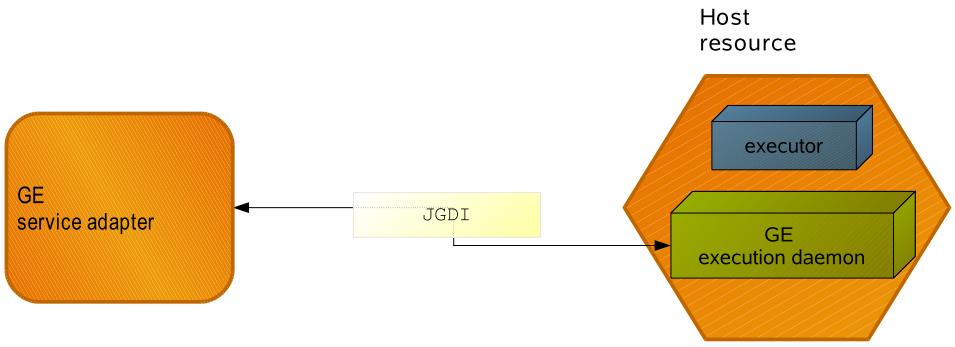
Feature	Details
Installing GE execution daemon on a host resource	Used when there is no execd running on the host resource and the resource was assigned to be part of a GE cluster that is managed by Hedeby. See the section called “ Use cases ”.
Uninstalling GE execution daemon on a host resource	Used when there execd running on the host resource and the resource was assigned to be removed from a GE cluster that is managed by Hedeby. See the section called “ Use cases ”.

The executor commands that handle functionality related to GE (installation/uninstallation of execd, etc.) are currently implemented as shell commands or shell scripts (GE has high quality CLI interface which allows to do that).



Service Adapter and executor on a host resource

Once the executor daemon is installed on the host resource grid engine service adapter will use GE Java api to retrieve information from (or pass to) the resource.



Service Adapter and executor on a managed host resource

See the section called “Executor” for general concept of executor and the section called “Executors” for configuration details.

SLO/SLA

According to Hedeby project specification, SLO defines the acceptance state of a given service attribute (with appropriate metrics). Very simple example of SLO for a GE service is that "the number of pending jobs should be less than 100" in a GE cluster. More about SLO can be found at the section called “Service Level Objectives (SLO) ”.

The original plan was to support following SLOs for managed grids :

Table 1.4. Originaly planned SLOs

SLO	Details
Minimum number of hosts	Allows to set minimum number of host
Minimum number of hosts of a given architecture	Allows to set minimum number of host of certain architecture
Maximum number of pending jobs	Allows to set maximum number of pending jobs
Maximum number of pending jobs over a given time period	Allows to set maximum number of pending jobs over a specified time period
Maximum number of pending jobs for a given architecture	Allows to set maximum number of pending jobs for the host of certain architecture
Maximum average number of pending jobs over a given period for a give architecture	Allows to set maximum average number of pending jobs for the host of certain architecture
Minimum number of free slots	Allows to set minimum number of free slots
Minimum average number of free slots over a given time period	Allows to set minimum number of free slots over a specified time period
Minimum average number of free slots for a given architecture	Allows to set minimum number of free slots for the host of certain architecture
Minimum average number of free slots over a given time period for a given architecture	Allows to set minimum number of free slots for the host of certain architecture over a specified time period
Fixed host	Allows to set a bunch of static hosts

While priority has been placed on other core Hedeby features, only a subset of the above mentioned SLOs has been implemented so far. Currently supported SLOs are summarized in the next table (with short comments).

Table 1.5. Currently implemented SLOs

SLO	Details
Minimum number of hosts	Allows to set minimum number of host. Current implementation allows to add filter for hw architecture so this SLO is also implementation of "Minimum number of hosts of certain architecture" SLO
Maximum number of pending jobs	Allows to set maximum number of pending jobs. Current implementation allows to add filter for hw architecture so this SLO is also implementation of "Maximum number of pending jobs of certain architecture" SLO
Fixed host	Allows to set a bunch of static hosts

Each of currently implemented SLOs has several attributes that can be customized. See samples with short explanation in the following tables.

Table 1.6. Minimum number of hosts

Attribute	Details
Attribute	The GE attribute name to monitor
Value	The value the GE attribute needs to have to be part of the SLO
Urgency	The urgency the resources should have that are part of the min amount
Min	The mininum value of the specified attribute / value pair

Table 1.7. Maximum number of pending jobs

Attribute	Details
Attribute	The GE attribute name to monitor
Value	The value the GE attribute needs to have to be part of the SLO
Urgency	The urgency the resources should have that are part of the min amount
Max	The maximum number of pending jobs for the specified attribute / value pair

Table 1.8. Fixed host

Attribute	Details
Host	Static resource identifier - hostname of the static host

Primary goals

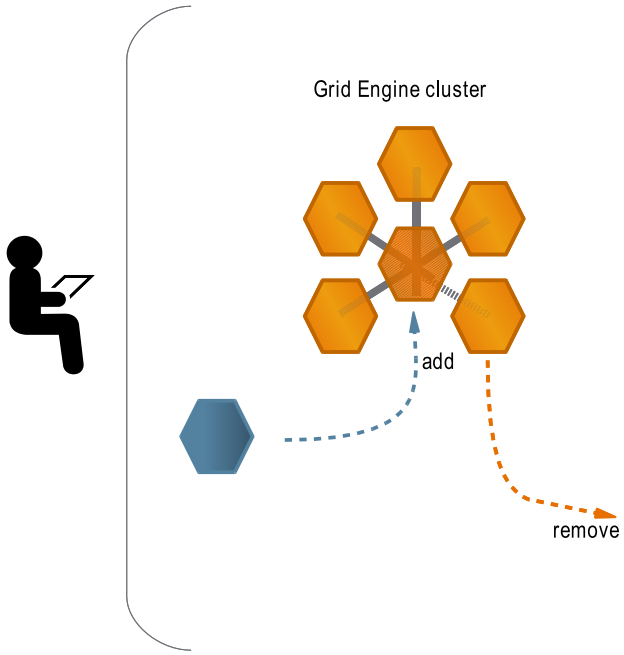
Primary goal of Hedeby's support for GE was to deliver automatic resource management tool for GE clusters. The tool should allow unattended configuration of GE cluster to deliver optimal job throughput and utilization of various resources assigned to GE cluster. This chapter focuses on a description of Hedeby managing GE cluster, its performance and usability.

Use cases

Due fact that only host resource are currently supported core Hedeby functionality related to GE cluster management can be described by relatively short set of use cases. These use cases are compiled in the following chapters.

GE cluster without Hedeby

The only instance of GE cluster (regardless of total nodes in the cluster) that is not managed by Hedeby is the simplest example of Hedeby functionality. As the GE cluster instance is not managed by Hedeby, there is not much that Hedeby can do - in fact it does nothing. All resources (the only supported resource type is "HOST" therefore resources equals nodes) are staticaly assigned to GE cluster.



Sun GE cluster managed by administrator

Only the GE administrator can add/remove nodes to cluster or start/stop GE daemons on nodes. To increase a performance of a GE cluster, administrator would have to perform following steps.

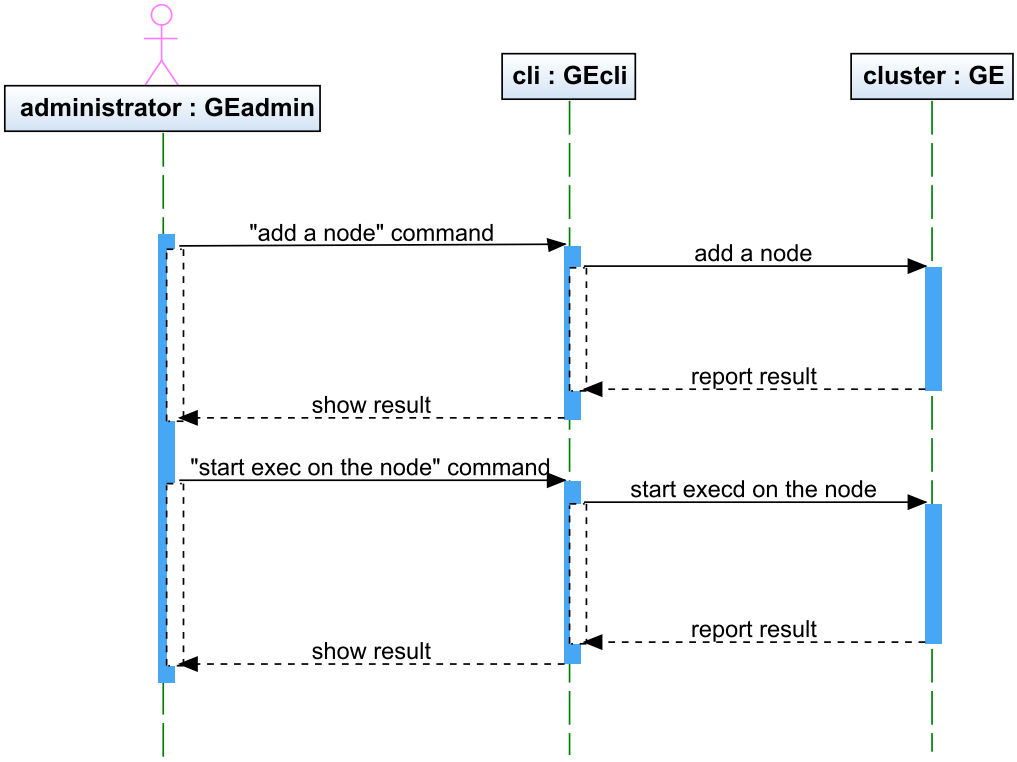
Procedure 1.1. Increasing the performance of a GE cluster

1. Add the node configuration to the GE cluster.

   Administrator would use GE administrator interface (qconf or qmon).

2. Start the grid daemon (execution daemon) on the node.

   Administrator would use shell script to start the daemon (sge_execd).



A resource added by GE administrator - sequence diagram

Similar procedure is needed when administrator has to remove a node from a GE cluster (to use it in different GE cluster, for example)

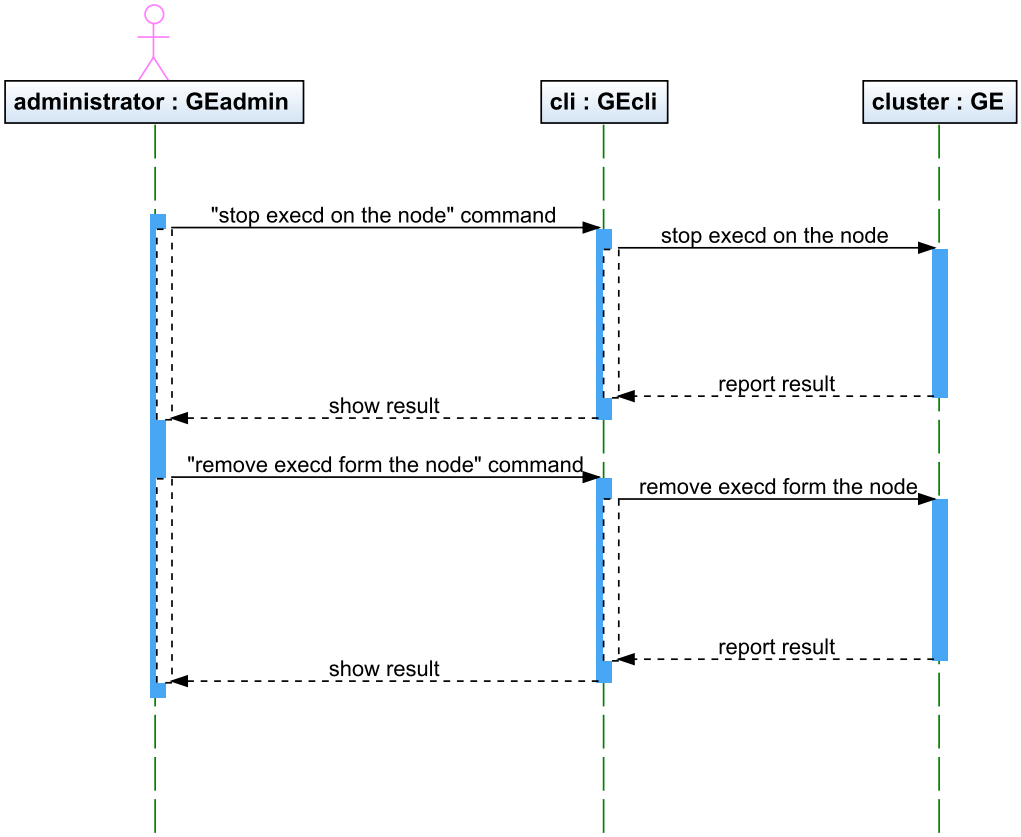
Procedure 1.2. Decreasing the performance of a GE cluster

1. Stop the grid daemon (execution daemon) on the node.

   Administrator would use shell command to kill the daemon (**pkill sge_execd** or **qconf -ke**).

2. Remove the node configuration from the GE cluster.

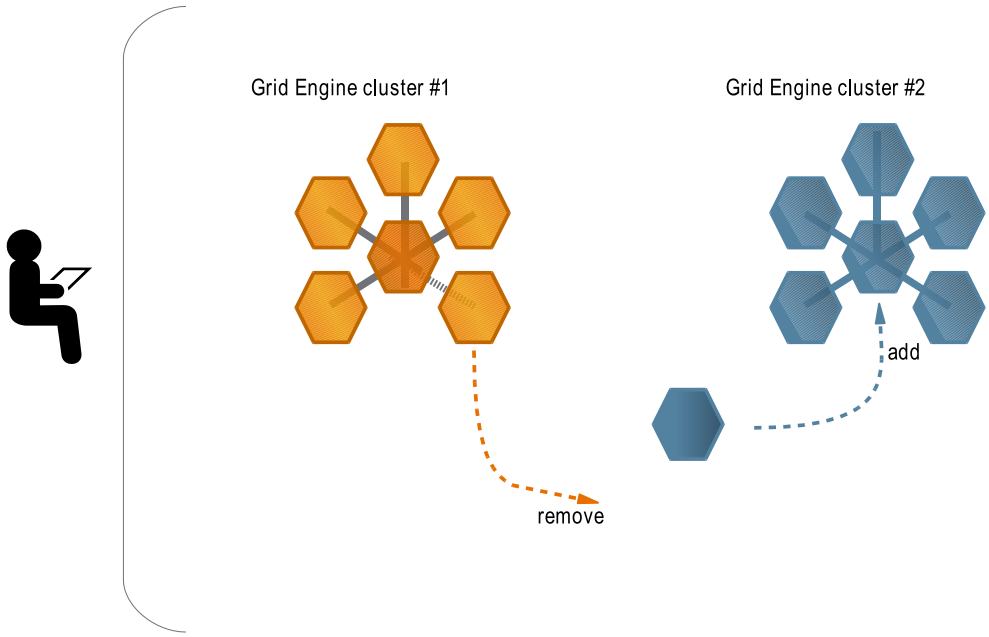
   Administrator would use GE administrator interface (qconf or qmon).



A resource removed by GE administrator - sequence diagram

Two GE clusters without Hedeby

Follow up to previous use case is the situation of two (3,4, many) instances of GE cluster (again, regardless of total nodes in the cluster) that none of them is managed by Hedeby. And again, as none of the instances is managed by Hedeby, Hedeby does nothing in this use case. All resources (the only supported resource type is "HOST" therefore resources are GE nodes) are statically assigned to a corresponding GE cluster.



Two Sun GE clusters managed by administrator

Only the GE administrator can add/remove nodes to cluster or start/stop GE daemons on nodes. To increase a performance of a GE cluster #2 using a node taken from a GE cluster #1 (assuming that it has spare capacity), administrator would have to perform following steps.

Procedure 1.3. Increasing the performance of a GE cluster #2 using a spare resource from a GE cluster #1

1. Stop the grid daemon (execution daemon) on a node that is part of cluster #1.

   Administrator would use shell command to kill the daemon (**pkill sge_execd** or **qconf -ke**).

2. Remove the node configuration from the GE cluster #1.

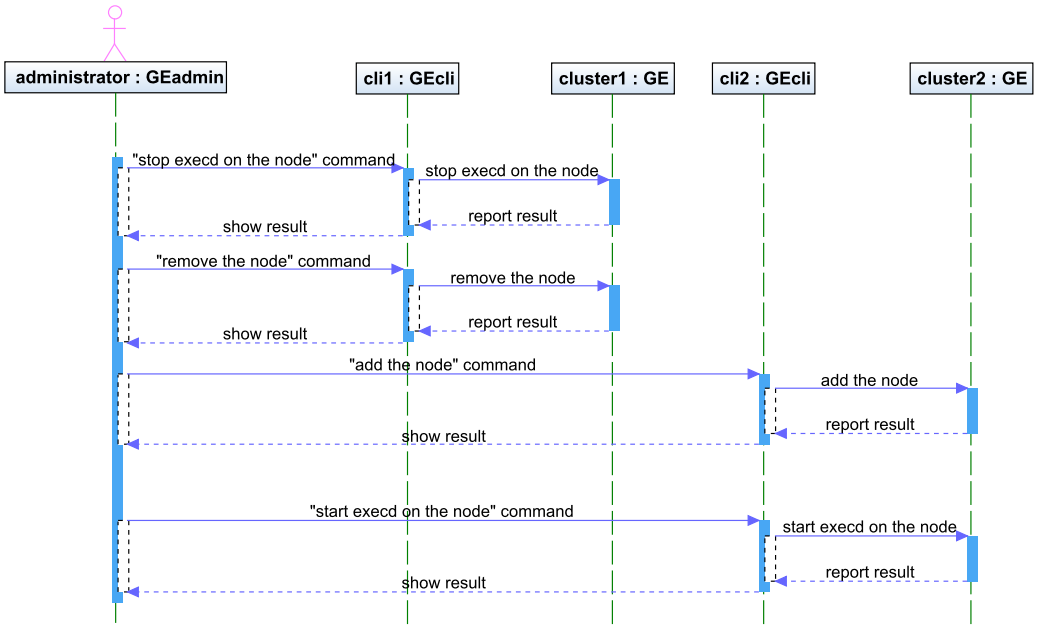
   Administrator would use GE administrator interface (qconf or qmon).

3. Add the node configuration to the GE cluster #2.

   Administrator would use GE administrator interface (qconf or qmon).

4. Start the grid daemon (execution daemon) on the node that is now part of the GE cluster #2.

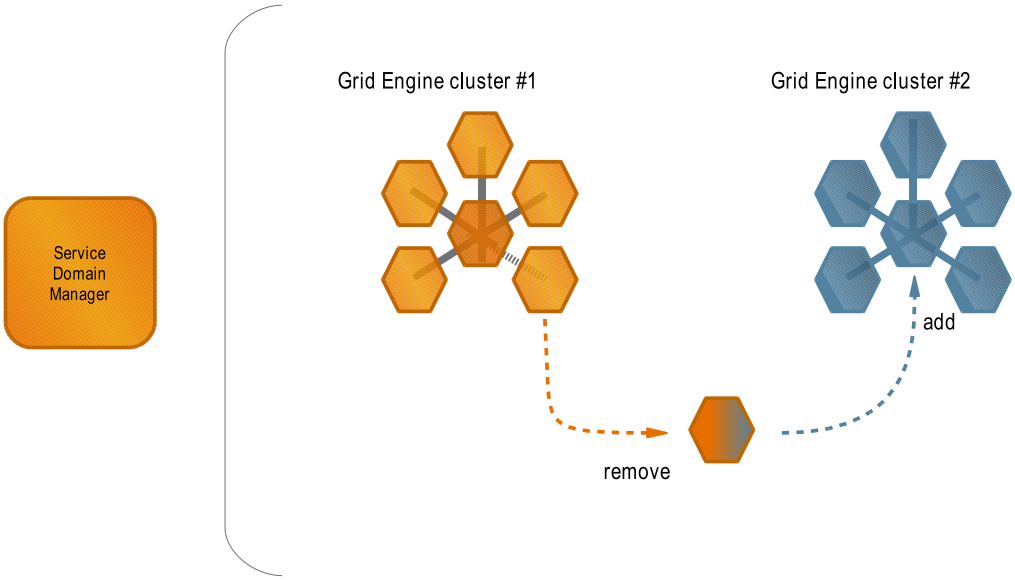
   Administrator would use shell script to start the daemon (sge_execd).



A resource shuffled between two grids by GE administrator - sequence diagram

Two GE clusters with Hedeby

Logical "upgrade" to previous use case is the situation where thoe instances of GE cluster are managed by Hedeby - from the Hedeby's view, those GE clusters are managed services. Each resource (the only supported resource type is "HOST" therefore resource equals node) are dynamically assigned to GE cluster which expresses need for additional resource (with the only exception of static/fixed hosts). Hedeby is therefore responsible for automatical adding/removing of nodes to relevant cluster or starting/stopping of GE daemons on related nodes.



Two Sun GE clusters managed by Hedeby

If a managed GE cluster #2 expresses need for a resource and if there is another managed GE cluster #1, Hedeby could take following set of actions.

Procedure 1.4. Hedeby satisfies need of a managed GE cluster #2

1. SLO of the GE cluster #2 is not met

   Service Adapter for the GE cluster #2 gathers information and finds that at least one of the configured SLO for the GE cluster #2 is not met.

2. Resource provider receives non-satisfied SLO event from GE cluster #2

   Resource provider looks for any resource that can be taken and assigned to the GE cluster #2. Resource provider considers any resource that:

   - is assigned to a service (GE cluster) with lower priority than the GE cluster #2
   - is not marked as fixed resource
   - is owned by a Spare Pool

   Regarding on the details of the need, some of the resources are be left out (if they do not match needed host architecture, operating system version etc.).

3. Resource provider finds an appropriate resource.

   (Assuming that the only available resource matching expressed need for a resource is assigned to a GE cluster #1) Resource provider asks the related service container to release the resource.

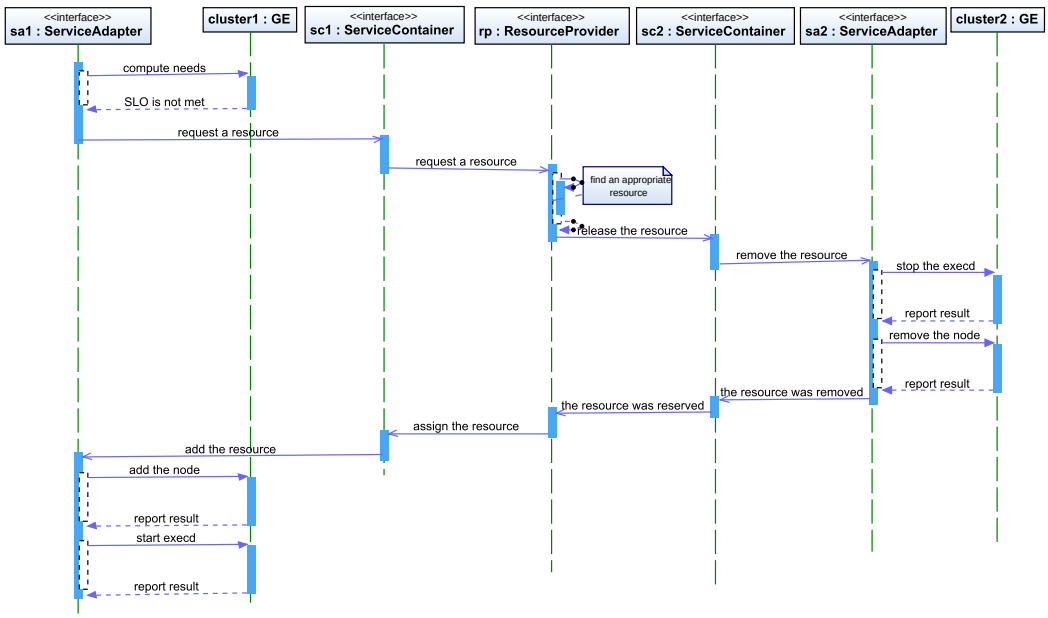
4. The GE cluster #1 removes the node.

   Service container dispatch to its service adapter message that the resource should be released form the managed grid engine cluster #1. GE service adapter contacts the executor running on the node with order to stop the grid daemon (execution daemon). Service Adapter then removes the node configuration from the GE cluster #1.

5. Resource provider assigns the resource to GE cluster #2.

   Once the resource (node) is released from GE cluster #1, it is temporarily marked as "RESERVED" which makes it reserved fo assignment to GE cluster #2. Resource provider asks service container related to GE cluster #2 to add the resource. Service container delegates the message to its service adapter which first adds a configuration for the node to the GE cluster #2. Service Adapter then again contacts executor running on the node with order to start the GE daemon (execution daemon) - this time the node will be part of the GE cluster #2.

Although the above procedure looks kind of complicated, it is worth to notice that it does not involve any single interaction of a human administrator.



A resource shuffled between two grids by Hedeby - sequence diagram

Juggling resource back and forth between managed GEs is not always enough to satisfy their needs - sometimes it is more convenient to add additional (new) resources to managed grids (typical example is buying a new hardware). With Hedeby there is no need first to manually add such resource to one of the managed GE clusters - it is possible to add such resource to Spare Pool and let Hedeby decide what to do with it.

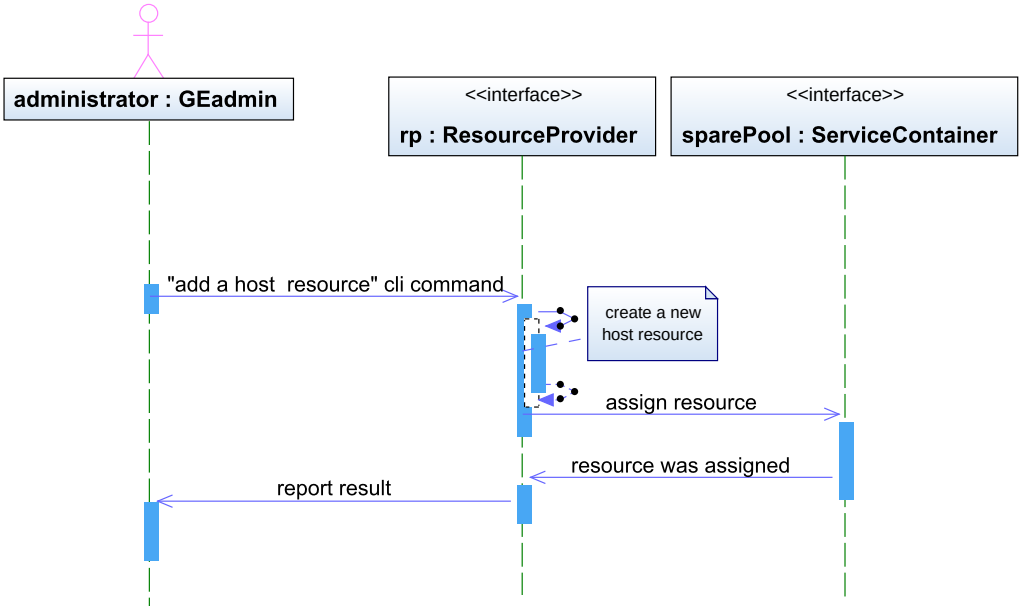
Procedure 1.5. Adding a resource to Hedeby Spare Pool

1. Administrator starts the executor on the resource

   Running executor is a MUST HAVE requirement to manage any of the hosts. Executor can be deployed on the resource using several ways (see the section called “Install Hedeby”).

2. Administrator adds the resource to the Spare Pool

   Administrator would use Hedeby administrator's interface - typically it would be "sdmadm add_resource" shell command. Once the command execution finishes, the resource is available to be assigned to any of managed GE clusters.



Adding a host resource to Spare Pool by GE administrator - sequence diagram

Target performance

Hedeby is supposed to manage 2 and more GE clusters (although in combination with Spare Pool it is possible to use it to manage only one GE cluster) and therefore it has to provide best scalability and performance. Test runs with 3 managed GE clusters and cca 3000 host resources showed very small memory and cpu consumption of Hedeby components, while overall response time of Hedeby system was very fast.

Next table shows list of basic actions and time that needs Hedeby to perform it.

Table 1.9. Performance and response times

Action	Details	Time (include CLI execution overhead)
Startup	JVM running CS component, JVM running RP and CA component, all JVMs running on one host	7 s
Assigning the GE cluster to be manged	This is automatical action	Less then 1s
Adding a new host resource to Spare Pool		2 s
Removing a host resource from Spare Pool		2 s
Adding a host resource to a GE cluster	GE adapter performs automatical Exec daemon installation on a host	4 s
Removing a host resource from a GE cluster	GE adapter performs automatical Exec daemon uninstallation on a host	4 s
Showing list of managed services	System was managing 3 GE clusters and 1 Spare pool	less than 3 s
Showing a resource details		2 s
Restarting the Resource Provider		8 s
Restarting the GE service		3 s
Shutdown	JVM running CS component, JVM running RP, Spare pool and Reporter, JVM running Executor and CA component, all JVMs running on same host	12 s
Shutdown	JVM running GE service and JVM running Executor, both JVMs running on same host	3 s
Shutdown	14 JVMs running various components running on 9 hosts (on some hosts more than 1 JVM was running)	16 s

The data introduced in this section were captured using following configuration:

Table 1.10. Hedeby testing system setup

Subcomponent	Details
Hedeby	Managing 2 GE clusters, both clusters having 2 execution daemons, 1 queue master and 1 shadow daemon. Hedeby used 2 JVMs - the first one was running all components except an executor, the second one running just an executor.
GE clusters	Each component (2 execution daemons, queue master, scheduler) was running in the separate Solaris 10 U2 zone on a machine equipped with 2xSparc@1.5GHz, 8GB RAM.
GE version	For testing was used Sun GE 6.2 maintrunk
Java	JDK 1.5_011 was used

Usability

The very first real-life application of Hedeby will be using it as an automatic resource manager for several Sun GE clusters. This premise has driven the implementation effort and caused that priority has been placed on certain of its features (while others are not implemented yet).

Here is the brief resume of all aspects that apply to usablity of Hedeby (in relation to Sun GE management) with short comments.

Table 1.11. Hedeby usability resume

Feature	Details
Adding a resource to a GE cluster	Either creating a new host resource using Hedeby CLI or adding a new node to GE cluster manually (Hedeby will find out that a new node was added to managed GE cluster)
Removing a resource from a GE cluster	Either removing a host resource using Hedeby CLI or removing a node from GE cluster manually (Hedeby will find out that a node was removed from managed GE cluster)
Supported GE resources	Only "host" resources are supported. Support for general GE resources (Complex Types) is planned.
Policy support	Currently there is only priority policy available for decision making process. Other policy types will be introduced in later relases of Hedeby.
Static resources	Any GE node that is running Queue master instance will automatically marked as static. In addition, any GE node running Exec daemon but not running Executor (Hedeby component) will be marked static, too.
Resource error state	Resource can be reset either using Hedeby CLI or by removing and then adding a node from GE cluster manually (Hedeby will find out that a node was removed from managed GE cluster and again added to GE cluster)
SLO definition	There is support for SLO creation/modification using CLI.

Requirements

OS and HW platforms

Hedeby as a Java application is virtually able to run on any platform that complies with the section called “Java”.

Hedeby does not rely on any special HW platform (although it contains few lines of a native code). In general it is aimed to support all platforms that are supported by GE 6.2.

Table 1.12. Tested OS and HW platforms

Operating system	Hardware
Solaris 9 (with applied updates)	Not tested yet
Solaris 10 (with applied updates)	Tested, works fine
Linux kernel 2.4	Tested, works fine
Linux kernel 2.6	Tested, works fine

Table 1.13. Known OS and HW platform limitations

Operating system	Hardware
Windows (any version)	Windows support is planned for later releases of Hedeby.
MacOS	Not tested yet.
Linux kernel xxx running on Solaris Sparc	Not tested yet.

Java

Hedeby makes strong use of Java features that were introduced in JDK 1.5 (generics, concurrency utilities etc.) which makes any pre 1.5 version of Java unusable for it. During the Hedeby's implementation phase JDK 6.0 was released and Hedeby is working fine with Java 6, too.

Table 1.14. Required Java

Version	Details
pre JRE 1.5	Not supported
Sun JRE 1.5	Tested, recommended to use
JRE 1.5 provided by other vendors (not Sun)	Not tested, should be working.
JRE 1.6	Tested
Non Sun JRE 1.6	Tested, minor issues. Not recommended to use yet.

GE

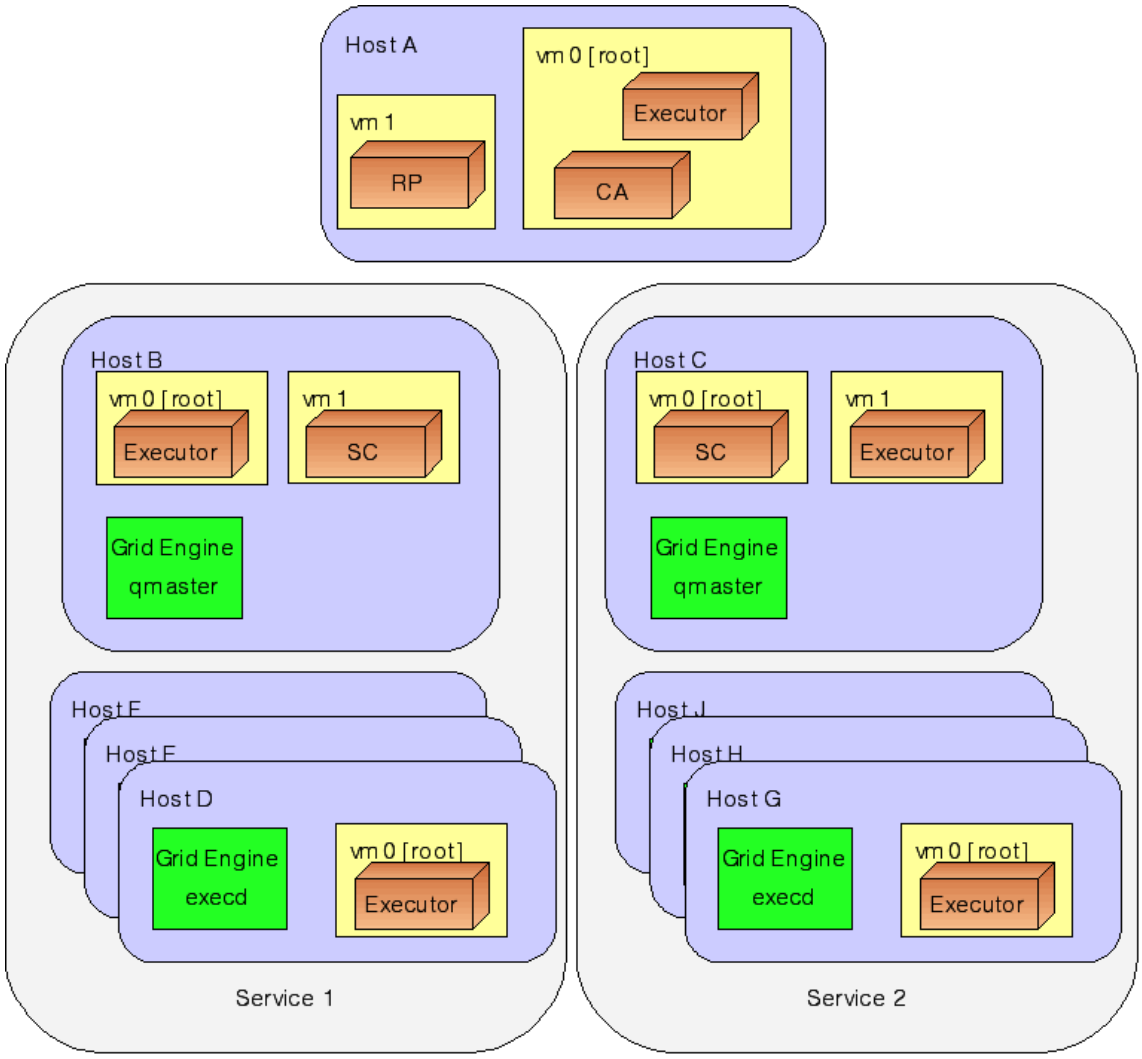
Hedeby is using to manage GE set of GE features that were introduced only in the latest updates to Sun GE and some features that are part of the development maintrunc of the Sun GE (not available for public use yet). The main restriction applies to JGDI (Java wrapper to GDI) which as a private Java GE API provides vital functionality for Hedeby and is not part of any Sun GE distribution.

Despite the restrictions stated in the above paragraph, support of the most of Sun GE releases is planned with the release of Hedeby reloaded.

Table 1.15. Supported GE releases

Version	Details
Sun GE 6.2	Tested, JGDI over JMX required
Other release than 6.2	Not supported

Chapter 2. Current Hedeby system



Typical deployment scenario for a Hedeby system

Install Hedeby

Install Hedeby

Plan the Installation

Before installing Hedeby please answer the following questions.

1. Which host will be used as master host?

   On the master host Hedeby will install three processes (Java processes). The cs_vm with configuration service component. The rp_vm with the Resource Provider, Eeporter and Spare Pool component and executor_vm with the executor and the CA component. cs_vm and rp_vm will run as sdm_admin user, executor_vm is started as user root.

   The CA component of Hedeby use Grid Engine's sge_ca script for managing the certificate authority. As consequence the Hedeby master host needs access to a Grid Engine 6.2 SGE_ROOT directory.

2. Which Grid Engine instances will be managed by Hedeby?

   For each Grid Engine instance which should be managed by Hedeby a GE Service Adapter component must be installed. This component can live in any JVM of the Hedeby. However the host where the JVM is running must have access to the SGE_ROOT directory of the Grid Engine instance. We recommend that the GE Service Adapter is installed on the Grid Engines Qmaster host.

   Hedeby can only manage Grid Engine instances 6.2 or higher. Qmaster's JMX server must be enabled.

   The Grid Engine service adapter will automatically discover the existing execution hosts of the Grid Engine instance. However it can only manage this hosts of a executor component is running on it. If you plan to manage the existing execution daemons with Hedeby you have to perform a managed host installation on this hosts.

3. Which hosts will be managed resources for the Hedeby system?

   Typically a Hedeby system will have a pool of hosts which are not assigned to any service. For each of this hosts a managed host installation is necessary.

4. Which user account is the Hedeby admin user?

   This user account must exist on any host of the Hedeby system. This will be the owner of the Hedeby distribution files. During the installation of Hedeby the security setup will create the necessary files for passwordless authentication of this user.

5. Where can Hedeby spool files?

   Each host of a Hedeby system needs a local spool directory. The default path is `/var/spool/sdm/<Hedeby system name>`. Hedeby has the following space requirements for the local spool directory:

Type of Host	**Space Requirements**
Master host	200MB
Managed host with GEServiceAdapter	50MB
Managed Host (Host Resource)	20MB

6. Which port can be used for the configuration server?

   Hedeby needs one static port for the JVM which runs the configuration service (CS port).

Requirements

Before the installation of the Hedeby system can begin please ensure Java 5 or Java 6 is installed on all of your hosts. The Java binary must be in PATH of the users which invokes Hedeby binaries. We do not recommend different Java versions on the different hosts Please perform on each host the following steps:

```
% java -version
java version "1.5.0_12"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_12-b04)
Java HotSpot(TM) Client VM (build 1.5.0_12-b04, mixed mode, sharing)
```

Once Java is available on each host please install the Hedeby binaries. Currently the distribution is only available as `tar.gz`. Please go to each host and install unpack the distrubtion. Please ensure the you have chooses the same path on each host. We recommend `/opt/sdm`.

```
% rlogin foo
% cd /opt
% su
# mkdir sdm
# cd sdm
# tar xzf <dist_dir>/sdm-1.0.tar.gz
```

Please ensure that the owner of all files of the distrubtion is the Hedeby admin user (e.g. sdm_admin).

```
# chown -R sdm_admin /opt/sdm
```

Please update your shell configuration files. Ensure that the bin directory of the distribution is in the path of the Hedeby admin user.

Install the master host

The first step for a Hedeby system installation is the installation of the master host. This step is performed with the **sdmadm inst_master_host** command. It must be started as user root. The following tasks are performed by this command.

1. Print out the software license and ask user to accept it.
2. Create the bootstrap configuration for the system. The bootstrap configuration for master host should be always installed with the system bootstrap configurations. (-p SYSTEM, see also the section called “Managing bootstrap configuration”)
3. Create the necessary files and directories in the local spool directory. Please note that the parent directory of the local spool dir must exist. The user executing the master host installation must have write permission on this directory.
4. Initialize the certificate authority.

   This step will create the certification and credentials for the Ceritifcate Authority, for the Hedeby processes (cs_vm, rp_vm and executor_vm) and for the Hedeby admin user.
5. Create the basic configuration for the master host (setup of three JVMs, setup of configuration service component, CA component, Resource Provider component, Spare Pool component and Executor component).

The **sdmadm inst_master_host** command has the following options:

Option	Description
`-l <local spool dir>`	Path of the local spool directory. This parameter is optional (default /var/spool/sdm/<system name>)
`-d <dist dir>`	Path of the distribution directory. This parameter is optional. The default value is derived from the path to the sdmadm command.
`-cs_port <port>`	Port where the config service should run
`-au <admin user>`	Name of the admin user
`-ca_country <country>`	Country name used for the certificate authority information (exact two letters)
`-ca_state <state>`	State name used for the certificate authority information
`-ca_location <location>`	Location token used for the certificate authority information

Option	Description
-ca_org <organisation>	Organization name used for the certificate authority information
-ca_org_unit <org unit>	Organization unit name used for the certificate authority information
-ca_admin_mail <admin email address>	Administrator mail address used for the certificate authority information
-sge_root <sge_root>	Hedeby uses the Grid Engine Certificate Authority utils. For the installation we need SGE_ROOT.
-autostart	This parameter is optional. It specifies whether the autostart feature should be installed on this host. For more information about autostart feature the section called “Autostart Feature in Hedeby”
-al	Automatically accept license. This parameter is optional. If the user uses this option, this means that he agrees with license. The license text will be just printed and installation will proceed.

Example 2.1. Example for a Master Host Installation

The "sdmadm inst_master" command is used to install the Hedeby master host on the local host "foo". After successfull execution the Hedeby system "hedeby1" has the bootstrap information stored in the system preferences. The CA component is using the specified "-ca_..." options for creating the CA certificate. The "cs_port" value specifies a free port on the master host used for the configuration service. All components will contact this service to get their configuration.

```
foo# mkdir /var/spool/sdm

foo# sdmadm -s hedeby1 -p system              \
          install_master_host                 \
          -ca_admin_mail "userFoo@foo.com"    \
          -ca_state "fooLand"                 \
          -ca_country "FO"                    \
          -ca_location "fooLocation"          \
          -ca_org_unit "fooUnit"              \
          -ca_org "fooOrg"                    \
          -a sdm_admin                        \
          -d /opt/sdm                         \
          -cs_port 31118                      \
          -l /var/spool/sdm/hedeby1           \
          -sge_root /opt/sge62
```

After installing the command **sdmadm startup_jvm** starts all processes on the master host

```
foo# sdmadm -s hedeby1 startup_jvm
jvm         host  result  message
--------------------------------
cs_vm       foo   STARTED
executor_vm foo   STARTED
rp_vm       foo   STARTED
```

Install Managed Hosts

After the the master host is has been installed and started the managed host installation must be performed. Go to each managed host and execute the **sdmadm inst_managed_host** command. The following steps will be executed:

1. Print out the software license and ask user to accept it.

2. Create the bootstrap configuration for the system. The bootstrap configuration for managed host should be always installed in the system preferences. (-p SYSTEM).

3. Create the necessary files and directories in the local spool directory. Please note the the parent directory of the local spool directory must exist. The user who performs the managed host installation must have write permissions into this directory.

4. Create security certificates and credentials.

   This step will create the certification and credentials the executor JVM and for the Hedeby admin user.

   For this step the **sdmadm inst_managed_host** command will contact the CA component on the master host. The user has to provide some credentials for authentication. Per default this command will prompt the username and password of the Hedeby admin user. For passwordless authentication it is possible to use the keystore of the admin user. This path to the keystore can be specified with the global option `-keystore`. The keystore of the admin user is created during the master host installation. It is stored in the local spool directory under `security/users/<admin username>.keystore`

   During the connection establishment the **sdmadm inst_managed_host** command will ask the user if the certificate of the is trusted. The question can be avoided if the certificate of the ca certificate is specified with the global `-cacert` option. The ca certificate is stored in the master hosts local spool directory under `security/ca/ca_top/cacert.pem`.

5. Create the basic configuration for the managed host (setup of the executor_vm and the executor component).

The following table shows all possible command options. Global command options are described in the section called “Global Cli Commands”.

Option	**Description**
`-l <local spool dir>`	Path of the local spool directory. This parameter is optional (default is `/var/spool/sdm/<system name>`)
`-d <dist dir>`	Path of the distribution directory. This parameter is optional. The default value is derived from the path to the sdmadm command.
`-au <admin user>`	Name of the admin user
`-cs_url <master host>:<cs_port>`	Name of the master host and cs port
`-autostart`	This parameter is optional. It specifies whether the autostart feature should be installed on this host. For more information about autostart feature the section called “Autostart Feature in Hedeby”

Example 2.2. Example Managed Host Installation

The **sdmadm inst_managed_host** command is used to make host foo2 to a managed host. After successfull execution the Hedeby system "hedeby1" has the bootstrap information stored in the system preferences. The command uses the keystore of the sdm_admin user for authentication against the configuration service.

```
foo1# mkdir /var/spool/sdm
foo1# sdmadm -s hedeby1 -p system                                          \
         --keystore /net/foo/export/home/sdm_admin/tmp/sdm_admin.keystore \
         --cacert /net/foo/export/home/sdm_admin/tmp/ca_cert.pem          \
         inst_managed_host                                                \
         -a sdm_admin                                                     \
         -d /opt/sdm                                                      \
         -l /var/spool/sdm/hedeby1                                        \
         -cs_url foo:31118
```

After installing the command **sdmadm start** is used to startup the Executor component on the managed host.

```
foo1# sdmadm -p system -s hedeby1 startup_jvm
jvm         host  result  message
---------------------------------
cs_vm       foo1  STARTED
```

Add Managed Services

The master host installation adds automatically the first service to the Hedeby system. It's the Spare Pool. After all managed hosts are installed additional services can be added. For now it is only Spare Pools and Grid Engine Services are possible.

Grid Engine Services are added with the **sdmadm add_ge_service** command. A detailed description of this command can be found at the section called “Add a Grid Engine Service”

Hedeby system administration

The Hedeby distribution contains the command line utility **sdmadm**. All administrative tasks can be executed with **sdmadm**.

Managing bootstrap configuration

Overview

The bootstrap configuration of Hedeby gives the **sdmadm** command the basic information about the available Hedeby systems.

Hedeby knows two different types of bootstrap configurations.

SYSTEM The SYSTEM boostrap configuration is the default location for installing a Hedeby system. On unix based system the SYSTEM bootstrap configurations is stored in the directory `/etc/sdm/bootstrap`. Normally only user root has write access to directory.

USER On unix based system the USER boostrap configurations are stored in the home directory in the sub directory `.sdm`. The user boostrap configurations can be used to define the bootstrap configuration for a Hedeby system on a host which is not part of Hedeby system. This is useful if an administrator want invoke **sdmadm** from his/her preferred workstation and not from a managed host (see also the section called “Add Bootstrap Config”). If the home directory is on a shared file system the user bootstrap is also shared.

Note

Windows™ is currently not supported by Hedeby, however it is on the TODO list. On Windows the boostrap configuration will be stored in the Registry.

The following information is stored in the boostrap configuration:

Target Hedeby System

There exists three different ways howto define the target system.

1. Specifying the system name with the global **-system** (shortcut -s) parameter when starting **sdmadm**.

```
% sdmadm -s test_system ...
```

2. Setting the environment variable SDM_SYSTEM. The following example uses bourne shell syntax:

```
% SDM_SYSTEM=test_system
% export SDM_SYSTEM
% sdmadm ...
```

3. Defining the a default system in the bootstrap configuration. This is done with the **sdmadm set_default_bootstrap_config command** (shortcut is sedbc).

```
% sdmadm -s test_system set_default_bootstrap_config
```

The **sdmadm set_default_bootstrap_config command** command stores the default system in the home directory of the user in the `.sdm` directory. This information is kept alive even if the command terminal is closed.

The **sdmadm show_bootstrap_config** command prints in the properties column the information what system is the default system. The following command shows that the system with name `test_system` is the default system.

```
% sdmadm show_bootstrap_config
system        type   host  port  properties version
-------------------------------------------------
test_system   SYSTEM foo   31016 default    1.0
prod_system   SYSTEM foo1  31118            1.0
```

If the user does not want a default system it can be reset with the **sdmadm unset_default_bootstrap_config**

The **sdmadm -system** overwrites the SDM_SYSTEM enviroment variable. SDM_SYSTEM overwrites the default system from the bootstrap config.

CS Contact Information

The central communication contact point of a Hedeby system is the host and port the JVM running the configuration service. Both values are stored in the bootstrap information.

Local Spool Directory

sdmadm and Hedeby JVMs has to known the path to the local spool directory. This directory contains e.g. trusted certificates and credentials for establishing a SSL connections. The local spool directory can be displayed with **sdmadm show_bootstrap_config -all**:

```
% sdmadm -s test_system show_bootstrap_config -all
system      type host  port  properties version
-----------------------------------------------
test_system SYSTEM foo   31016 default  1.0
    spool=/var/spool/sdm/test_system
...
```

The local spool directory of a single Hedeby system can have different path on different hosts.

Distribution Directory

The Hedeby bootstrap mechanism can start different versions of Hedeby with the same **sdmadm** command as long as systems have the same version the bootstrap configuration. To enable this feature the path to the distribution directory must be stored in the bootstrap configuration. **sdmadm** reads first the distribution directory from the bootstrap config and loads the libraries of the target system before invoking further actions.

The path to the distribution directory can be displayed with:

```
% sdmadm -s test_system show_bootstrap_config -all
system      type host  port  properties version
-----------------------------------------------
test_system SYSTEM foo   31016 default  1.0
    dist=/opt/sge
...
```

SSL Encryption

Per default the communication between the Hedeby component is encrypted with SSL. Turning of SSL encryption is dangerous and not recommended, however it is possible. In the bootstrap configuration the Hedeby system property no_ssl can be set to true. This is done with **sdmadm set_bootstrap_config_property**.

```
%sdmadm -s test_sytem -p user set_bootstrap_config_property \
        -p no_ssl -v true
System property "no_ssl" set to "true"
```

The **sdmadm sbc** print in the properties column no_ssl for systems where SSL is disabled.

```
% sdmadm sbc
system        type   host  port  properties     version
```

```
-----------------------------------------------------
test_sytem    USER   foo   31016 no_ssl,default 1.0
```

Boot Time Startup

Hedeby has boot time startup support. If the bootstrap configuration of a system has set the auto_start property it is started during boot time. The property can also set with **sdmadm set_bootstrap_config_property**.

```
%smadm -s ts31016user -p user set_bootstrap_config_property -p auto_start -v tr
System property "auto_start" set to "true"
```

The **sdmadm sbc** prints in the properties column auto_start for for system where auto_start property is set.

```
% sdmadm sbc
system        type   host  port  properties     version
-----------------------------------------------------
test_sytem    USER   foo   31016 auto_start,default 1.0
```

For more information about boot time startup please have a look at the section called “Autostart Feature in Hedeby”

Adding a System to the Bootstrap Configuration

The **sdmadm add_bootstrap_config** adds a Hedeby system to the boostrap configuration. The global option -s defines the system name, the -p switch defines used bootstrap configuration (SYSTEM or USER).

```
% sdmadm -s test_system -p user add_bootstrap_config -cs foo:31016
Using default local spool directory: /var/spool/sdm/test_system
Using default distribution directory: /opt/sdm
A configuration for system, "test_system", has been added
```

For more details please reference the cli command reference of **sdmadm add_bootstrap_config** (see the section called “Add Bootstrap Config”);

Once the system is defined in the boostrap configuration **sdmadm** can be used to invoke commands on the system. The **add_bootstrap_config** does not store any cretedentials for password less authentication to the local spool directory, **sdmadm** will report a "permission denied" error when contacting a Hedeby component.

```
% sdmadm sc
Error: permission denied
```

To enable username/password authentication the global option -ppw can be specified.

```
% sdmadm -ppw sc -t Other
username [sdm_admin] >
password >
host  jvm         component type  state
---------------------------------------------
foo   executor_vm ca        Other STARTED
      rp_vm       reporter  Other STARTED
```

For password less authentication the credentials of the user must be stored in form of a keystore in the local spool directory. This keystore can be obtained with the **sdmadm update_keystore**. The ca component must be active for the update_keystore command.

```
% sdmadm -ppw -s test_system update_keystore -t user -n sdm_admin
username [sdm_admin] >
password >
The specified keystore has been exported to the file
/var/spool/sdm/test_system/security/users/sdm_admin.keystore
```

Any further call of sdmadm will use this keystore for authenticating the user.

Removing a system from the boostrap configuration

If the bootstrap configuration of a system is not longer necessary it can be removed with **sdmadm remove_bootstrap_config**. The system can only be removed if no JVM of the system is running on the host.

```
 sdmadm -s test_system -p user remove_bootstrap_config
The configuration for system "test_system" has been removed
```

The **sdmadm** command does not delete the local spool directory.

JVMs and Components

Hedeby is a component based system. The known component types are Configuration Service, Resource Provider, Reporter, Service, CA, Executor. All these components runs inside of different java processes. These processes are called Java Virtual Machines (JVMs).

Managing JVMs

The default installation of Hedeby knows three different JVMs:

cs_vm — Runs the Configuration service. The cs_vm is the only process which must bind a static port. Only one instance of the cs_vm exists and this is running on the master host.

executor_vm — The executor_vm normally exists on any managed host (only with executor component) and also on the master host (runs the executor and the CA component).

The executor_vm must run as privileged user (on unix systems the user root) otherwise the executor component can not execute privileged action like starting command as different user or changing the owner ship of a file.

rp_vm — The rp_vm is the multi purpose JVM. It runs as non privileged user. After a default installation the rp_vm on the master host runs Resource Provider, Reporter and Spare Pool. GE Service component should run also inside the rp_vm on the qmaster of the Grid Engine installation.

Starting JVMs

Starting a Hedeby system means starting the JVMs of the system. Hedeby provides mechanism to start JVMs remotely. The administrator has to start **sdmadm startup_jvm** directly on the host.

If **sdmadm startup_jvm** without parameter is invoked it will start any not running JVM on the local host. If no component for a JVM is configured it is not started. To startup a JVM the CS component must be active (exception the cs_vm). **sdmadm startup_jvm** knows this dependy, if it is started on the master host it will first startup the cs_vm before starting any other JVM.

The following example shows a typical manual startup of a Hedeby system:

```
% rlogin <master_host>
% su -
password:
# sdmadm -s test_system startup_jvm
jvm         host            result  message
-------------------------------------------------------
cs_vm       <master_host>   STARTED
executor_vm <master_host>   STARTED
rp_vm       <master_host>   STARTED
#
```

In the next steps the administrator has to execute the **sdmadm startup_jvm** on each managed host.

```
% rlogin <managed_host>
% su -
# sdmadm -s test_system startup_jvm
jvm         host           result  message
-------------------------------------------------------
executor_vm <managed_host> STARTED
```

If the startup of a JVM failed the error is printed to the message column. So times the real error is written into the log file of the JVM. The log file is stored at `<local_spool_dir>/log/<jvm_name>-0.log`.

With the **sdmadm show_jvm** the state of the JVMs can be monitored.

```
% sdmadm -s test_system -p system set_default_bootstrap_config
```

```
Default system set to 'test_system'
% sdmadm sj
name        host            state      used_mem  max_mem   message
-------------------------------------------------------------------------------
cs_vm       <master_host>   STARTED           7M      113M
executor_vm <master_host>   STARTED           5M       28M
            <managed_host>  STARTED           5M       28M
...
rp_vm       <master_host>   STARTED           5M      113M
```

Stopping JVMs

The **sdmadm shutdown_jvm** stops JVMs. This command can perform the follwoing actions:

- Stop a single JVM
- Stop all JVMs with the same name on all hosts
- Stop all JVMs on a single host
- Stop all JVMs of a Hedeby system

For more details please have a look at the section called “Shutdown Jvm”

Managing Components

At JVM startup normally all components of are started automatically. **sdmadm startup_component** and **sdmadm shutdown_component** manages the lifecylce of a component. **sdmadm show_component** display the state of components. The following example shows the usage of these commands. For more information please have a look at the section called “Component Cli Commands”.

```
% sdmadm show_component
host  jvm         component         type             state
-------------------------------------------------------------------
foo   executor_vm ca                Other            STARTED
                  executor          Executor         STARTED
      rp_vm       reporter          Other            STARTED
                  resource_provider ResourceProvider STARTED
                  spare_pool        Service          STARTED
% sdmadm shutdown_component -c ca -h foo
comp host  message
----------------------------
ca   foo shutdown triggered
% sdmadm show_component -c ca
host  jvm         component type  state
-------------------------------------------
foo executor_vm ca        Other STOPPED
% sdmadm suc -c ca
comp host  message
---------------------------
ca   foo   startup triggered
```

Change the Configuration of a Component

The configuration of the components is stored in the configuration service. The **sdmadm mod_component** opens the configuration in an prefered text editor (EDITOR environment variable). The user can make the modifications. After saving the changes and closing the editor the modified configuration is validated and stored in the configuration service.

```
% sdmadm mod_component -c ca
-----------------------------
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<common:componentConfig xsi:type="security:CAComponentConfig"
                        caHost="foo"
                        adminUser="sdm_admin"
    <security:sgeCaScript>/opt/sge62/util/sgeCA/sge_ca</security:sgeCaScript>
</common:componentConfig>
-----------------------------
:x
Component configuration updated
```

The new configuration is now stored in the configuration service. However all instances of the component still uses the old configuration. The **sdmadm update_component** command triggers the component to reload the configuration from the configuration service.

```
% sdmadm update_component -c ca
comp host  message
--------------------------
ca   foo   reload triggered
```

Managing Services

If a service should be managed by Hedeby it is necessary to add the service to the system and start the component with service. Once a service is started it will be autodiscovered by Resource Provider and then the system starts to manage the service.

In order to be able to add a service to the system, the JVM with CS has to be run. If the Resource Provider is not running during the start up of service, this service is not managed by system. Resource Provider is the key component for managing services.

- The Resource Provider component must be configured (see the section called “Resource Provider”) and started (see the section called “Install Hedeby”). It is also recommended to update the policies for the service which should be registered at the Resource Provider (see Policy Engine Configuration).

Add a Spare Pool

The default installation has already defined a Spare Pool. Normally it is not necessary to add new spare pools. However with **sdmadm add_spare_pool_service** command it is possible. The admininstrator has to specify the name of the Spare Pool, the host where it should live and the name of the JVM. If the JVM is running the **add_spare_pool_service** will start the spare spool component if the **-start** option is specified.

```
% sdmadm add_spare_pool_service -n spare_pool1 -h foo -j rp_vm -start
service_name hostname jvm_name message
-------------------------------------
spare_pool1  foo      rp_vm    ADDED
```

The **sdmadm show_service** displays the new Spare Pool.

```
grawp% sdmadm show_service
host        service     cstate  sstate
-------------------------------------
master_host spare_pool  STARTED RUNNING
foo         spare_pool1 STARTED RUNNING
...
```

Add a Grid Engine Service

Before adding a Grid Engine Service to a Hedeby system the following requirements must be fulfilled

- The version of the Grid Engine instance must be 6.2 or higher.
- The Java Management Extension (JMX, new feature of Grid Engine 6.2) must be enabled in qmaster.

  For more information about enabling the JMX feature please have a look into the Grid Engine 6.2 Installation Guide.
- The Certitifcate Authority of the Grid Engine must be initialized and a java keystore for an Grid Engine admin user must exist.

  The Grid Engine Service will use this keystore for password less authentication against Grid Engine. For more information please have a look at the man page of Grid Engines **sge_ca** tool (man sge_ca).
- The host where the Grid Engine Adapter should run must have access on the Grid Engine Root directory (SGE_ROOT). This host must be a managed host of the Hedeby system. Normally the Grid Engine Adapter should run on the same host as Grid Engines qmaster process.

The Grid Engine service can be added with the **sdmadm add_ge_service**. For the correct configuration the following parameters are required:

- Host and JVM where the GE service component should run
- Path to SGE_ROOT
- Name of the Grid Engine cell
- Name of the Grid Engine admin user
- Path to the keystore of the Grid Engine admin user (`/var/sgeCA/port$SGE_QMASTER_PORT/$SGE_CELL/userkeys/<user_name>/keystore`)
- Password of the keystore if it is encrypted

- Port of the qmaster JMX server (defined in `$SGE_ROOT/$SGE_CELL/common/jmx/management.properties`
- Qmaster port
- Execd port
- Clustername of Grid Engine (defined in `$SGE_ROOT/$SGE_CELL/common/cluster_name`)

The **sdmadm add_ge_service** opens a template for a Grid Engine Service configuration in the editor. The Administrator has to editor the parameters. After saving the changes and closing the editor the GE Service component is created. For more details about the configuration of the Grid Engine Service please have a look at the section called “GEAdapter Configuration”.

If the JVM of the Grid Engine Service is already started the service component can be started by **sdmadm add_ge_service** (add the -start option). Otherwise the JVM has to be started after adding the service (**sdmadm startup_jvm**).

```
% sdmadm add_ge_service -n ge -h <qmaster_host> -j rp_vm -start
--- output of the editor -------------------------------------------------
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<common:componentConfig xsi:type="ge_adapter:GEServiceConfig"
                        mapping="default">
    ...
    <ge_adapter:connection
      keystore="/var/sgeCA/port31004/default/userkeys/sge_admin/keystore"
      password="changeit"
      username="sge_admin" jmxPort="31016"
      execdPort="31005" masterPort="31004"
      cell="default"
      root="/opt/sge62"
      clusterName="sge62"/>
     ...
</common:componentConfig>
------------------------------------------------------------------------
% sdmadm show_serivce
host         service     cstate  sstate
----------------------------------------
master_host  spare_pool  STARTED RUNNING
foo          spare_pool1 STARTED RUNNING
qmaster_host ge          STARTED RUNNING
...
```

The component state of the ge service (column cstate) should be STARTED. The service state (column sstate) will be RUNNING of all configuration parameter where correct and qmaster could be contacted. If the service state is not RUNNING please check the configuration parameters and reconfiguration the GE service component (**sdmadm mod_component -c ge**).

Starting and Stopping a Service

A service component has two states. The component state reflects the state of the component, while the service state reflects the state of the service. The service will only produce resource requests if it the service state is RUNNING. Assigning and Unassigning is also only possible if the service state is

RUNNING. The service state will automatically set to UNKNOWN if the service adapter detects that the communication to the service is interrupted (e.g in the case of a Grid Engine service if qmaster is not reachable).

To change the service state of a service manually the command **sdmadm startup_service/ shutdown_service** are used.

```
% sdmadm show_service
host  service    cstate  sstate
--------------------------------
foo spare_pool STARTED RUNNING
% sdmadm show_resource_request
type    service     slo     urgency
---------------------------------
pending spare_pool  default 10

% sdmadm shutdown_service -s spare_pool
service    result  message
-------------------------
spare_pool STOPPED
% sdmadm show_service
host  service    cstate  sstate
--------------------------------
foo spare_pool   STARTED STOPPED
% sdmadm srr
type    service     slo                 urgency
-----------------------------------------------
% sdmadm startup_service -s spare_pool
service    result  message
-------------------------
spare_pool STARTED

% sdmadm show_resource_request
type    service     slo     urgency
---------------------------------
pending spare_pool  default 10
```

Removing a Service

A service can be removed with **sdmadm remove_service**. This command can be used to remove any kind of service (Spare Pool and ge services). If the service has assigned resources they are also removed from the Hedeby system. A component can only be removed if it is not running.

```
% sdmadm show_serivce
host         service     cstate  sstate
---------------------------------------
master_host  spare_pool  STARTED RUNNING
foo          spare_pool1 STARTED RUNNING
qmaster_host ge          STARTED RUNNING
% sdmadm shutdown_component -c spare_pool1 -h foo
comp        host  message
-----------------------------------
```

```
spare_pool1 foo   shutdown triggered
% sdmadm remove_service -s spare_pool1
service     host  result
-------------------------
spare_pool1 foo   REMOVED
% sdmadm ss
host  service    cstate  sstate
--------------------------------
grawp spare_pool STARTED RUNNING
```

Managing Resources

Resource Definition Overview

Resource definition for the first use case

In order to manage resources in the Hedeby system it is necessary to define what a resource is. The current implementation only supports host resources. Resources are represented by an aggregation of properties. Each property is a name/value pair which represents some required attribute of the resource. When a resource is requested these properties are used for to find matching resources, a matching resource is one that fulfills each of the requested resource properties values.

All resources have some common resource properties:

- Resource ID: This is a unique resource identifier for resources of type "host". The resource id for a host resource is based on the host's fully qualified name (FQN).
- Type: Represents the type of a resource, such as "host" or "license" (currently only the type "host" is supported).
- State: Represents the current state of the resource, with respect to assignment. Valid states are "assigned" (to a service), "unassigned" (not assigned to a service), "assigning" (to a service), "unassigning" (from a service), "inprocess" (resource has been just assigned to RP and there is no info about resource state) and "error" (general resource problem).
- Annotation: The annotation is a human readable segment of text, assigned by the Resource Provider, which provides additional information about the resources state for the administrator, such as text describing why the resource is in an error state.
- Properties: Each host resource has a free definable set of name/value pairs defined in the properties list. Depending on the type of the resource some properties can be mandatory. The following example prints show the predefined properties of a host resource:

```
% sdmadm show_resource_types
name property                  flags type
-------------------------------------------
host ambiguous                 M     Boolean
     annotation                      String
     hardwareCpuArchitecture         String
     hardwareCpuCount                Integer
     hardwareCpuFrequency            String
     operatingSystemName             String
     operatingSystemPatchlevel       String
```

```
operatingSystemRelease          String
operatingSystemVendor           String
resourceHostname          M     Hostname
resourceIPAddress               String
static                    M     Boolean
usage                     M     Usage
```

All resource properties with the M flag are mandatory. All other properties are optional. The resource may have additional properties which are not defined in the type of the resource

Hedeby system is using the following property values which have to be defined for a host resources:

Table 2.1. Used host resource property names

Property	Description
resourceHostname	Resolvable name of the host.
resourceIPAddress	IP address of the host.
hardwareCpuArchitecture	Architecture type, e.g. "sparc64", "x86".
hardwareCpuCount	Number of CPUs
operatingSystemName	Operating system name, e.g. "solaris", "windows".
operatingSystemRelease	Operating system version number.

The following properties might be used in future releases:

Table 2.2. Reserved host resource property names

Property	Description
hardwareCpuFrequency	CPU clock speed in MHz.
operatingSystemPatchlevel	Operating system patch level.
operatingSystemVendor	Operating system vendor, e.g. "Sun Microsystems".

What can I do with a resource?

The current implementation can add host resources to the Spare Pool. When the hosts resource is added the Resource Provider itselfs will assign the hosts to the services among reported needs from the services and the specified Resource Provider policies. The administrator can assign and remove the host resources from a service or to a service manually. For a detailed information about resource operations see the section called “Managing Resources” and the section called “Resource”.

Adding a Resource to the System

sdmadm add_resource|ar -pf <resource_properties> This command will add a resource to the system. Its up to Resource Provider to which service it will be added. Resource will go through the decision process and will end up in a service if needed or will be rejected if there is no service that wants that resource. The -pf switch is required. It specify the file name for a resource properties file.

```
% sdmadm -s test ar -pf resource1.properties
```

Other optional switches: i - Id of the resource. It could be obtain from resource(generated). t - Type of the resource. You can specify type of the resource if not defined in resource properties file. s - Service id. Specify the service that you want resource to be assign to.

For more information about the **add_resource** command please have a look at the section called “Add resource to the system”

Removing a Resource from the System

sdmadm remove_resource|rr <resource_id> ... This command will remove a resource from the system. It is possible to specify just one resource or more at one time to remove. Specifying at least one is necessary.

```
% sdmadm -s test rr foo3.czech foo4.czech
```

For more information about the **remove_resource** command please have a look at the section called “Remove resource to the service”.

Moving an existing resource to a Service

Normally Resource Provider does automatic resource assignments according to the configured SLOs. With the **sdmadm move_resource** resources can be moved manually. The resource id and the target service must be specified.

```
% sdmadm -s test mvr -r foo3.czech -s spare_pool2
```

There is no guarantee that the resource is really moved the targeted service. It highly depends on the configured SLOs. It can also happen that after moving the resource to the target service Resource Provider moves this resource back to the orignal service.

For more information about the **remove_resource** command please have a look at the section called “Move resource to the service”.

How to Monitor the System

The current Hedeby system has the possibility to show the current available resources and services via command line interface. The components itself are logging information into log files in the Hedeby local spool directory on the local host.

Show Registered Services

The **sdmadm show_services** command is used to show all services which are managed by the Resource Provider component:

```
%  sdmadm show_services
host  service    cstate  sstate
-------------------------------
```

```
host1 spare_pool STARTED RUNNING
host2 ge         STARTED RUNNING
```

The listing shows that the Hedeby system is managing two services. Service `spare_spool` runns on host `host1`, service `ge` runs host host `host2`. The components representing representing the services are both running. The state of both services is `RUNNING`.

Show available Resources and their States

The **sdmadm show_resource|sr** command is used to show all resources and their states in Hedeby.

```
% sdmadm -s test sr
service      id     state   type
--------------------------------
ge           host1 ASSIGNED host
             host2 ASSIGNED host
spare_pool   host3 ASSIGNED host
```

The **sdmadm show_resource|sr -s <service_id>** command is used to show all resources and their states just for given service id.

```
% sdmadm -s test sr -s spare_pool -f
service    id     state    type
-------------------------------
spare_pool host3 ASSIGNED host
```

The **sdmadm show_resource|sr -r <resource_id>** command is used to show resource state and service that its already assigned to for given resource id.

```
% sdmadm -s test sr -r host1
service id    state    type
---------------------------
ge      host1 ASSIGNED host
```

The **sdmadm show_resource|sr -r <resource_id> -f** command is used to show resource state and service that its already assigned to a given Resource id. By specifying -f switch we will get full information about given Resource like, type, annotation and all its properties.

```
% sdmadm -s test sr -f
service      id     state    type annotation
-------------------------------------------
```

```
    ge           host1 ASSIGNED host
        resourceHostname=host1
        resourceIPAddress=192.168.0.101
                 host2 ASSIGNED host
        hardwareCpuArchitecture=amd64
        operatingSystemName=Solaris
        resourceHostname=host2
        resourceIPAddress=129.157.141.102
    spare_pool   host3 ASSIGNED host
        operatingSystemName=Solaris
        resourceHostname=host3
        resourceIPAddress=192.168.0.103
```

type - indicates the type of the resource(i.e. host). *annotation* - indicates the reason why are resource has reached this state. *properties* - indicates all properties that describe the resource. More information about the resource and it properties you can find here the section called “Resource definition for the first use case”

JVM Log Files

Each JVM of a SDM system writes log message into the subdirectory log in the local spool directory. The following files are stored there

- `<jvm name>-<sequence-number>.log` - SDM uses the Java logging module. All log messages of a jvm are written into these files. The jvm will perform a log file rotation.
- `<jvm name>.stdout` - All messages written to stdout are redirected into this file. This file should have the size 0.
- `<jvm name>.stderr` - All messages written to stderr are redirected into this file. This file should have the size 0.

The configuration of the Java logging module is stored in the local spool directory in the file `logging.properties`. Changes in these file will become active after the next restart of the jvm. For more information about the Java logging module please have a look at http://java.sun.com/j2se/1.5.0/docs/guide/logging/index.html [http://java.sun.com/j2se/1.5.0/docs/guide/logging/index.html]

Control log file rotation

With the default configuration the maximum size of a log file is 5MB, the maximum number of logging files is 4. This values can be adjusted by changing the following lines in logging.properties:

```
# approximate maximum amount to write (in bytes) to any one file. If this is ze
# then there is no limit. (Defaults to no limit).
# set the maximum amount per file to 5 MBs
#   (1 MB = 2^20 bytes = 1048576 bytes )
java.util.logging.FileHandler.limit= 5242880

# how many output files to cycle through (defaults to 1).
java.util.logging.FileHandler.count = 4
```

Control log levels

The Java logging module defines the following log levels:

- SEVERE (highest value)
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST (lowest value)

The global log level is defined in logging.properties. With the default installation the globel log level is set to INFO.

```
.level=INFO
```

The Java logging module provides different loggers for writing log messages. For each logger the logging level can be defined in logging.properties. SDM provides for each java class a separate logger. The name of the logger is the full qualified Java class of the class. It is possible to define the individial log levels on java package name level or java class level.

All java class of the SDM system are in a sub package of com.sun.grid.grm. The following configuration sets the log level of all SDM logger to FINE.

```
# set log level for the java package com.sun.grid.grm to FINE
com.sun.grid.grm.level=FINE
```

For debugging the resource provider and service it is useful to set the log level if the package com.sun.grid.grm.service to FINE.

```
# set log level for the java package com.sun.grid.grm.service to FINE
com.sun.grid.grm.service.level=FINE
```

Reporter Component

For more information about reporter component see the section called “Reporter” and the section called “Reporter”. Reporter component stores its files in "<local_spool>/spool/reporter" directory at startup. The listing below shows the files for the Reporter component while running:

```
% localSpool/spool/reporter ls
report-0.log
report-0.log.lck
...
```

The file contains data that are in the Arco database format. They aren't much readable for the user and they look like this:

```
1200415399278:NOTIFICATION:1:CS_OBJECT_ADDED:10:cs:Configuration object [resour
1200415399278:CONFIGURATION:1:resource_provider:active_component:Other
1200415399293:NOTIFICATION:2:SERVICE_ADDED:1:spare_pool:Service [spare_pool] ha
1200415399633:NOTIFICATION:3:CS_OBJECT_ADDED:11:cs:Configuration object [report
1200415399633:CONFIGURATION:3:reporter:active_component:Other
1200415425377:NOTIFICATION:4:CS_OBJECT_REMOVED:12:cs:Configuration object [spar
1200415425377:CONFIGURATION:4:spare_pool:active_component:com.sun.grid.grm.serv
1200415425382:NOTIFICATION:5:CS_OBJECT_REMOVED:13:cs:Configuration object [reso
1200415425382:CONFIGURATION:5:resource_provider:active_component:Other
....
```

so the better way to obtain the data is to use CLI commands.

The command to print out all the data from reporter is **sdmadm show_history | shist**. This command will print out all the data that are stored in the reporter files. The output will look like this:

```
time_stamp              type                          service_name      resource_
-------------------------------------------------------------------------------
16/01/2008 11:28:07.888 SERVICE_ADDED                 spare_pool
16/01/2008 11:28:08.073 CS_OBJECT_ADDED               cs
16/01/2008 11:28:08.303 CS_OBJECT_ADDED               cs
16/01/2008 11:28:34.855 RESOURCE_ADD                  resource_provider ge6@resou
16/01/2008 11:28:34.869 RESOURCE_ADDED                resource_provider ge6@resou
16/01/2008 11:28:34.878 RESOURCE_ADD                  resource_provider ge6@spare
16/01/2008 11:28:34.887 RESOURCE_ADDED                resource_provider ge6@spare
16/01/2008 11:28:35.092 RESOURCE_ADD                  spare_pool        ge6
16/01/2008 11:28:35.122 RESOURCE_ADDED                spare_pool        ge6
16/01/2008 11:28:35.129 RESOURCE_REQUEST              spare_pool
16/01/2008 11:28:35.135 RESOURCE_REMOVE               resource_provider ge6@resou
16/01/2008 11:28:35.136 RESOURCE_REMOVED              resource_provider ge6@resou
16/01/2008 11:28:35.172 RESOURCE_PROPERTIES_CHANGED   resource_provider ge6@spare
16/01/2008 11:28:35.173 RESOURCE_REMOVE               resource_provider ge6@spare
                        RESOURCE_REMOVED              resource_provider ge6@spare
16/01/2008 11:28:35.174 REQUEST_QUEUED                spare_pool
                        REQUEST_PROCESS               spare_pool
16/01/2008 11:28:35.175 REQUEST_PENDING               spare_pool
                        REQUEST_PROCESSED             spare_pool
16/01/2008 11:28:35.199 RESOURCE_PROPERTIES_CHANGED   spare_pool        ge6
16/01/2008 11:28:35.201 REQUEST_QUEUED                spare_pool
16/01/2008 11:28:35.202 REQUEST_PROCESS               spare_pool
16/01/2008 11:28:35.203 REQUEST_PENDING               spare_pool
16/01/2008 11:28:35.204 REQUEST_PROCESSED             spare_pool
16/01/2008 11:28:35.225 REQUEST_QUEUED                spare_pool
16/01/2008 11:28:35.226 REQUEST_PROCESS               spare_pool
16/01/2008 11:28:35.227 REQUEST_PENDING               spare_pool
                        REQUEST_PROCESSED             spare_pool

....
```

column *time_stamp* - show us the time_stamp when event occur in the system. column *type* - this is the type of the notification. column *service_name* - the service or component on which event occur

column *resource_id* - the resource id of resource that was involved in the event. column *description* - comprehensive description of what acctually happend in the system.

You can always use "grep" command to parse an output of the shist command, but there are some more convient ways to obtain needed data. By using shist command with its option you can filter the output as you want. More information about shist command you can find here: the section called “Show History”

Autostart Feature in Hedeby

Currently, the Hedeby support, for autostart during the bootstrap of the system, is done in two possible ways: using startup scripts or by leveraging from Solaris SMF (Service Management Facility) support. Autostart is supported only for systems that are installed using SYSTEM preferences. The autostart is host specific feature. This means that one host in the system can have autostart feature and the others do not need to have this feature installed. During the installation of the system (for master or managed host) flag -autostart needs to be specified to install autostart feature. If the current host supports SMF, the SMF support will be installed. Otherwise the autostart will be done using bootstrab scripts.

Support with Bootstrap Scripts

Currently, the scipts installed during installation are for all Hedeby systems on the host. This means that once the scripts are installed all Hedeby systems with flag autostart will be started during boot time of machine. Shutdown will also be triggered for those system when the machine is going down. The scripts support will be uninstalled once the last Hedeby system on current host is being uninstalled using **sdmadm** uninstalled command. **sdmadm uninstall_hosts** is described in

SMF Support

SMF support is installed when the autostart flag was set in installation command and current host support SMF. Once such installation is completed the services are automatically enabled. This means that after installation the system on current host is started.

The services are installed per one JVM. This means that each JVM. in the system, has one SMF service. To check the status of services the command: **svcs** should be used. After the installation the output of svcs command should be the same as in example below:

```
# svcs "*sdm*" STATE          STIME    FMRI
online         13:56:29 svc:/application/management/sdm/smfsystem/cs_vm:default
online         13:56:38 svc:/application/management/sdm/smfsystem/jvm:rp_vm
online         13:56:38 svc:/application/management/sdm/smfsystem/jvm:executor_
```

We can distinguish two types of Hedeby services, first one is service for CS component. There is only one such service in the whole system and it represents the JVM with Configuration Service (CS) component. The other services that reside on master host have dependency on CS_VM service. The SMF services, that are running on managed host, do not have dependecy on CS_VM as it is not possible to check whether the service on remote host is running.

Note: If the CS_VM service onmaster host is not online (cs_vm in system is not running), the start of SMF services on managed hosts will fail and the service will go into maintenance state. Once SMF service is in maintenence state it can be brought back online by using command **svcadm**

```
# svcadm enable svc:/application/management/sdm/smfsystem/cs_vm:default
# svcadm disable svc:/application/management/sdm/smfsystem/cs_vm:default
# svcadm clear svc:/application/management/sdm/smfsystem/cs_vm:default
```

In the example above there are three basic operations that can be done on SMF service using **svcadm**.

Table 2.3. Supported system properties

Command	Description
svcadm enable svc:/application/management/sdm/smfsystem/cs_vm:default	This command will try to start the SMF service. In this particular example it will try to start cs_vm. If the SMF service was already in "online" state, nothing will be done. If the service was in "disabled" state the cs_vm will be started. If the start operation will fail, the SMF service will go into maintenance state.
svcadm disable svc:/application/management/sdm/smfsystem/cs_vm:default	This command will try to stop the SMF service. In this particular example it will try to stop cs_vm. If the SMF service was already in "disabled" state, nothing will be done. If the service was in "online" state the cs_vm will be stopped. If the stop operation will fail, the SMF service will go into maintenance state. If the service was in the "maintenance" state, the state will be changed to "disabled"
svcadm clear svc:/application/management/sdm/smfsystem/cs_vm:default	This command will try to restart the SMF service. In this particular example it will try to restart cs_vm. This command takes affect only if service is in "maintenance" state. If the "clear" operation will finish successfully the SMF service will become "online".

Note: The SMF service can also be in temporary states (online/disabled) that means that this states are valid till the reboot of machine. This means if service is temaporary in "disabled" state, it will try to become online after the reboot of machine.

The usage of **sdmadm** can influence on SMF services. This means that if SMF service for rp_vm is online, and the administrator will shutdown the rp_vm using **sdmadm** command, the SMF will become temporary disabled (it will become available after the reboot of host). The example can be found in the following example.

```
# svcs "*sdm*"
STATE          STIME    FMRI
disabled       14:12:33 svc:/application/management/sdm/smfsystem/jvm:executor_
online         13:56:38 svc:/application/management/sdm/smfsystem/jvm:rp_vm
online         14:38:04 svc:/application/management/sdm/smfsystem/cs_vm:default
# sdmadm -s smfsystem sdj -j rp_vm
Shutdown has been triggered for following JVMs: [rp_vm]
# svcs "*sdm*"
STATE          STIME    FMRI
disabled       14:12:33 svc:/application/management/sdm/smfsystem/jvm:executor_
disabled       14:39:01 svc:/application/management/sdm/smfsystem/jvm:rp_vm
```

```
online         14:38:04 svc:/application/management/sdm/smfsystem/cs_vm:default
# svcs -lp svc:/application/management/sdm/smfsystem/jvm:rp_vm
fmri         svc:/application/management/sdm/smfsystem/jvm:rp_vm
name         Service Domain Manager - SDM
enabled      false (temporary)
state        disabled
next_state   none
state_time   Tue Jan 15 14:39:01 2008
logfile      /var/svc/log/application-management-sdm-smfsystem-jvm:rp_vm.log
restarter    svc:/system/svc/restarter:default
contract_id
dependency   require_all/none svc:/system/filesystem/local (online)
dependency   optional_all/none svc:/system/filesystem/autofs (online)
dependency   require_all/none svc:/milestone/network (online)
dependency   optional_all/none svc:/network/nfs/client (online)
dependency   require_all/none svc:/application/management/sdm/smfsystem/cs_vm:d
```

Naming explanation

The following table explains the naming structure for SMF service.

Table 2.4. Supported system properties

Prefix that should be the same for all Hedeby SMF services. It describes the type of service.	The name of application - Service Domain Manager	Name of the Hedeby system.	Type of JVM - for CS component JVM it is "CS_VM", otherwise it is "JVM"	Name of the instance - for CS component JVM it is "default", otherwise it is the name of JVM
svc:/application/management/	sdm/	smfsystem/	cs_vm:	default
svc:/application/management/	sdm/	smfsystem/	JVM:	rp_vm

Command for showing smf support

To see the installed SMF services for given system the **sdmadm ssmf** should be used. It prints the table with JVM_Names and corresponding SMF service names. If SMF is not supported on the host the proper message is printed that SMF is not supported.

```
jvm name    service name
------------------------------------------------------------------------
cs_vm       svc:/application/management/sdm/smfsystem/cs_vm:default
executor_vm svc:/application/management/sdm/smfsystem/jvm:executor_vm
rp_vm       svc:/application/management/sdm/smfsystem/jvm:rp_vm
```

How to manage security

TODO, needs cleanup

Cli Commands

Global Cli Commands

Global options can be used with all Cli commands in SDM. Those options are used for setting the general behaviour of the CLI.

Option	Shortcut	Description
`-debug`	`-d`	Turn on debugging information With the `-debug` option the Java platform stacktraces of all reported errors will be printed.
`-Debug`	`-D`	Start JVM in debug mode The `-Debug` Starts JVM in debug mode. The port will be printed to stdout. The JVM will be suspended until a JDPA Debugger is attached.
`-log`	`-l`	Set level for individual loggers This option modifies the loggers levels of the java.util.logging system. The argument of this option has the form [<logger name>=]<log level>(:[<logger name>=]<log level>)* **Examples** `-log FINE` Set the global log level to FINE `-log foo.bar=FINE` Set the level of the logger foo.bar to INFO `-log INFO:foo.bar=FINE` Set the global log level to `INFO` and the level of logger `foo.bar` to `FINE` Possible log levels are `SEVERE`, `WARNING`, `INFO`, `CONFIG`, `FINE`, `FINER`, `FINEST` and `ALL`
`-system`	`-s`	Specifies the system name. Specifies the name of the GRM system on which the command will be executed.
`-prefs`	`-p`	Use system or user preferences This option controls whether the system can be found in user or system preferences
`-cacert`	`-cc`	Specifies the path to the CA certificate file With this option it is possible to specify a different path to the CA certificate file (default is `<local_spool_dir>/security/ca/ca_top/ca_cert.pem`).
`-keystore`	`-k`	Specifies the path to the keystore file. With this option it is possible to overwrite the default path to the keystore file which is used for authentication (default is `<local_spool_dir>/security/users/<username>.keystore`).

Option	Shortcut	Description
-promptPW	-ppw	Prompt for password if no user keystore is available When there is no valid keystore available for the Hedeby user starting a cli command, the targeted JVM assumes that the user is in the "nobody" role. The "nobody" role has only the rights for reading the component configurations. If a user without valid keystore wants to connect to Hedeby this option is used to enable user/password prompt for the user to get authenticated for the Hedeby system.
-help	-hlp	Show global help output

Some of Cli commands are using table ouput for presenting the results of executed operation. This output can be customized by the following options:

Option	Example value	Description
-coldel <char>	|	Define delimiter character between two columns
-dupval	N/A	The normal table output does not display duplicate values. With this options it is possible to display the duplicate values.
-headerdel <char>	-	This character is used to print the delimiter line between the table header and the table body
-noheader	N/A	Switches off the printing of the header
-sort <columns>	n1,n2	comma separated list with the names of sorted columns

Cli commands to manage security in SDM

Managing the Admin User List

The Admin User List contains the name of the users which have admin privileges on the system. The following commands manages this list:

Adding the admin user

```
sdmadm add_admin_user|aau  <-au username>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-au <username>	Mandatory	Name of the user that should be added to the administator list.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

Adds an admin user to the admin user list

```
% sdmadm -s mySystem aau -au testuser
Admin user "testuser" added
```

Remove the admin user

```
sdmadm remove_admin_user|rau  <-au username>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-au <username>	Mandatory	Name of the user that should be removed from the administators list.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

Removes an admin user from the admin user list

```
% sdmadm -s mySystem rau -au testuser
Admin user "testuser" removed
```

Show the admin users

```
sdmadm show_admin_users|sau
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

Show the list of admin users

```
% sdmadm -s mySystem sau
admin_user
----------
testuser
root
```

Managing keystores and certificates

Following commands are used for creating/renewing the certificates for users and daemons (JVMs).

Add the certificate for the admin user

```
sdmadm add_admin_user_cert|aaucrt  <-au username> <-au username> <-cu user_comm
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-au <username>	Mandatory	Name of the user for which the certificate will be added.
-cn <user_common_name>	Mandatory	Common name for the user to be set.
-e <user_email>	Mandatory	Email address for the user.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

For passwordless authentication a user needs a valid certificate and private keys in the certificate authority of the sdm system. With this command the certificate and private key of a user is created. The update_keystore command can be used to distribute the keystore onto a remote host.

```
% sdmadm -s mySystem aaucrt -au testuser -cn user -e user@test.com
Private key and certificate for user, testuser, succesfully created
```

Note: As a result of execution of this command the certificate is generated. The certificate can be found in the following location: <sdm_local_spool_directory>/security/ca/ca_top/usercerts/<username>/ To install the generated certificate and keystore on host the sdmadm update_keystore command should be used the section called “Update keystores for the user or daemon on the host ”.

Add the certificate for the daemon

```
sdmadm add_daemon_cert|adcrt  <-j jvmname>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-j <jvmname>	Mandatory	Name of the JVM for which the certificate will be created.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command the certificate and private key of a daemon is created. The get_keystore command can be used to distribute the keystore onto a remote host.

```
% sdmadm -s mySystem adcrt -j r_vm
Private key and certificate for daemon, r_vm, succesfully created
```

Note: As a result of execution of this command the certificate is generated. The certificate can be found in the following location: <sdm_local_spool_directory>/security/ca/ca_local_top/daemons/<jvmname>/ To install the generated certificate and keystore on host the sdmadm update_keystore command should be used the section called “Update keystores for the user or daemon on the host ”.

Update keystores for the user or daemon on the host

This command updates keystores for users of SDM system or deamons (JVMs).

```
sdmadm update_keystore|uk  <-t type> <-n name> <-f file> <-p> <-hex>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-f file_path	Optional	The path to the keystore file ("-" prints to stdout).
-n name	Mandatory	Specifies the user's or daemon's name
-p	Optional	Prompt for passwords for the keystore.
-hex	Optional	Store the keystore as hex string.
-t type	Mandatory	Sets the certificate type (user|daemon).
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This commands asks the CA component of the grm system for a keystore of a user or a daemon. In the Hedeby system a user name is the unix name of an admin user and the deamon name is the name of a JVM. If the -f option is not specified user keystores are stored in <local_spool>/security/users/<username>.keystore and daemon keystores are stored in <local_spool>/security/<daemons>/<daemonname>.keystore.

```
% sdmadm -s mySystem uk -t daemon -n r_vm
The specified keystore has been exported to the file
/var/tmp/mySystem/security/daemons/r_vm.keystore
```

Renew Certificate

This command is used to renew certificate for a user or daemon for given number of days.

```
sdmadm renew_cert|rncrt  <-t type> <-n name> <-days days>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-n name	Mandatory/ Oprional	Sets the name of the user or daemon. It is mandatory when the type of certificate is for daemon or user. If the type is ca the name is ignored.
-t type	Mandatory	Sets the certificate type (user|daemon|ca).
-days num_days	Mandatory	Sets the number of days that certificate has to be valid for.

Option	Optional	Description
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

With this command a certifcate of a user, daemon or the CA can be renewed. In the Hedeby system a user name is the unix name of an admin user and the deamon name is the name of a JVM. If num_days is a negative number the certificate will be revoked.

```
% sdmadm -s mySystem rncrt -t daemon -n rp_vm -days 30
The certificate has been renewed
```

Show Certificate

This command is used to print certificate of a user or daemon.

```
sdmadm show_cert|scrt  <-t type> <-n name>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-n name	Mandatory	Sets the name of the user or daemon.
-t type	Mandatory	Sets the certificate type (user|daemon).
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

Show the certificate of a user or a daemon. In the Hedeby system a user name is the unix name of an admin user and the deamon name is the name of a JVM. Needs global option -system

```
% sdmadm -s mySystem scrt -t daemon -n rp_vm
[
[
  Version: V3
  Subject: EMAILADDRESS=root@ge5, UID=grm_daemon_root, CN=rp_vm, OU=as, O=as, L=
  Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4

  Key:  SunPKCS11-Solaris RSA public key, 1024 bits (id 16651232, session objec
  modulus: 13732573801935030993818250421726596787082414878387646834230440604752
  public exponent: 65537
  Validity: [From: Wed Jan 30 11:41:57 CET 2008,
              To: Fri Feb 29 11:41:57 CET 2008]
  Issuer: EMAILADDRESS=as, UID=CA, CN=SGE Certificate Authority, OU=as, O=as, L=
  SerialNumber: [    09]
```

```
Certificate Extensions: 4
[1]: ObjectId: 2.16.840.1.113730.1.13 Criticality=false
Extension unknown: DER encoded OCTET string =
0000: 04 1F 16 1D 4F 70 65 6E   53 53 4C 20 47 65 6E 65  ....OpenSSL Gene
0010: 72 61 74 65 64 20 43 65   72 74 69 66 69 63 61 74  rated Certificat
0020: 65                                                 e


[2]: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: C4 DB 35 CF F6 36 5B A4   1B A7 58 CB 58 35 7C 09  ..5..6[...X.X5..
0010: E0 8F 66 DC                                        ..f.
]
]

[3]: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: 6E D1 42 8A B7 19 B8 40   51 D5 64 45 AA DD 6C 60  n.B....@Q.dE..l`
0010: 8D 6F A9 39                                        .o.9
]

[EMAILADDRESS=as, UID=CA, CN=SGE Certificate Authority, OU=as, O=as, L=as, ST=a
SerialNumber: [    c17785b0 7557cad5]
]

[4]: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
CA:false
PathLen: undefined
]

]
  Algorithm: [MD5withRSA]
  Signature:
0000: 0C BF 6E F7 A1 C9 CB 84   5C 25 E9 DE 6F 0C 6D 97  ..n.....\%..o.m.
0010: 11 5C D0 1E 37 3B C3 EF   34 08 D6 FA 6C 2B 41 2F  .\..7;..4...l+A/
0020: 5C 55 16 FC 25 2D 4A 57   9D D5 BE 7D 78 5F 39 6F  \U..%-JW....x_9o
0030: 8F 70 FE E3 9A 2E 1B F9   BA 5C 70 73 AC 0C 47 3B  .p.......\ps..G;
0040: 4C 62 B6 44 FB A3 B7 D9   B4 75 DC 2B 12 FF C1 42  Lb.D.....u.+...B
0050: BE 36 3F BC 72 DB 08 6D   2B E0 D4 FE 0F 26 FE 9E  .6?.r..m+....&..
0060: D1 79 99 C0 26 DF 4D FF   86 54 F7 46 CD 3A A7 27  .y..&.M..T.F.:.'
0070: 8D F0 80 97 64 8C 41 B7   CD D2 55 94 49 DB A8 C7  ....d.A...U.I...

]
```

Update CA Certificate

This command is used to update CA certificate on local host.

```
sdmadm update_ca_cert|uccrt
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.

Option	Optional	Description
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

The update_ca_cert is useful of the CA certificate of a Hedeby system has been renewed. Normally during the installation of a managed host a copy of the CA certificate is stored in the local spool directory. If the CA certificate has been renewed this copy in not longer valid. This command gets the renewed certificate and stores in the local spool dir. The user is asked if the new certificate is trusted.

```
% sdmadm -s mySystem uccrt
CA certificate updated in /var/tmp/mySystem/security/ca/ca_top/cacert.pem
```

Monitoring Cli Commands

Show Blacklist

If the resource is not usable the service sends resource provider a message. The RP sets the state of the resource to UNASSIGNED. The RP stores in it's internal storage that the resource is not usable for this service (Blacklist).

Command **sdmadm show_blacklist** will print out the resources that, are not usable for services.

Usage:

```
smdadm show_blacklist | sb [-s service_name]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-s service_name]	Optional	With this option you can see the blacklist just for service, that you are interested in.

Command description:

This command allows to monitor resource ids that are ignored by a service(s). It can be specified whether the list of such resource ids is shown for a specified service or for each service service. If blacklist for specified service only should be shown, use -s option and specify service name.

Example:

```
% sdmadm show_blacklist
service        resource id
-------------------------
spare_pool     foo3
spare_pool2    foo5

% sdmadm sb -s spare_pool
service        resource id
-------------------------
```

```
spare_pool     foo3
```

Column Name	Description
service	Name of the service that has resource on blacklist.
resource id	Id of the resource that is on the blacklist for a service.

Show Bootstrap Configs

This command print out all the configuration, that are installed on the system. By configurations, we meant instances of SDM.

Usage:

```
smdadm show_bootstrap_configs | sbc [-auto] [-nossl] [-names] [-all]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-auto]	Optional	This option will print out only the system with the flag "auto_start" set to TRUE.
[-nossl]	Optional	This option will print out only the system with the disabled security.
[-names]	Optional	This option will print out only the system names.
[-all]	Optional	This option will print out all the information about the system alongs with the spool and distribution directory.

Command description:

The command will show the bootstrap configurations of the systems that are defined. The command is looking for bootstrap configurations defined in system and/or user preferences depending on the -prefs switch.

Example:

```
% sdmadm show_bootstrap_config -all
system type host port  properties version
----------------------------------------
some   USER foo3 31123            0.1
    spool=/var/spool/sdm/some
     dist=/opt/sdm
```

Column Name	Description
system	Name of the system.
host	Hostname where the CS component is located.
port	Port number of the CS Jvm.
properties	Displays the properties of the system. The following properties are defined.

Column Name	Description	
	`no_ssl`	If the ssl encryption is disabled for the system.
	`auto_start`	If the auto start flag for the system is set (system will be started at boottime).
	`smf`	If SMF support for the system is installed.
	`default`	If this system is the default system (the default system can be set with **sdmadm set_default_system**, see the section called “Managing bootstrap configuration”).
version	Version of the SDM system.	
spool(available with -all switch)	Path to the local spool directory on the host.	
dist(available with -all switch)	Path to the SGE dist directory.	

Show Components

This command print out all the components on JVM on the given hosts. The component type and actual state of the components is also printed.

Usage:

```
smdadm show_component | sc
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
-c component_name	Optional	Show only components with a name matching against component_name (can be a regular expression)
-h host_name	Optional	how only components running on a host with a name matching against host_name (can be a regular expression)
-j jvm_name	Optional	Show only components running in a JVM with a name matching against jvm_name (can be a regular expression)
-t type	Optional	Show only components of a specific type (valid types are ResourceProvider, Service, Executor or Other

Command description:

The command will show status of all system components. The output is grouped by system, host and JVM.

Example:

```
% sdmadm sc
host jvm         component         type              state
-------------------------------------------------------------
foo  executor_vm ca                Other             STARTED
                 executor          Executor          STARTED
```

```
      rp_vm       reporter          Other            STARTED
                  resource_provider ResourceProvider STARTED
                  spare_pool        Service          STARTED
```

Column Name	Description
host	host on which JVM and component is running.
jvm	Name of the Jvm.
component	Name of the component.
type	Type of the component.
state	State of the component.

Show Grid Engine Complex Mappings

GEAdpater automatically updates the properties of the assigned host resource. With the "Complex to Resource Property Mapping" the administrator can define what complex values are used. After the installation of the first Grid Engine Service the default mapping is installed. It can be displayed with the **sdmadm show_ge_complex_mapping** command.

Usage:

```
smdadm show_ge_complex_mapping | sgcm [-match match_name] [-output names|xml|ta
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-match match_name]	Optional	If this option is set only the matching mappings are printed
[-output names|xml|table]	Optional	Defines the output as xml, table or print just a name.

Command description:

This command prints out the available complex \ mapping for Grid Engine services

Example:

```
%  sdmadm show_ge_complex_mapping
name    complex complex value resource property       resource value
-------------------------------------------------------------------
default                       hardwareCpuArchitecture amd64


                                                      ia64
                                                      sparcv9
                                                      x86


                              operatingSystemName     Linux
```

```
                                                    Solaris

                                                    Windows
                              operatingSystemRelease  24

                                                    26
```

Column Name	Description
name	Name of the mapping.
complex	Name of the complex mapping.For example "arch".
complex value	Value of the complex mapping.For example for "arch" it could be "lx26-x86" value.
resource property	This is the SDM known property to which it will be mapped. For Example "hardwareCpuArchitecture".
resource value	This is the value of property. For Example for "hardwareCpuArchitecture" it will be "amd64".

Show Modules

This command print out all the components on JVM on the given hosts. The component type and actual state of the components is also printed.

Usage:

```
smdadm show_module | sm
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Command description:

Show all available modules in the system

Example:

```
% sdmadm show_module
module             version vendor
-----------------------------------------
gridengine-adapter 1.0     Sun Microsystems
security           1.0     Sun Microsystems
```

Column Name	Description
module	Module name.
version	Version of the module.

Column Name	Description
vendor	Vendor name of the module.

Show History

This command print out all the information that were stored by Reporter component. More information about Reporter component you can find here the section called “Reporter” or to the section called “Reporter”.

Usage:

```
smdadm show_history | shist [-sd date_string] [-ed date_string] [-r resource] [
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-r resource]	Optional	Print all messages related to this resource.
[-s service]	Optional	Print all messages related to this service.
[-t event_type]	Optional	Print all messages related to this event_type.
[-sd date_string]	Optional	Print all messages starting from this date. Possible formats: mm, HH:mm, dd HH:mm, MM/dd HH:mm, yyyy/MM/dd HH:mm.
[-ed date_string]	Optional	Print all messages till this date. Possible formats: mm, HH:mm, dd HH:mm, MM/dd HH:mm, yyyy/MM/dd HH:mm.
[-f advanced_filter]	Optional	Define special filter for reporter that could be compound with many row columns. Regular expressions can be used with this filter.

Command description:

Gets the data stored by reporter component. This command allows to monitor the system. You can specify the exact resource name or service name or just event type or combine them together. By specifing service name you will see all messages that were sent by this service. By specifing the resource name you will se all the messages related to this resource. You can also look for messages with the event type. Additionally you can define a reporter filter expression with the -f switch, regular expressions are allowed here. If you don't specify anything all messageses in the system will be printed. Using -sd (start date) and -ed (end date) switches you can specify exact date and get only the messages that were sent in the given time period. Example: sdmadm shist -sd 12:03 -ed 12:06 -f 'service=\"spare_pool\" && resource=\"foo\"' It will print out all the messages that are related to service with the name "spare_pool", resource with the name "foo" and the time period between 12:03 and 12:06 of present day.

Output example:

```
% sdmadm show_history
time_stamp              type                        service_name      resource_
-------------------------------------------------------------------------------
16/01/2008 11:28:07.888 SERVICE_ADDED               spare_pool
16/01/2008 11:28:08.073 CS_OBJECT_ADDED             cs
16/01/2008 11:28:08.303 CS_OBJECT_ADDED             cs
16/01/2008 11:28:34.855 RESOURCE_ADD                resource_provider foo@resou
16/01/2008 11:28:34.869 RESOURCE_ADDED              resource_provider foo@resou
16/01/2008 11:28:34.878 RESOURCE_ADD                resource_provider foo@spare_
16/01/2008 11:28:34.887 RESOURCE_ADDED              resource_provider foo@spare_
16/01/2008 11:28:35.092 RESOURCE_ADD                spare_pool        foo
```

```
16/01/2008 11:28:35.122 RESOURCE_ADDED              spare_pool        foo
16/01/2008 11:28:35.129 RESOURCE_REQUEST            spare_pool
16/01/2008 11:28:35.135 RESOURCE_REMOVE             resource_provider foo@resou
16/01/2008 11:28:35.136 RESOURCE_REMOVED            resource_provider foo@resou
16/01/2008 11:28:35.172 RESOURCE_PROPERTIES_CHANGED resource_provider foo@spare
16/01/2008 11:28:35.173 RESOURCE_REMOVE             resource_provider foo@spare
                        RESOURCE_REMOVED            resource_provider foo@spare
16/01/2008 11:28:35.174 REQUEST_QUEUED              spare_pool
                        REQUEST_PROCESS             spare_pool
16/01/2008 11:28:35.175 REQUEST_PENDING             spare_pool
                        REQUEST_PROCESSED           spare_pool
16/01/2008 11:28:35.199 RESOURCE_PROPERTIES_CHANGED spare_pool        foo
16/01/2008 11:28:35.201 REQUEST_QUEUED              spare_pool
16/01/2008 11:28:35.202 REQUEST_PROCESS             spare_pool
16/01/2008 11:28:35.203 REQUEST_PENDING             spare_pool
16/01/2008 11:28:35.204 REQUEST_PROCESSED           spare_pool
16/01/2008 11:28:35.225 REQUEST_QUEUED              spare_pool
16/01/2008 11:28:35.226 REQUEST_PROCESS             spare_pool
16/01/2008 11:28:35.227 REQUEST_PENDING             spare_pool
                        REQUEST_PROCESSED           spare_pool
```

Column Name	Description
time_stamp	Show us the time_stamp when event occur in the system.
type	This is the type of the notification.
service_name	The service or component on which event occur.
resource_id	The resource id of resource that was involved in the event.
description	Comprehensive description of what acctually happend in the system.

Show Jvms

This command print out all the JVM that are running in the system along with their status.

Usage:

```
smdadm show_jvm | sj [-j jvm_name] [-h host_name]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
-j jvm_name	Optional	Print only JVMs with a matching jvm_name. Can be a regular expression (for more information about regular expressions please have a look at Javadoc of package java.util.regexp [http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/package-summary.html]).
-j jvm_name	Optional	Print only JVMs running on host with a matching hostname. Can be a regular expression (for more information about regular expressions please have a look at Javadoc of package java.util.regexp [http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/package-summary.html]).

Command description:

Show runninng Jvms in Hedeby System. There could be Jvms which dont have running components and this command will show them.

Example:

```
% sdmadm show_jvm
name        host state
------------------------
cs_vm       foo  STARTED
executor_vm foo  STARTED
rp_vm       foo  STARTED
```

Column Name	Description
name	Name of the Jvm.
host	Host on which Jvm is running.
state	State of the Jvm.

Show Resources

This command print out all the resources that are recognizable by the system.

Usage:

```
smdadm show_resource | sr [-all] [-cached] [-r resource_id] [-s service] [-rf r
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-all]	Optional	It will print full description of resource, type, annotation and its properties.
[-cached]	Optional	It will print cached resource information.
[-r resource_id]	Optional	Id of the resource. Can be a java regular expression (for more information about regular expressions please have a look at Javadoc of package java.util.regexp [http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/package-summary.html]).
[-s service]	Optional	Name of the service that we want to print the state of resources from. Can be a regular expression
[-rf resource_filter]	Optional	Advanced filter for resources. Resource filter must be a valid filter expression (see also the section called “Filtering”). The command will only print out resources which matches this filter.
[-o res_ids|table]	Optional	Defines the output format (default is table). If the output format is res_ids the ids of all matching resource will be printed in a comma separated list. This list can be used as input for **reset_resource**, **move_resource** or **remove_resource**. If the output format is table the resource will be printed in the table format.
[-se]	Optional	Show service without resources. If this option is used services which do not have any assigned resource will be shown in the output. The resource values are set to "N/A".

Command description:

This command allows to monitor resource(s) state(s) in the system. You can specify the exact resource name that you want to check the status. By specifying service name you will see all resource states for the given service. If you don't specify anything all resources in the system will be printed. If you want to see full description of resource like type, annotation and its properties use -all option. If you want to see information about the cached resources, use -c option.

Example:

```
% sdmadm sr -all -rf 'hardwareCpuCount > 2'
service           id          state     type flags usage annotation
-------------------------------------------------------------------------------
resource_provider foo@spare33 INPROCESS host       1
    resourceHostname=foo
    hardwareCpuCount=4
    resourceIPAddress=xx.xx.xxx.xx
spare3            foo         ASSIGNED  host A     1     Resource is used by tw
    resourceHostname=foo
    hardwareCpuCount=4
    resourceIPAddress=xx.xx.xxx.xx
spare33           n/a         n/a       n/a  n/a   n/a   n/a
```

Move all not static resourcs of a service into spare_pool:

```
% sdmadm mvr -r `sdmadm sr -s sge1 -rf 'static = false' -o res_ids` -s spare_po
resource   message
-----------------------------------------------
foo        Resource move triggered
foo1       Resource move triggered
```

Column Name	Description
service	Name of the service.
id	Id of the resource.
state	State of the resource.
type	Type of the resource.
flags	Resource flags.
usage	Resource usage.
annotation	Resource annotations.
resource properties(available only with the -all switch)	Full description of the resource.

Show Resource Types

Shows information about all known resource types of a SDM system.

Usage:

```
show_resource_types | srt
```

The exit value of this command is always 0. The exit value may change in future (once the definition of resource types is stored in configuration service).

With SDM version 1.0 only the host resource type is supported. For future releases it is planned that users can modify the resource types.

Command description:

Shows all known resource types and its predefined resource property type in a table.

Example:

```
% sdmadm srt
name property                   flags type
-------------------------------------------
host ambiguous                  M     Boolean
     annotation                       String
     hardwareCpuArchitecture          String
     hardwareCpuCount                 Integer
     hardwareCpuFrequency             String
     operatingSystemName              String
     operatingSystemPatchlevel        String
     operatingSystemRelease           String
     operatingSystemVendor            String
     resourceHostname                 Hostname
     resourceIPAddress                String
     static                     M     Boolean
     usage                      M     Usage
```

Column	Description
name	Name of the resource type
property	Contains the names of the predefines properties for resources of this resource type.
flags	Flags of the properties. Currently only the M flag (mandatory) is supported. M means for each resource of this resource type this property must be defined.
type	Type of the resource property.

Type	Description
String	This is the default type. If the type of a resource property is not defined it is assumed that it is a string
Short	Integer value from -32768 to 32767
Integer	Integer value from -2147483648 to 2147483647
Long	Integer value from -9223372036854775808 to 9223372036854775807
Boolean	Boolean value (true or false)
Float	Single-precision 32-bit format IEEE 754 values
Double	Double-precision 64-bit format IEEE 754 values
Usage	The usage is a special integer value. It's range is limit from 0 to 100.

Type	Description
Hostname	A Hostname is a string. For comparing Hostnames a hostname resolving is done.

Show Resource Requests

Show registered resource requests at Resource Provider.

Usage:

```
smdadm show_resource_request | srr [-all] [-pp] [-pq] [-s service_name] [-slo s
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-all]	Optional	It will print full description of resource (detailed needs).
[-pq]	Optional	Enables printing of the queued requests.
[-pp]	Optional	Enables printing of the pending requests.
[-s service_name]	Optional	Print only requests for matching services (Can be a regular expression)
[-slo slo_name]	Optional	Print only requests for matching SLOs (Can be a regular expression)

Command description:

This command allows to monitor request(s) that are registered by Resource Provider. It can be specified whether the queued (waiting to be processed) or pending (already processed, waiting to be queued). If no type filter (or both of them) are specfied, requests of both types are printed. If full description of request type is needed, use -all option.

Example:

```
%  sdmadm srr -all
type    service  slo  urgency quantity
-------------------------------------
pending spare55  null 49      1
    annotation=
    type=host
    usage=1
    ambiguous=false
    static=false
        spare557 null 49      1
    annotation=
    type=host
    usage=1
    ambiguous=false
    static=false
```

Column Name	Description
type	Type of the resource request.
service	Service that has request.

Column Name	Description
slo	Name of the SLO.
urgency	Urgency of that request.
quantity	The number of requested resources.
resource properties(available only with the -all switch)	Full description of the resource.

Show Services

This command print out all the services registered in the system along with their component and service status.

Usage:

```
smdadm show_service | ss
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Command description:

The command will show services managed by system.

Example:

```
% sdmadm show_service
host service    cstate  sstate
------------------------------
foo  spare3     STARTED RUNNING
     spare33    STARTED RUNNING
     spare55    STARTED RUNNING
     spare557   STARTED RUNNING
     spare_pool STARTED RUNNING
```

Column Name	Description
host	Hostname where the service is running.
service	Name of the service.
cstate	State of the service component.
sstate	State of the service.

Show SLOs

This command print out all the current slos that services have with their decription. Right now we have following SLO:

- *Fixed Usage SLO* - This SLO is usable for all kind of services. It gives each resource of the service a fixed usage. It will never produce any Need.
- *Minimum Resources SLO* - This SLO count the number of assigned resource which matches a resource filter. If the number of assigned resources falls below the minimum a need is produced.

The usage map of this SLO contains min assigned resource. They all have as usage the urgency of the SLO.

- *Permanent Request SLO* - This SLO is never satisfied with resources that matches a provided resource filter. Need is produced everytime an update is called. The usage map of this SLO contains min assigned resource. They all have as usage the urgency of the SLO.

Usage:

```
smdadm show_slo | sslo [-s service_name] [-u]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-s service_name]	Optional	Print slo`s just for given service name (can be a regular expression).
[-u]	Optional	Print the resource usage for the SLO.
-r resource_name	Optional	Print only the usage of those resources matching the resource_name(This parameter has only effects of -u is specified).

Command description:

The command print information about SLOs of services.

Example:

```
% sdmadm -s hhe3 show_slo
service    slo                 quantity urgency request
-------------------------------------------------------------------------------
spare_pool PermanentRequestSLO 1        1       host->[]
```

Column Name	Description
service	Name of the service.
slo	Name of the slo.
quantity	How many resources are needed.
urgency	What is the urgency of this request.
request	What is the service need.

Show SMF Support

Show all installed smf services for this system. Command will print error message if SMF is not supported on host.

Usage:

```
smdadm show_smf_support | ssmf
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Command description:

Show all installed smf services for this system. Command will print message if SMF is not supported on host.

Example:

```
% sdmadm ssmf
jvm name    service name
--------------------------------------------------------------------
cs_vm       svc:/application/management/sdm/smfnew/cs_vm:default
executor_vm svc:/application/management/sdm/smfnew/jvm:executor_vm
rp_vm       svc:/application/management/sdm/smfnew/jvm:rp_vm
```

Column Name	Description
jvm name	Name of the JVM.
service name	Name of the smf service that correspond.

Administration Cli Commands

Add Bootstrap Config

This command will add a bootstrap config of the new the system. This configuration can be stored on the host which is neither master nor managed host. Security doesnt have to be bootstraped. User can use user/password authentication.

Usage:

```
smdadm add_bootstrap_config | abc -cs host:port [-l local_spool] [-d dist_dir]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
-cs host:port	Mandatory	With this mandatory parameter configuration service data are set. You have to specify hostname and port on which CS will be running.
[-d dist_dir]	Optional	Specify the path to Grid Engine dist directory.
[-l local_spool]	Optional	Specify the path to local_spool directory. When not used default directory will be set.

Command description:

This command adds the entries for a Hedeby system bootstrap configuration into the user or system preferences. This command requires the global -s and -p options.

Example:

```
% sdmadm -p USER -s test abc -cs foo:45321
Using default local spool directory: /var/spool/sdm/test
Using default distribution directory: /net/foo.foo/SGE_ROOT
```

```
A configuration for system, "test", has been added
```

Add Grid Engine Complex Mapping

With this command you can add your own resource mappings to adjust them to SDM properties. After executing this command VI editor will show up with the xml template of the file, You can adjust it to your needs.

Usage:

```
smdadm add_ge_complex_mapping | agcm [-t mapping_name] [-f file_name] -m mappin
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-t mapping_name]	Optional	Name of the mapping which should be used as template.
[-f file_name]	Optional	File name of the mapping file.
-m mapping name	Mandatory	Name of the new mapping.

Command description:

Grid Engine services needs a mapping for building the resource properties of autodiscovered exec daemons. With this command a new mapping for complex values can be added.

Example:

```
% sdmadm -s test agcm new_mapping
```

Install Managed Host

With this command you can install managed host. CAUTION to install it properly you must specify security global option. Keystore and cacert or -ppw to be prompt for user/password authentication.

Usage:

```
smdadm (-keystore keystore_file -cacert cacert_file | -ppw)
                    install_managed_host|imgdh
                    [-autostart] [-nossl] -au admin_user -cs_url cs_url
                    [-l local_spool_dir] [-d dist_dir]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-autostart]	Optional	Determine wheter system should be in autostart mode using RC scripts.
[-nossl]	Optional	Determine wheter system should have security disabled or not.
-au admin_user	Mandatory	Name of admin user.
-cs_url cs_ur	Mandatory	Url to the configuration service component. The proper pattern is host:port.
[-d dist_dir]	Optional	Path to the dist directory.

Option	isOptional	Description
[-l local_spool]	Optional	Path to the local spool directory.
-keystore | -k keystore_file	Mandatory	This is global option switch that has to be used with that command along with cacert switch. Its a path to the keystore file located in <local_spool>/security/users/user.keystore. As alternative you can use -ppw switch which will prompt user for password.
-cacert | -cc cacert_file	Mandatory	This is global option switch that has to be used with that command along with keystore switch. Its a path to the cacert file located in <local_spool>/security/ca/ca_top/cacert.pem. As alternative you can use -ppw switch which will prompt user for password.
-promptPW | -ppw	Mandatory	This is global option switch that has to be used with that command. User will be prompted for password. As alternative you can use using of keystore and cacert.

Command description:

This command is used to install the Hedeby managed host components on the local host.

Example:

```
% sdmadm -k <username.keystore path> -cc <cacert.pem path> -s test -p USER imstl
```

Install Master Host

With this command you can install master host. With the master installation there will be 3 JVMs installed: cs_vm which is JVM with the Configuration Service running on it, executor_vm with the executor component on it and rp_vm with the Resource Provider, Reporter and Spare Pool component on it.

Usage:

```
install_master_host|imsth [-autostart] [-nossl] -au admin_user
-ca_admin_mail mail_adress -ca_country country -ca_location loca
-ca_org_unit unit -ca_state state -cs_port port [-l local_spool_
[-d dist_dir] [-al]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-autostart]	Optional	Determine wheter system should be in autostart mode using RC scripts.
[-nossl]	Optional	Determine wheter system should have security disabled or not.
-au admin_user	Mandatory	Name of admin user.
-ca_admin_mail mail_adress	Mandatory	Mail of the admin user for security reasons.
-ca_country country	Mandatory	Country character (just 2 letters) of admin user for security reasons.

Option	isOptional	Description
-ca_location location	Mandatory	Location of admin user for security reasons.
-ca_org organization	Mandatory	Organization of the admin user for security reasons.
-ca_org_unit unit	Mandatory	Unit of the admin user for security reasons.
-ca_state state	Mandatory	State of the admin user for security reasons.
[-d dist_dir]	Optional	Path to the Grid Engine dist directory.
[-l local_spool]	Optional	Path to the local spool directory.
[-al]	Optional	Automatically accept license. This parameter is optional. If the user uses this option, this means that he agrees with license. The license text will be just printed and installation will proceed.
-sge_root sge_root	Mandatory	Path to the Grid Engine sge root directory.
-cs_port port	Mandatory	Port of the configuration service.

Command description:

This command is used to install the Hedeby master host components on the local host.

Example:

```
% sdmadm -s test -p USER imsth -au johndoe
-ca_admin_mail johndoe@foo.com -ca_country US -ca_location Nevada -ca_org SUN
-ca_org_unit GE -ca_state ok -cs_port 23456 -sge_root <SGE_ROOT_DIR>
```

Modify Component

This command allow modyfing of the component configuration. After executing this command VI editor will show up with the xml template of the file, You can adjust it to your needs.

Usage:

```
mod_component|mc -c component_name
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
-c component	Mandatory	Component name that we want to edit configuration of.

Command description:

This command opens the Component configuration in an editor.

Example:

```
% sdmadm -s test mc -c reporter
```

Modify Grid Engine Complex Mapping

This command allow modyfing of the ge complex mapping. After executing this command VI editor will show up with the xml template of the file, You can adjust it to your needs.

Usage:

```
mod_ge_complex_mapping|mgcm -m mapping_name
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
-m mapping_name	Mandatory	Mapping name that we want to edit ge complex mappping of.

Command description:

Grid Engine services needs a mapping for building the resource properties of autodiscovered exec daemons. With this command a existing mapping for complex values can be modified.

Example:

```
% sdmadm -s test mgcm -m default
```

Modify Global Config

This command allow modyfing of the system global configuration. After executing this command VI editor will show up with the xml template of the file, You can adjust it to your needs.

Usage:

```
mod_global_config | mgc
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Command description:

This command opens the global configuration in an editor.

Example:

```
% sdmadm -s test mgc
```

Remove Bootstrap Config

This command will remove bootstrap config for the specified system.

Usage:

```
remove_bootstrap_config | rbc
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Command description:

This command removes a system bootstrap configuration from the java preferences of a system or a user. This command requires the global option -system.

Example:

```
% sdmadm -p USER -s test rbc
```

Remove Grid Engine Complex Mapping

This command will remove Grid Engine Complex mapping.

Usage:

```
remove_ge_complex_mapping | rgcm -m mapping_name
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
-m mapping_name	Mandatory	Name of the mapping that we want to remove.

Command description:

Grid Engine services needs a mapping for building the resource properties of autodiscovered exec daemons. With this command a existing mapping for complex values can be removed.

Example:

```
% sdmadm -s test rgcm -m test_mapping
Complex mapping test_mapping removed
```

Set Property Of The Bootstrap Config

This command will set the property of system bootstrap config.

Usage:

```
set_bootstrap_config_property | sebcp -p property -v value
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Option	isOptional	Description
-p property	Mandatory	Name of the property that we want to set. Supported are auto_start and no_ssl
-v value	Mandatory	Value of the properties, right now the only possible value are TRUE or FALSE.

Command description:

Supported properties are "auto_start" and "no_ssl". If "auto_start" is set to "true", the system will be start at boot time. If "no_ssl" is set to "true" ssl encryption and authentication of incoming request is disabled. Disabling security should only be done for test systems, hence every user with execute permissions on sdmadm can modify the system.

Example:

```
% sdmadm -s test sebcp -p no_ssl -v TRUE
```

Set Default Bootstrap Config

This command will set specified system as default. Then when -s will not be specify this system will be taken.

Usage:

```
set_default_bootstrap_config | sedbc
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Command description:

If a default bootstrap system configuration is set, the sdmadm command can be used without the -system options. All actions will be executed on the default system.

Example:

```
% sdmadm -p USER -s test sedbc
```

Unistall Host

This command will unistall host in case of SDM system.

Usage:

```
uninstall_host  | uh [-forced] [-master]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-force]	Optional	The forced option specifies that the host will be removed from the system even if it can not contact running CS component.
[-master]	Optional	The master option is for uninstallation of Hedeby master host.

Command description:

Uninstall host from system, if the current host is a Master host, the uninstallation will cause that system will not be available anymore. If current host is managed host and it does not have running JVMs, the uninstallation will remove from configuration all components that are running on this host. Preferences and localspool directory will be cleaned up. To remove host even when CS can not be contacted use option -forced.

Example:

```
% sdmadm -s test uh -forced -master
Uninstallation of master host: foo finished successfully.
```

UnSet Default Bootstrap Config

This command will unset specified system as default.

Usage:

```
unset_default_bootstrap_config | usdbc
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Command description:

After unsetting the default system bootstrap configuration **sdmadm** needs the -system option to find a system. This command does not delete the system which is marked as default system. Use remove_system command to delete a system bootstrap configuration.

Example:

```
% sdmadm -p USER -s test usdbc
```

Component Cli Commands

Shutdown Component

This command will shutdown the component on specific host or all components on all hosts when -all switch is specify. Hostname, component name or -all has to be specified.

Usage:

```
smdadm shutdown_component | sdc [-c component] [-force] [-h host] [-all]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-c component]	Optional	Name of the component to shutdown. If no component name specify, system will try to stop all the components with -all switch or specific component with -h switch.
[-h host]	Optional	Name of the host where is the component(s) to shutdown. If no host name specify, system will try to stop all the components with -all switch or specific component with -c switch.
[-force]	Optional	The action that force parameter will execute depends on the implementation of the component. Check component description to see what happen if you use this flag.
[-all]	Optional	When specify it will shutdown all components on host if specify or all components with the same name on all host with -c switch. If no host and no component name specified it will shutdown all components on all hosts.

Command description:

The command will stop the specified component on a specified host. If only the "component" is not specified, the command will stop all components that are configured for the specified host. If only the "host" is not specified, the command will stop the specified component on all hosts where it should be running. If both the "component" and the "host" are not specified, the command will stop all components on all hosts where they should be running.

Example:

```
% sdmadm sdc -c spare3
comp   host message
------------------------------
spare3 foo  shutdown triggered
```

Column Name	Description
comp	Name of the component.
host	Hostname where component was located.
message	message describing the action that was taken.

Shutdown Jvm

This command will shutdown the JVM on specific host or all JVMs on all hosts when -all switch is specify. Hostname, jvm_name or -all has to be specify.

Usage:

```
smdadm shutdown_jvm | sdj [-j jvm_name] [-h host] [-all]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.

Option	isOptional	Description
[-j jvm_name]	Optional	Name of the JVM to shutdown. If no JVM name specified, system will try to stop all the JVMs with the -all switch or specific JVM with -h switch.To stop all JVM with the same name on all hosts JVM name and -all switch has to be specify.
[-h host]	Optional	Name of the host where is the component(s) to shutdown. If no host name specified, system will try to stop all the JVMs with -all switch or specific JVM with -j switch.To stop all JVM with on same hostname, hostname and -all switch has to be specify.
[-all]	Optional	It will shutdown all components on host if specified or all components on all host withe the given name if host not specify. Will shutdown all JVMs on all hosts when used alone without jvmname and hostname.

Command description:

The command will stop the specified JVM on a specified host. If only the JVM is not specified, the command will stop all JVMs that are configured for the specified host. If only the "host" is not specified, the command will stop the specified JVM on all hosts where it is running. If both the JVM and the "host" are not specified, the command will stop all JVMs on all hosts where are they running.

Example:

```
% sdmadm sdj -j executor_vm
jvm         host result  message
--------------------------------
executor_vm foo  STOPPED
```

Column Name	Description
JVM	Name of the JVM.
host	Hostname where JVM was located.
result	result of the action that was taken.
message	message describing the action that was taken.

Startup Component

This command will startup the component on specific host or on localhost if no host was specified.

Usage:

```
smdadm startup_component | suc [-c component] [-h host]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-c component]	Optional	Name of the component to startup. If no component name specify, system will try to start all the components.
[-h host]	Optional	Name of the host where is the component(s) to startup. If no host name specify, system will try to start all the components.

Command description:

The command will start the specified component on a specified host. If only the "component" is not specified, the command will start all components that are configured for the specified host. If only the "host" is not specified, the command will start the specified component on all hosts where it should be running. If both the "component" and the "host" are not specified, the command will start all components on all hosts where they should be running.

Example:

```
%  sdmadm suc -c spare3
comp   host message
----------------------------
spare3 foo  startup triggered
```

Column Name	Description
comp	Name of the component.
host	Hostname where component was located.
message	message describing the action that was taken.

Startup Jvm

This command will startup the JVM on localhost host.

Usage:

```
smdadm startup_jvm | suj [-j jvm_name] [-force] [-dp port]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-j jvm_name]	Optional	Name of the JVM to startup. If no JVM name specify, system will try to start all the JVMs on localhost.
[-dp port]	Optional	Start JVM in debug mode. The valid port number has to be bigger than 1024 and lower than 65535..
[-force]	Optional	With this option you can force start up of the JVM(s) even if the pid files exist. This command can be used after JVMs crashed and pid files are still in run directory. Remember you have to be sure that Hedeby java processes arent existing any more.

Command description:

The command will start the specified JVM on a localhost. If JVM is not specified, the command will start all JVMs that are configured for the localhost. By specifying the -d parameter and as argument port number after JVM name you can run it in debug mode. Examples: sdmadm -s <system_name> start starts all JVMs on the localhost sdmadm -s <system_name> start -f starts all JVMs on the localhost and force the start despite if pid file exists sdmadm -s <system_name> start -j JVM1 starts JVM1 on the localhost sdmadm -s <system_name> start -j JVM1 -d 33445 starts JVM1 on the localhost in debug mode on port 33445

Example:

```
% sdmadm suj -j executor_vm
jvm         host result  message
```

```
--------------------------------
executor_vm foo  STARTED
```

Column Name	Description
JVM	Name of the JVM.
host	Hostname where JVM will be located.
result	result of the action that was taken.
message	message describing the action that was taken.

Update Component

This command will try to reload configuration of the component on specific host or on localhost if no host was specified.

Usage:

```
smdadm update_component | uc [-c component] [-force] [-h host]
```

Option	isOptional	Description
-help/ -hlp	Optional	Print usage of this command.
[-c component]	Optional	Name of the component to reload. If no component name specify, system will try to reload all the components.
[-h host]	Optional	Name of the host where is the component(s) to reload. If no host name specify, system will try to reload all the components.
[-force]	Optional	The action that force parameter will execute depends on the implementation of the component. Check component description to see what happen if you use this flag.

Command description:

The command will reload the specified component configuration on a specified host. If only the "component" is not specified, the command will reload configuration of all components that are configured for the specified host. If only the "host" is not specified, the command will reload configuration of the specified component on all hosts where it should be running. If both the "component" and the "host" are not specified, the command will reload configuration of all components on all hosts where they should be running.

Example:

```
%  sdmadm update_component -c executor
comp     host message
-----------------------------
executor foo  reload triggered
```

Column Name	Description
comp	Name of the component.
host	Hostname where component was located.
message	message describing the action that was taken.

Resource Cli Commands

Add resource to the system

```
sdmadm add_resource|ar  <-t resource_type> <-f prop_file> <-r resourceId> <-s s
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-f <prop_file>	Optional, can not be used together with -r	File containing the definition of the resource properties. If prop_file is "-" the resource properties are read from stdin.
-s <service_name>	Optional	The name of the service to which the resource should be assigned. If no service is specified it's up to the Resource Provider to which service the resource is initialy assigned.
-t <resource_type>	Optional	The type of the resource. Currently only host value is valid, so host is also the default value for this option.
-r <resourceId>	Optional, can not be used together with -f	Id for the added resource. If not specified the Id will be generated.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

If the -f parameter is specified the complete definintion of the resources is read from the rpFile. It is a regular property file (<name=value>) that describes the characteristics of a resource. A line starting with === marks the begin of a new resource.

```
% cat ./foo
resourceHostname=foo
hardwareCpuArchitecture=amd64
hardwareCpuCount=1
hardwareCpuFrequency=2600
operatingSystemName=Linux
operatingSystemRelease=7.04
operatingSystemVendor=Ubuntu
===
resourceHostname=foo1
hardwareCpuArchitecture=amd64
hardwareCpuCount=2
hardwareCpuFrequency=2800
operatingSystemName=Solaris
operatingSystemRelease=10u4
operatingSystemVendor=Sun Microsystems
```

```
% sdmadm -s mySystem ar -f ./foo
resource    message
-------------------------------------------------------------------------
foo         Resource was added to the system.
foo1        Resource was added to the system.
```

Without -f parameter the resource is constructed out of the resource type definition. If a resource id is specified (-r parameter) all resource properties which can be derived from the resource id are automatically set. The following example shows how to add a host resource to the system. The host resource type needs only the resourceHostname. This property can be constructed out of the resource id, in this case no editor is started:

```
% sdmadm -s mySystem ar -type host -r foo
resource   message
-----------------------------------------
foo        Resource was added to the system.
```

If there are missing mandatory paramater the resource definition is opened in an editor. The following example adds a new host resource to the system. The -r parameter is not specified, so the editor is started.

```
% sdmadm -s mySystem ar
#
# Default values for host resources
#
resourceHostname = <Hostname, mandatory>❶
static = false
# hardwareCpuArchitecture = <String, optional>❷
# hardwareCpuCount = <Integer, optional>
# hardwareCpuFrequency = <String, optional>
# operatingSystemName = <String, optional>
# operatingSystemPatchlevel = <String, optional>
# operatingSystemRelease = <String, optional>
# operatingSystemVendor = <String, optional>
# resourceIPAddress = <String, optional>
```

❶ The user must specify all mandatory parameters. In the case of a host resource only the resourceHostname is necessary. The static properties is set by default to false.

❷ The editor shows also all optional predefined resource properties , they are commented out. User can also specify these properties.

After storing the changes and closing the editor the add_resource command validates the resource properties and adds the resource to the system.

```
resource   message
```

```
-----------------------------------------
foo        Resource was added to the system.
```

Here is description of the output

Column	Example value	Description
resource	foo	The Id of the resource on which operation was performed.
message	Resource was added to the system.	String describing the result of performed operation.

Move resource to the service

```
sdmadm move_resource|mvr  <-s service_name> <-r resourceId>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-s <service_name>	Mandatory	The name of the service to which the resources should be moved.
-r <resourceId1,resourceId2,...>	Mandatory	List of resource ids.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command allows to move resources to a specific service. The resource ids are needed as well as target service name is required. There is no guarantee that the resource will be moved from one service to antoher.

```
% sdmadm -s mySystem mvr -r foo,foo1 -s spare_pool
    resource    message
--------------------------------------
foo             Resource move triggered
foo1            Resource move triggered
```

Here is description of the output

Column	Example value	Description
resource	foo	The Id of the resource on which operation was performed.
message	Resource was moved.	String describing the result of performed operation.

Reset resource to the service

```
sdmadm reset_resource|rsr <-r resourceId1,resourceId2,...> -s <service>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-r <resourceId1,resourceId2,...>	Mandatory	Ids of the resources that should be reset. The list seperated with ",".
-s <service>	Optional	Name of the target service (owner of the resource).
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command allows to reset a resource or a list of resources. It is possible to specify just one resource id or more at once. Specifying at least one resource id is necessary. If ambiguous resource has to be reset, service name has to be specified.

```
% sdmadm -s mySystem rsr -r foo
resource                       message
-------------------------------------------------------------------------------
foo The resource was reset.
```

Here is description of the output

Column	Example value	Description
resource	foo	The Id of the resource on which operation was performed.
message	The resource was reset.	String describing the result of performed operation.

Remove resource to the service

```
sdmadm remove_resource|rr <-r resourceId1,resourceId2,...> <-s service_name> <-
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-r <resourceId>	Mandatory	Ids of the resources that should be removed. List of resources should use "," as separator character.
-s <service_name>	Optional	Name of the service from which the resource should be removed. It should be used when ambiguous resources are in the system.

Option	Optional	Description
-force	Optional	Boolean flag signaling whether the resource should be removed from its current service by force.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command allows to remove resource or resources from the system. It is possible to specify just one resource or more at one time to remove. Specifying at least one is necessary. If multiple resource are specified to be removed, the resource ids should be separated with coma sign and there should not be any white spaces between elements.

```
% sdmadm -s mySystem rr -r foo
resource                       message
-------------------------------------------------------------------------------
foo The resource was removed from the system.
```

Here is description of the output

Column	Example value	Description
resource	foo	The Id of the resource on which operation was performed.
message	The resource was removed from the system.	String describing the result of performed operation.

Modify resource

```
sdmadm mod_resource|mr <-r resourceId> <-f file>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-r <resourceId>	Mandatory	Ids of the resources that should be removed.
-f <file>	Optinal	The file which contains modified resource properties. If this parameter is specified the command run in the non interactive mode. Without -f option the add_resource command get the resource properties of the resource from the resource provider and opens them in the editor. The

Option	Optional	Description
		modified resource properties are sent back to Resource Provider.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command allows user to modify an existing resource. The two required parameters are a file name of resource.properties file and a resource id. The resource.properties file is a regular property file (<name=value>) that specifies the properties of a resource that has to be changed.

```
% sdmadm -s mySystem mr -r foo -f ./foo
resource                        message
-------------------------------------------------------------------------------
foo The resource was modified.
```

Here is description of the output

Column	Example value	Description
resource	foo	The Id of the resource on which operation was performed.
message	The resource was modified.	String describing the result of performed operation.

Add resources to the black list of the service

The resources which ids are on the black list of the service will not be accepted by this service.

```
sdmadm add_resource_to_blacklist|artb <-s service_name> <-r resourceId1,resourc
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-r <resourceId1,resourceId2,...>	Mandatory	Ids of the resources that should be added to black list of the service. List of resources should use "," as separator character.
-s <service_name>	Mandatory	The name of the service which black list will be modified.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command allows to add resource or resources (their ids) to a service's resource blacklist. It is possible to specify just one or more resources at once. Specifying at least one is necessary. The service name is mandatory.

```
% sdmadm -s mySystem artb -s spare_pool -r foo
resource                       message
-------------------------------------------------------------------------------

foo The resource was added to the spare_pool's
                               blacklist.
```

Here is description of the output

Column	Example value	Description
resource	foo	The Id of the resource on which operation was performed.
message	The resource was modified.	String describing the result of performed operation.

Remove resource from blacklist of the service

The resources which ids are on the black list of the service will not be accepted by this service.

```
sdmadm remove_resource_from_blacklist|rrfb <-s service_name> <-r resourceIds>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-r <resourceId1,resourceId2,...>	Mandatory	Ids of the resources that should be removed from black list of the service. List of resources should use "," as separator character.
-s <service_name>	Mandatory	The name of the service which black list will be modified.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command allows to remove resource or resources (their ids) from a service's resource blacklist. It is possible to specify just one or more resources at once. Specifying at least one is necessary. The service name is mandatory.

```
% sdmadm -s mySystem rrfb -s spare_pool -r foo
```

```
resource                        message
-------------------------------------------------------------------------------

foo The resource was removed from the spare_pool's
                                blacklist.
```

Here is description of the output

Column	Example value	Description
resource	foo	The Id of the resource on which operation was performed.
message	The resource was modified.	String describing the result of performed operation.

Cli commands for SDM services

Managing the lifecycle of the service

Startup of the service

```
sdmadm startup_service|sus  <-s service_name>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-s <service_name>	Mandatory	Name of the service that should be started.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

The command will start service that is represented by component, to start the service the component has to be already started

```
% sdmadm -s mySystem sus -s sp1
service result  message
-----------------------
sp1     STARTED
```

Here is descriptiopn of the output

Column	Example value	Description
service	sp1	The name of the service on which the start operation was performed.

Column	Example value	Description
result	STARTED	String describing the result of starting operation.
message	Error details	Error description or other meaningful messages that notify user about expected or unexpected operations that were performed.

Shutdown the service

```
sdmadm shutdown_service|sds  <-s servicename> <-fr>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-s <service_name>	Mandatory	Name of the service that should be stopped.
-fr	Optional	This flag specifies if the service should return all non-static resources to the system so other services will be able to use these resources once this one is down. If this flag is not specified the service will not free the resources.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

The command will stop service that is represented by component, it stops service not the component

```
% sdmadm -s mySystem sds -s sp1
service result  message
-----------------------
sp1     STOPPED
```

Here is descriptiopn of the output

Column	Example value	Description
service	sp1	The name of the service on which the stop operation was performed.
result	STOPPED	String describing the result of stopping.
message	Error details	Error description or other meaningful messages that notify user about expected or unexpected operations that were performed.

Add the services to the system

Currently there are supported two types of services in the system: spare_pool and ge services. These services can be added to system with the following commands.

Add Spare Pool Service

```
sdmadm add_spare_pool_service|asps  <-s service_name> <-h host_name> <-j jvm_na
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-s <service_name>	Mandatory	Name of the service that should be added.
-h <host_name>	Mandatory	Name of the host on which the service should.
-j <jvm_name>	Mandatory	Name of the JVM to which the service should be added.
-u <urgency>	Optional	Usage level to be set for service SLO. By default the value is set to 1.
-start	Optional	This flag specifies whether the service should be added to the already running JVM or whether the service should be available after the restart of JVM. The error will occur when this flag is specified and the JVM for the service is not running.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

The command will add Spare Pool to system, once this command is finished the Spare Pool can be started.

```
% sdmadm -s mySystem asps -s sp2 -h foo -j rp_vm -u 15 -start
service_name hostname jvm_name message
-------------------------------------
sp2          foo      rp_vm    ADDED
```

Column Name	Description
service_name	Name of the service that has been added.
hostname	Name of the host on which the service should run.
jvm_name	Name of JVM to which service was added.
message	Result message describing the outcome of command.

Add GE service

```
sdmadm add_ge_service|ags  <-s service_name> <-h host_name> <-j jvm_name> <-f f
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-s <service_name>	Mandatory	Name of the service that should be added.
-h <host_name>	Mandatory	Name of the host on which the service should.
-j <jvm_name>	Mandatory	Name of the JVM to which the service should be added.
-f <file>	Optional	Path to the file with configuration of service that should be imported. If this switch is not used the configuration template will appear.
-start	Optional	This flag specifies whether the service should be added to the already running JVM or whether the service should be available

Option	Optional	Description
		after the restart of JVM. The error will occur when this flag is specified and the JVM for the service is not running.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

This command adds entry to global configuration, creates GE service configuration and then it allows to start the service

Example for adding the GE service with a template:

```
% sdmadm -s mySystem ags -j rp_vm -h foo -s ge -start
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<common:componentConfig xmlns:common="http://hedeby.sunsource.net/hedeby-common
                        xmlns:resource_provider="http://hedeby.sunsource.net/he
                        xmlns:security="http://hedeby.sunsource.net/hedeby-secu
                        xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gr
                        xmlns:executor="http://hedeby.sunsource.net/hedeby-exec
                        xmlns:reporter="http://hedeby.sunsource.net/hedeby-repo
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xsi:type="ge_adapter:GEServiceConfig"
                        mapping="default">
    <common:slos>
        <common:slo xsi:type="common:FixedUsageSLOConfig"
                    urgency="50"
                    name="fixed_usage"/>
    </common:slos>
    <ge_adapter:connection keystore="/var/tmp/keystore"
                           password="password&amp;"
                           username="username"
                           jmxPort="35646"
                           execdPort="35645"
                           masterPort="35644"
                           cell="default"
                           root="/ws/rm199614/test/root"
                           clusterName="p35644"/>
    <ge_adapter:sloUpdateInterval unit="minutes"
                                  value="5"/>
    <ge_adapter:execd adminUsername="rm199614"
                      defaultDomain=""
                      ignoreFQDN="true"
                      rcScript="false"
                      adminHost="true"
                      submitHost="false"
                      cleanupDefault="true"/>
</common:componentConfig>
```

And the result of the command will be like:

```
Grid Engine service ge added
```

Here is explanation of some important fields from template presented above. For more information please refer to description of configuration of Grid Engine adapter.

Parameter	Example value	Description
keystore	"/var/tmp/keystore"	The path for user keystore used to authenticate user when connecting to Grid Engine. It is not mandatory, if the path is empty the username/password authentication is used.
password	"mypassword"	The user password used to authenticate user when connecting to Grid Engine. It is not mandatory, this is used when the keystore contains empty path.
username	"testUser"	The name of admin user used for connecting to Grid Engine
jmxPort	"23456"	Port number on which Grid Engine JMX MBean server is running.
execdPort	"23455"	Execution daemon port for Grid Engine.
masterPort	"23454"	Qmaster port for Grid Engine.
cell	"default"	Name of the cell of Grid Engine.
root	"/net/host/sgeroot"	SGE_ROOT path for Grid Engine
clusterName	"p23454"	Cluster name of Grid Engine.
adminUsername	"testUser"	Name of user used for installation of execution deamons by Grid Engine adapter.

Remove Service

```
sdmadm remove_service|rs <-s service_name> <-force>
```

Option	Optional	Description
-help/ -hlp	Optional	Print usage of this command.
-s <service_name>	Mandatory	Name of the service that should be removed.
-force	Optional	Specify this flag if the running service should be removed.
[global_options]	Optional	Global options that can be specify before the command. More info you can find here the section called “Global Cli Commands”

Command description:

The command will remove service from system, to remove running component (with service being STOPPED) -force option has to be specified.

And the result of the command will be like:

```
% sdmadm -s mySystem rs -s spare_pool -force
service    host result
-----------------------
spare_pool foo  REMOVED
```

Column Name	Description
service_name	Name of the service that has been removed.
hostname	Name of the host on which the service was running.
result	Result message describing the outcome of command.

Hedeby system configuration

Basics

Preferences

There are two ways how Hedeby can be installed. We call this ways preferences. Preferneces can be set for particular user - in that case we speak about USER preferences or for whole system platform - SYSTEM preferences and this installation has to be performed by superuser. The use of SYSTEM preferences makes available such features like: autostart/smf support for hosts within the Hedeby system.

The Hedeby is using its own implementation of java preferences. SYSTEM preferences are located in */etc/sdm/bootstrap/<system_name>*. USER preferences are located in *<USER_HOME>/.sdm/bootstrap/<system_name>*. More about java preferences you can find here [http://java.sun.com/javase/6/docs/technotes/guides/preferences/index.html]

The files structure for preferences looks like:

```
<system_name> --\
                |
                |-- hosts --\
                |           |-- <host_name> --\
                |           |                 |-- smf --\
                |           |                 |         |
                |           |                 |         \----prefs.properties
                |           |                 |
                |           |                 \----prefs.properties
                |           |
                |           \-- prefs.properties
                |
                \-- prefs.properties
```

The exemple content of prefs.properties file in <system_name> directory, it is the main bootstrap information about system:

```
version=0.1
localspool=/var/spool/sdm/localspool/
csInfo=foo\:2324
smf=true
ssl_disable=false
dist=/net/foo/sdm_dist
auto_start=false
```

Table 2.5. Description of the file content:

Parameter	Value	Description
auto_start	true/false	This optional parameter defines if Hedeby system has to be started during the machine boot up process. **WARNING**: It works only when Hedeby is installed with the SYSTEM preferences.
version	float value	Parameter that specifies the version of system preferences.
smf	true/false	This optional parameter defines if Hedeby system is installed with SMF support feature.
cs_info	String	This parameter defines the location of CS component of Hedeby system. The format is host_name:port.
dist	String (Path to the Hedeby dist directory)	More information you can find here the section called “Dist directory”
localSpool	String (Path to the Hedeby local spool directory)	More information you can find here the section called “Local Spool directory”

The exemple content of prefs.properties file in <system_name>/host/<host_name> directory, it is the host specific information about system:

```
localspool=/var/spool/sdm/localspool/
master=false
dist=/net/foo/sdm_dist
```

Table 2.6. Description of the file prefs.properties content. File located in <system_name>/host/<host_name>

Parameter	Value	Description
master	true/false	This parameter defines this host is master host or managed host.
dist	String (Path to the Hedeby dist directory)	Optional parameter if not defined the value will be taken from the main prefs.properties file
localSpool	String (Path to the Hedeby local spool directory)	Optional parameter if not defined the value will be taken from the main prefs.properties file

The exemple content of prefs.properties file in smf directory, that contains SMF information about system:

```
rp_vm=svc\:/application/management/sdm/mySystem/jvm\:rp_vm
```

```
executor_vm=svc\:/application/management/sdm/mySystem/jvm\:executor_vm
```

Table 2.7. Description of the file prefs.properties content. File located in <system_name>/host/<host_name>/smf

Parameter	Value	Description
<jvm_name>	String (smf service_name for JVM)	This pairs defines which JVM was installed with SMF support and which SMF service name is assigned to JVM. More information about SMF can be found here the section called “SMF Support”

Directories

Dist directory

This is directory with the Hedeby system installation files. You can find here binaries, installation scripts, libraries, manuals and state files that are used by Installer.

```
sdm_dist --\
           |
           |-- bin --\
           |         \-- sdmadm
           |
           |-- util --\
           |          |-- arch
           |          |-- arch_variables
           |          |-- sdmsmf.sh
           |          |-- smf_sdmsvc
           |          |-- supportRc.sh
           |          |-- templates --\
           |                          |-- sdm.env.template
           |                          |-- jaas.config.template
           |                          |-- java.policy.template
           |                          |-- sdmsvc
           |                          |-- sdm_template.xml
           |                          |-- sdm_template_masterhost.xml
           |                          |-- start_sh.template
           |                          |-- logging.properties.template
           |                          \-- ge-adapter --\
           |                                           |-- install_execd.conf
           |                                           |-- install_execd.sh
           |                                           |-- uninstall_execd.conf
           |                                           |-- uninstall_execd.sh
           |
           |-- lib --\
           |         |-- ext --\
           |         |         |-- endorsed --\
           |         |         |              \-- *.jar
           |         |         \-- *.jar
           |         |
           |         |-- <PLATFORM_ARCHITECTURE> --\
           |         |                             \-- libplatform.so
           |         |-- sdm-common.jar
           |         |-- sdm-starter.jar
```

```
|         |-- sdm-ge-adapter.jar
|         |-- sdm-ge-adapter-impl.jar
|         |-- sdm-security.jar
|         \-- sdm-security-impl.jar
|
\-- man --\
          \-- man1 --\
                     |-- sdmadm.1
```

Table 2.8. Description of Dist directory content:

File/Directory	Description
bin	This directory contents an executables for Hedeby system.
sdmadm	Command line util for administrating Hedeby. More info you can find here the section called “Hedeby system administration”
util/templates	This directory contains templates (necessary for installation). and arch script which is used to detect the system architecture.
util	This directory contains utility scripts used by Hedeby.
util/arch	Script for detecting architecture for machine on which the Hedeby is run.
util/arch_variables	Script containing definitions of variables for different architectures supported by Hedeby.
util/supportRc.sh	Script used by Hedeby for installation/ uninstallation of RC scripts support .
util/sdmsmf	Script used by Hedeby for installation/ uninstallation of SMF support .
util/smf_sdmsvc	Script used by Hedeby SMF support to manage lifecycle of Hedeby system.
man	This directory contents manual for **sdmadm**.
lib	This directory contains libraries for Hedeby system.

Local Spool directory

Local spool directory has to be specify on the local file system and for each managed host. The localspool directory can be different for each host. In local spool directory information about running componant are stored, but only this which JVMs are on that host, there are also logs from JVMs. This is also place for spooling local data and for security infos like certificates and keystores.

```
localSpool --\
             |-- log --\
             |         \-- log files
             |
             |-- run --\
             |         \-- files with the pids of running components
             |             on local host
             |
             |-- security --\
             |              |-- ca --\
```

```
|              |        \ (files for Grid Engine CA)
|              |
|              |-- deamons --\
|              |             \-- keystores for JVM`s
|              |
|              \-- users --\
|                          \-- keystores for Hedeby users
|
|-- spool --\
|           \-- cs    spool directory for configuration
|           \-- ...   spool directories for Hedeby components
|
|-- logging.properties
|
|
\-- tmp --\
          \-- tmp directories for Hedeby components
```

Table 2.9. Description of Local spool directory content:

File/Directory	Description
log	This directory is for keeping the log files of JVMs which are running on that host. the section called “How to Monitor the System”
run	This directory is used for keeping the prints of running JVMs on this host. Inside the file that coresponding with JVM name you can find a PID number of procss on which this component is running.
security	This directory contains security informations like certificates or keystores of components and Hedeby users. You can find more herethe section called “Security”
spool	This directory is used by the Hedeby components as persistent data storage. It contains the complete configuration of the Hedeby System (only in the host where the configuration service is running)
tmp	This directory is used by the Hedeby components as temporary data storage
logging.properties	This file is used to store logging settings for Hedeby system. More info about monitoring Hedeby system you can find here the section called “How to Monitor the System”

Java Virtual Machines

Overview (new)

The typical starting point for the Hedeby system configuration is to define the java virtual machines where the components should run. This also includes to define on which host a component should run.

When a Hedeby component should be started on a local host (Which might be done by **sdmadm start** command) the global configuration is used to find out which components have to be started in which virtual java machines on the local host.

One important task in the JVM configuration is to define the environment variables for the JVM. The current configuration supports the setting the LD_LIBRARY_PATH environment variables. All components startet in a JVM will have this defined environment setting.

The global configuration file is loaded/stored by the configuration service. It is stored in the local spool directory of the host where CS is running (<local spool>/spool/cs/global.xml).

The configuration file is written in XML format. The structure of the file is defined in the xml schema `hedeby-common.xsd`.

Definition of JVMs

Example 2.3. JVM configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<global ...>
    <jvm❶ name="root_jvm"❷ user="root"❸ port="0"❹>
        <component ...>❺
        </component>
        ...
        <jvmArg>-Dfoo.bar.prop=aaa</jvmArg>❻
        <jvmArg>-Xmx512M</jvmArg>

        <ldLibraryPath>❼
            <pathelement>/opt/sge/lib/${ARCH}</pathelement>
        </ldLibraryPath>
    </jvm>
    ...
</global>
```

❶ Each Hedeby system defines a set of Java virtual machines (JVMS). Each JVM is defined in a `<jvm>` tag.

❷ Each JVM needs a name. The name has to be unique in the Hedeby system.

❸ The `user` attribute specifies the name of the process owner of the JVM. Hedeby will try to start the JVM under this user account. This is only possible if a priviliged user starts Hedeby (on unix system user root).

❹ Each JVM hosts a JMX server. The `port` attribute specifies the port where incoming request are accepted. If the value is 0 the port will be dynamically allocated.

❺ Hedeby components (see the section called “Definition of Components”) are defined in the a `<component>` of each JVM.

❻ With `<jvmArg>` tags it is possible to pass additional startup parameter to the JVM.

❼ The `<ldLibraryPath>` allows the definition of the `LD_LIBRARY_PATH` for the JVM in a platform independed way.

Definition of Components

Inside the JVM configuration it is possible to define the components for this JVM. We have two different types of components:

Component types

Singleton	Only one instance of `Singleton` can exit in a Hedeby system. The configuration of a singleton only allows one host.
MultiComponent	One each host of the Hedeby one instance of this component can exist. The configuration defines a pattern for the hosts.

Example 2.4. Component configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<global ...>
    <jvm ...>
        <component xsi:type="MultiComponent"❶
                   name="executor"❷
                   classname="com.sun.grid.grm....ExecutorImpl"❸>
            <hosts>❹
                <include>.*</include>
                <exclude>foo.bar</exclude>
            </hosts>
            <config>executor</config>❺
        </component>
        <component xsi:type="Singleton"❻
                   name="ge_service"
                   classname="com.sun.grid.grm....GEServiceImpl"
                   host="ge_master"❼>
            <config>ge_service</config>
        </component>
        ...
    </jvm>
    ...
</global>
```

❶ This component is a MultiComponent. On each of the hosts of a Hedeby system one instance of the component can exit.

❷ Each component has a system wide unique name.

❸ The `classname` attribute specifies the name of the java class which implements the component. This class must be available in the classpath of the JVMS.

❹ Not each component will be started on each host. The `<hosts>` tag defines the hosts where the component will be started. It is possible to include and exclude hosts. The value of the `<include>` and `<exclude>` tag can be a java regular expression. The hosts tag is only allowed for MultiComponents.

❺ Define the path to the configuration of the component With the parameter the component can load the configuration from the config service.

❼ The component ge_service is a Singleton. It exists only one instance of this component inside of the Hedeby system. The host attribute defines the name of the host where this Singleton is started.

Configuration Service

The Configuration Service (CS) is a special component. It is not explicitly defined in a component tag. The Hedeby system detects automatically the JVM which hosts CS. It compares the hostname and the port with the CS URL stored in the preferences. The CS JVM must have a static port.

```
% sdmadm -s system1 show_configs -f
system  type    host       port   properties
-----------------------------------------

system1 SYSTEM master_host 31006❶
    spool=/var/spool/hedeby/system1
     dist=/opt/hedeby
```

```
<?xml version="1.0" encoding="UTF-8"?>
<global ...>
<jvm name="cs_jvm" user="sdm_admin" port="31006"(2)>
</jvm>
...
</global>
```

(1) The **sdmadm show_config** command shows the CS contact information including the hostname of the port of the CS component.

(2) At the startup of the JVM `cs` on the master host the Hedeby system automatically dectects that this JVM hosts CS.

Components and their Configuration

This chapter describes configuration of the Hedeby components.

Resource Provider

Overview

The Resource Provider is the main component of the Hedeby system which provides information about resources and processes all information from the managed services. There exists only one Resource Provider in a Hedeby system and it should run in the JVM started as admin user (No root privileges required).

Once the Resource Provider component has been started the command line clients can get information about resources and which resources are assigned to services.

It is necessary to define the following component configurations before the Resource Provider can be started:

- Basic preferences and directories (see the section called “Basics”)
- Global JVM configuration (see the section called “Java Virtual Machines”)
- CA Security (see the section called “Security”)

Configure the Resource Provider in the JVM system configuration

A typical component configuration entry in the system configuration file for the Resource Provider looks as follows:

Example 2.5. Example for Hedeby Resource Provider system component configuration

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<common:global name="mySystem"
               xmlns:executor="http://hedeby.sunsource.net/hedeby-executor"
               xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
               xmlns:security="http://hedeby.sunsource.net/hedeby-security"
               xmlns:resource_provider="http://hedeby.sunsource.net/hedeby-reso
               xmlns:common="http://hedeby.sunsource.net/hedeby-common"
               xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gridengine-

    <common:jvm port="0"
                user="root"
                name="rp_vm">
        <common:component xsi:type="common:Singleton"
                          host="foo"
                          autostart="true"
                          classname="com.sun.grid.grm.resource.impl.ResourcePro
                          name="resource_provider"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        ...
    </common:jvm>
    ...
</common:global>
```

Resource Provider configuration

The Resource Provider component configuration is stored as a component configuration in the config service.

The configuration file is written in XML format. The structure is defined in the xml schema `hedeby-resource-provider.xsd`.

Example 2.6. Example for Resource Provider configuration

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<common:componentConfig xsi:type="resource_provider:ResourceProviderConfig"❶
                        xmlns:executor="http://hedeby.sunsource.net/hedeby-exec
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xmlns:reporter="http://hedeby.sunsource.net/hedeby-repo
                        xmlns:security="http://hedeby.sunsource.net/hedeby-secu
                        xmlns:resource_provider="http://hedeby.sunsource.net/he
                        xmlns:common="http://hedeby.sunsource.net/hedeby-common
                        xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gr
    period="60">❷

    <resource_provider:policies xsi:type="resource_provider:PriorityPolicyManag
       defaultPriority="49">❸
        <resource_provider:priority service="spare_pool"
                                    name="spare_pool_priority">1</resource_prov
    </resource_provider:policies>
</common:componentConfig>
```

1. The Resource Provider configuration is special component configuration. the `xsi:type` attribute defines the real type (must be `ResourceProviderConfig` for the Resource Provider).
2. The polling period at which to reprocess unsatisfied requests. Resource Provider will periodically query the managed Service Containers in search of available resources, until either the needed resources are located, or the Service Container rescinds the resource request.
3. The `<policies` tag defines the configuration for the Policy Engine. Currently only a priority based Policy Engine is defined (`xsi:type="resource_provider:PriorityPolicies"`).
4. The `<priority>` defines the priority of each service. With the priority the urgencies of the services needs are weighted.

Reporter

Overview

Reporter component is a log/monitoring tool for Hedeby.

The role of reporter component is to intercept and gather informations about what is going on in the system. Administrator can sprecify what kinf of data he is interested in. Right now reporter is able to store informations and notifications that comes from Configuration Service, Resource Provider and all services that are installed in the system.

The reporter component is prepared to store data in arco data base. By prepared we mean that, there is a special arco format file created, that stores suitable for Arco data.

The data from Arco file arent so much readable for normal user, thats why Administrator can get and print out on the screen data using CLI commands. The data can be filtered using avaiable filters.

Reporter configuration

The Reporter component configuration is stored as a component configuration in the config service. The path to the configuration is defined in the `<config>` tag of the component definition.

The configuration file is written in XML format. The structure is defined in the xml schema `hedeby-reporter.xsd`.

Example 2.7. Example for Reporter configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<reporter:reporter
    xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
    xmlns:common="http://hedeby.sunsource.net/hedeby-common"
    filePattern="report-%g.log"
    fileCount="4"
    fileSize="5242880"/>
```

- *level*

  As default set to ALL. Specify the verbose level of the reporter component. With the ALL as a default all kind of rows will be stored in the file. For INFO only the NOTIFICATION. For FINE also RESOURCE, STATE, CONFIGURATION, NEED. For FINER - NEED_PROP and RES_PROP.

- *filePattern*

  As default set to report-%g.log. This value represent pattern for reporting file that will be used to store records in. The directory where this file could be found is the local spool of reporter component.

- *fileCount*

  As default set to 4. This value represent max number of reporting files that can be created for reporting data.

- *fileSize*

  As default set to 5242880. This value represent max size of reporting files that can be created for reporting data.

- *fileAppend*

  As default set to true. This value idicates wheter data can append to file or new file will be created instead.

- *showResourceProviderEvents*

  As default set to true. This value idicates wheter we want to report Resource Provider events or not.

- *showManagementEvents*

  As default set to true. This value idicates wheter we want to report Resource Provider resource processing events or not.

- *showCSEvents>*

  As default set to true. This value idicates wheter we want to report cs events or not.

Service Adapters, Grid Engine Adapter

Service Adapter is a component representing service that is managed by Hedeby System. Currently Hedeby supports only two type of services: Spare Pools and GE Adapter (managing Grid Engine).

The Service Adapter is capable of starting and stopping the currently running Service. To start a Service Adapter without starting a Service means to connect the Service Adapter to an already running Service. To stop a Service Adapter without stopping the associated Service means to disconnect from the Service but leave it running. Stopping a Service Adapter without stopping the associated Service also means that any resources assigned to that Service are effectively lost to Hedeby system.

Service Adapter talks to a specific Service. As Hedeby is supporting only Grid Engine (GE) there exists only Grid Engine Adapter. The specifics of gathering information in order to evaluate SLO's and the process of preparing and adding a new resource or releasing a current resource is handled by the Service Adapter. To achieve this GE Adapter uses JGDI and Executor the section called “Executors”.

The Service Adapter acts as the container for SLO's associated with its Service. The Service Adapter interacts with the Service to maintain a current perspective on the Service's SLO's. When an SLO is not being met, the Service Adapter must normalize that SLO into an urgency, an integer from 0 to 99. The Service Adapter also maintains a list of SLO priorities. When SLO non-compliance is raised to the Resource Provider, the Service Adapter first applies the SLO priority to the event, before sending it to the Resource Provider.

GEAdapter Configuration

A Grid Engine service in Hedeby is defined over an entry in the global configuration. Normally it is not necessary to modify this configuration. It is automatically created with the **sdmadm add_ge_service**. The following samples shows a typical definition of a Grid Engine Service.

Example 2.8. Example for Hedeby service system component configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<global name="new_hedeby_system"
        xmlns='http://hedeby.sunsource.net/hedeby-common'
        xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>
    ...
    <jvm name="ge_jvm" user="grm_admin" port="0">
        <component name="ge_service"
                   classname="com.sun.grid.grm.service.impl.ge.GEServiceImpl">❶
            <classpath>
                <pathelement>${GRM_DIST}/lib/sdm-ge-adapter.jar/>❷
            </classpath>
            <hosts>❸
                <include>master_host</include>
            </hosts>
            <config>ge_service</config>
        </component>
    </jvm>
    ...
</global>
```

❶ Name of the class which implements the service. In the case of the Grid Engine service this is always `com.sun.grid.grm.service.impl.ge.GEServiceImpl`.

❷ Necessary classpath for loading the service implementation. The Grid Engine Service needs also `jgdi.jar` in the classpath. It will be loaded in a extra classloader since it depends on `SGE_ROOT` (defined in the `GEServiceConfig`).

❸ Defines the name of the host where the GE service will run. Normally a service component is only started on one host. A Grid Engine service component runs normally on the host where qmaster is installed.

As all other configurations Grid Engine Service configuration is defined in an xml structure. It can be modified with the **sdmadm modify_component -c <component name>** command. This command opens the configuration in an editor. The following section describes the configuration of a GEAdapter:

Example 2.9. Example for Grid Engine service configuration

```
<?xml version="1.0" encoding="UTF-8"?>

<componentConfig xsi:type="ge:GEServiceConfig"❶
                    xmlns='http://hedeby.sunsource.net/hedeby-common'
                    xmlns:ge='http://hedeby.sunsource.net/hedeby-gridengine-ada
                    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>

    <slos>❷
        ...
    </slos>

    <ge:connection clusterName="ge"❸
                   root="/opt/sge" cell="default"
                   masterPort="31002" execdPort="31003" jmxPort="23011"
                   username="sge_admin" password="secret"
                   keystore="/var/sgeCA/port31002/default/userkeys/sge_admin/ke

    <ge:sloUpdateInterval unit="minutes" value="5"/>❹
    <ge:qmasterCheckInterval unit="seconds" value="2"/>❺
    <ge:qmasterReconnectInterval unit="seconds" value="2"/>❻
    <ge:execStartupTimeout unit="seconds" value="60"/>❼
    <ge:staticHosts>❽
        <include>foo.bar</include>
    </ge:staticHosts>


    <!-- Configuration for Solaris execds -->
    <ge:execd adminHost="true"❾
              submitHost="false" cleanupDefault="false" rcScript="false"
              ignoreFQDN="false" defaultDomain="foo.bar">

       <ge:filter>operatingSystemName = "SunOs" | operatingSystemName = "Solari

       <ge:localSpoolDir>/export/home/sge/execd</ge:localSpoolDir>

       <ge_adapter:installTemplate executeOn="exec_host">
            <ge_adapter:script>/opt/hedeby/grm_util/templates/ge-adapter/instal
            <ge_adapter:conf>/opt/hedeby/grm_util/templates/ge-adapter/install_
        </ge_adapter:installTemplate>
        <ge_adapter:uninstallTemplate executeOn="exec_host">
            <ge_adapter:script>/opt/hedeby/util/templates/ge-adapter/uninstall_
            <ge_adapter:conf>/opt/hedeby/grm_util/templates/ge-adapter/uninstal
       </ge_adapter:uninstallTemplate>

    <ge:execd>

    <-- Default execd install configuration -->
    </ge:execd>
       <ge:localSpoolDir>/usr/local/sge/execd</ge:localSpoolDir>❿
    <ge:execd mapping="default">

</componentConfig>
```

❶ A GEServiceConfiguration is a special component configuration. the `<xsi:type>` specifies the type. Must be `ge:GEServiceConfig` for a Grid Engine service.

❷ In the first section of a `GEServiceConfig` always defines the list of SLOs (for details see the section called “SLO definition”).

❸ The `<connection>` defines the connection parameters to the Grid Engine instance. The following parameters are required:

clusterName	Name of the cluster. The cluster name for Grid Engine has been introduced with version 6.2. It defines a unique name for the cluster. This parameter is necessary for the execd installation/uninstallation.
root	Path to root directory of the Grid Engine instance (`SGE_ROOT`).
cell	Cell name of the Grid Engine instance (`SGE_CELL`).
masterPort	Port where qmaster is listening. Required for installing an execd on a managed host. (`SGE_MASTER_PORT`).
execdPort	Port where execd is listening. Required for installing an execd on a managed host. (`SGE_EXECD_PORT`).
jmxPort	The Grid Engine Adapter communicates via JMX with qmaster. The `jmxPort` defines the port where qmasters MBeanServer is listening.
username	Username for JMX authentication. Must be a valid Grid Engine admin user.
password	Password for JMX authentication. Storing the password of inside of the GEAdpater configuration is not advisable. If possible a keystore for authentication should be used.
keystore	Defines the path to the keystore which contains the private key of the admin user. This credential will be used for JMX authentication against qmaster. The keystore can be created with the `sge_ca` script (included in Grid Engine 6.2 distribution, `$SGE_ROOT/util/sgeCA/sge_ca`). If the keystore is password protected the credentials can be specified in the password attribute of the connection element.

❹ The Grid Engine Adapter updates periodially it's SLOs. The `sloUpdateInterval` defines how often this update is executed. For each update a qstat over the JMX connection is executed.

The administrator has to specify the value and the unit of the update interval. Valid values for unit are `seconds`, `minutes` and `hours`.

❺ The Grid Engine Adapter observes the connection to qmaster periodically. The `qmasterCheckInterval` defines how often connection test is executed.

The administrator has to specify the value and the unit of the update interval. Valid values for unit are `seconds`, `minutes` and `hours`.

❻ If the connection to qmaster breaks down GEAdapter tries periodically a reconnection. The `qmasterReconnectInterval` parameter defines how ofter the reconnect is executed.

The administrator has to specify the value and the unit of the update interval. Valid values for unit are `seconds`, `minutes` and `hours`.

❼ Whenever GEAdapter installs an execd on a manage host it is waiting for the signal from qmaster that the execd is now running. The `execStartupTimeout` element defines how long GEAdapter is waiting for this signal. If this timeout occurs GEAdapter sets the resource into error state.

❽ In the staticHosts list the administrator defines a list of hosts the Grid Engine Adapter will never give up.

❾ Defines the parameter for installing/uninstalling an execd on a managed host. The following attributes are defined.

adminHost — The default value is `false`. If this attribute is set to `true` the host will become Grid Engine admin host. If it is set to `false` the host will only be admin host during the execution of the exec installation script.

submitHost — If this attibute is set to `true` assigned host will become Grid Engine submit host. The default value is `false`.

cleanupDefault — The default execd installation adds the new host on the Grid Engine side to the `@allhost` host group and to the `all.q` cluster queue. If this parameter is set to `true` GEAdapter will remove this host from the `@allhost` host group and the `all.q` cluster queue if after uninstalling the execd. The default value for this attribute is `false`.

rcScript — If this parameter is set to `true` the boot time startup scripts for the execd will be installed/uninstalled for the host.

ignoreFQDN — If `true` the execd will ignore the domain names during hostname resolving. If `false` execd will use the fully qualified domain name. The default value is

defaultDomain — Name of the default domain of the execd.

mapping — Name of the used complex to resource property mapping (the section called “Complex to Resource Property Mapping”).

A Grid Engine Adapter configuration can have serveral `execd` definitions. The `filter` element defines which execd settings are used for a specific host. The value of this element must be a valid resource property filter expression (the section called “Filtering”). The order of the `execd` elements matters. The first element where filter expression against matches against the resource properties of a host resource is used for the exec installation/uninstallation.

An `execd` element without `filter` element matches against each host resource. Each GEAdapter should have such a element with the default settings at the end of the configuration.

Defines the local spool directory used for the execd.

The default installation of an execd is done with the Grid Engine auto install feature. GEAdapter uses some templates and replaces in these templates placesholders with configured values. The Hedeby distribution contains default templates. The administrator can override the pathes in the templates to modify customize installation/uninstallation (the section called “Execd installation/uninstallation”) . Each template has a script and a configuration. In both files the configured values are replaced. The following templates are available.

installTemplate — This template is used for creating the script for installing a execd (default script is `<dist>/grm_util/templates/ge-adapter/install_execd.sh`, default configuration template is `<dist>/grm_util/templates/ge-adapter/install_execd.conf`).

uninstallTemplate — This template is used for creating the script for uninstalling a execd (default script is `<dist>/grm_util/templates/ge-adapter/uninstall_execd.sh.template`, default configuration template is `<dist>/grm_util/templates/ge-adapter/uninstall_execd.conf`)

The `executeOn` attribute of a install template defines where the scripts generated out of the template will be executed. If `executeOn` is set to `qmaster_host`, the script will be executed of the executor component running on Grid Engines qmaster host. If `executeOn` is set to `exec_host`, the script will be executed over the executor of the execd host (managed host).

In the above example the GEAdapter will create the file `install_execd.sh` and `install_execd.conf` in a temporar directory on the managed host. The executor will execute the following command on the managed host:

```
# cd <tmp directory<
# ./install_execd.sh install_execd.conf
```

SLO definition

```
<?xml version="1.0" encoding="UTF-8"?>

<componentConfig ...>
    <slos>
        <slo name="<name of SLO>"❶
             xsi:type="<type of SLO>"❷
             urgency="13"❸
             ...❹>
             <resourceFilter>❺
                 hardwareCpuArchitecture = "amd64" & operatingSystemName = "Sol
             </resourceFilter>
             ...❻
        </slo>
    </slos>
</componentConfig>
```

❶ Each SLO has a service wide unique name.

❷ Concrete type of the SLO. Valid values for a Grid Engine Adapater are `MinResourceSLOConfig`, `FixedUsageSLOConfig` and `MaxPendingJobsSLOConfig`.

❹ Depending on the SLO type additional attributes can follow.

❸ Urgency of this SLO. The urgency is a number between 0 and 99. Each resources which is required by this SLO has a usage which is more or equals to the urgency of the SLO (see also the section called “Resource Usage”).

❺ The `request` element defines the concrete resource which request once the SLO is not met. The request defines the required properties of a resource to full fill this SLO.
Depending on the SLO type additional elements can follow.

MinResourceSLO

The `MinResourceSLO` counts the number of assigned resources of a service which matches to a resource filter. This SLO ensure that the number these resources is not less then the defined minimum.

It further defines that all required resources for this SLO has the at least an usage which is equal or higher then the urgency of the SLO.

```
<?xml version="1.0" encoding="UTF-8"?>

<componentConfig ...>
    <slos>
        <slo name="minHostResourceSample"
```

```
                xsi:type="MinHostResourceSLOConfig"❶
                urgency="13"
                min="20"❷>
                <request>...</request>
                <resourceFilter>❸
                <![CDATA[
                    hardwareCpuArchitecture = "amd64" &
                    operatingSystemName ="Solaris"
                ]]>
                </resourceFilter>
            </slo>
        </slos>
</componentConfig>
```

❶ The concrete type is `MinResourceSLOConfig`.

❷ The `min` attribute defines the minimum number of matching resource for a met SLO.

❸ The MinResourceSLO allows a resourceFilter. This filter defines set of resources which a considered by this SLO. The fiter is applied against the properties if the assigned resource (assigned to the service, see also the section called “Filtering”).

PermanentRequestSLO

The `PermanentRequestSLO` sends permanent requests, a need for a resource. This SLO is mainly used in spare pools components to gather unused resources. Typicaly it has the lovest urgency set.

It is also possible to define an urgency and resource type that is wanted by the service which is using this SLO.

```
<?xml version="1.0" encoding="UTF-8"?>

<common:slos>
    <common:slo xsi:type="common:PermanentRequestSLOConfig❶
                urgency="1" ❷
                name="PermanentRequestSLO">
        <common:request>type = "host"</common:request>❸
    </common:slo>
</common:slos>
```

❶ The concrete type is `PermanentRequestSLOConfig`.

❷ The `urgency` parameter defines the urgency for the needs sent by this service

❸ This element defines the properties that describes a need which is sent by this service. It could be resource type host for example.

FixedUsageSLO

The `FixedUsageSLO` is a special SLO. It never produces any need. It only ensures the all assigned resources of the service which matches the resourceFilter of this SLO has a usage which is equal or higher then the urgency of the SLO.

```
<?xml version="1.0" encoding="UTF-8"?>

<componentConfig ...>
```

```
    <slos>
        <slo name="fixUsageSample"
             xsi:type="FixedUsageSLOConfig"❶
             urgency="13">
             <resourceFilter>❷
             <![CDATA[
                 hardwareCpuArchitecture = "amd64" &
                 operatingSystemName = "Solaris"
             ]]>
             </resourceFilter>
        </slo>
        ...
    </slos>
    ...
</componentConfig>
```

❶ The concrete type is `FixedUsageSLOConfig`.

❷ All resources which matches this resourceFilter are considered by the SLO. (the section called “Filtering”)

MaxPendingJobsSLO

The `MaxPendingJobsSLO` counts the number of pending jobs of a Grid Engine instance which matches a job filter. If the number of jobs is higher then a maximum value a need is produced.

Each host resource which matches the request filter and which runs jobs matching to the job filter will have at least the usage of the MaxPendingJobsSLO.

```
<?xml version="1.0" encoding="UTF-8"?>

<componentConfig xmlns:ge="http://hedeby.sunsource.net/hedeby-gridengine-adapte
    ...>
    <slos>
        <slo name="maxPendingJobs"
             xsi:type="ge:MaxPendingJobsSLOConfig"❶
             urgency="13"
             max="100"❷>
             <resourceFilter>❸
             <![CDATA[
                 hardwareCpuArchitecture="amd64" &
                 operatingSystemName="Solaris"
             ]]>
             </resourceFilter>
             <ge:jobFilter> license_1 = "1"</ge:jobFilter>❹
        </slo>
        ...
    </slos>
    ...
</componentConfig>
```

❶ The concrete type is `ge:MaxPendingJobsSLOConfig`. The configuration of this SLO is defined in a different xml namespace (`xmlns:ge="http://hedeby.sunsource.net/hedeby-gridengine-adapter"`). The type of the configuration must have a namespace prefix. All elements which are specific to the MaxPendingJobsSLO must also have the prefix.

❸ If the MaxPendingJobsSLO detects that the cluster has more pending jobs as defined in attribute `max` it produces a need. The properties of the needed resources are defined in the `request` element.

❹ All resources which matches this jobFilter are considered by the SLO. (the section called “Filtering”) The jobFilter allows filtering of pending jobs according to the hard resource requests.

Filtering

The configuration of the Hedeby components allows on serveral place the defintion of filters. All this filters uses the same filter language. The following listing shows the generic definition of a filter in Backus–Naur form (BNF):

```
filter:         orExpr <EOF>
orExpr:         andExpr ("|" andExpr)*
andExpr:        expr ("&" expr)*
expr:           "(" orExpr ")" | "!" orExpr | booleanExpr
booleanExpr:    compareExpr | matchEpr
compareExpr:    value ( "<"|"<="|"="|"!="|">="|">" ) value
matchExpr:      "matches" stringLiteral
value:          identifier | constant
constant:       int_literal | float_literal | string_literal | char_literal
identifier:     ["a"-"z","A"-"Z","_"] ( ["a"-"z","A"-"Z","_","0"-"9","."] )*
int_literal:    integer literal as in java (e.g. 10, 1L, xFF)
bool_literal:   "true" | "false"
float_literal:  floating point literal as in java (e.g. 12.0E3)
char_literal:   characater literal as in java (e.g. 'a')
string_literal: string literal as in java (e.g. "a")
```

Example 2.10. Job filters for the `MaxPendingJobsSLO`

The `MaxPendingJobsSLO` SLO allows a filter for matching jobs. For such job filters all hard request of the job can be used for filtering.

```
arch = "sol-amd64" & lic = "1"
```

The above filter will match if a Grid Engine job is submitted with **qsub -l arch=sol-amd64 -l lic=1**.

```
arch matches "sol-.*"
```

With the matches expression regular expressions can be used for filtering. The above filter matches against all jobs with a hard request for complex arch starting with sol.

Example 2.11. Resource Request Filter for SLOs

If a SLO has a need it sends a request to Resource Provider. This request contains a request filter which is used to find matching resource. The request filter has access to any resource property of the resources.

```
(hardwareCpuArchitecture = amd64 & hardwareCpuCount >= 2)  |
(hardwareCpuArchitecture = sol-sparc64 & hardwareCpuCount >= 4)
```

The above resource filter will match against all resources with more then one amd64 CPU or all sol-sparc64 hosts with more then 3 CPUs

```
state != "ERRROR"
```

The last example shows that the request filter can also filter resource according to it's state. Only resource which are not in error state will match.

Complex to Resource Property Mapping

GEAdpater automatically updates the properties of the assigned host resource. With the "Complex to Resource Property Mapping" the administrator can define what complex values are used. After the installation of the first Grid Engine Service the default mapping is installed. It can be displayed with the **sdmadm show_ge_complex_mapping** command.

For a complex mapping the administrator has to define the name of the complex the value of the complex and the list of resource property which will be used.

```
<ge:mapping>
    <ge:resource>
        <source name="arch">sol-sparc64</source>❶
        <target>
            <property name="hardwareCpuArchitecture">sparcv9</property>
            <property name="operatingSystemName">Solaris</property>
        </target>
    </resource>
    <ge:resource>
        <source name="num_procs">*</source>❷
        <target>
            <property name="hardwareCpuCount">$VALUE</property>
        </target>
    </resource>
</ge:mapping>
```

❶ Each host which has the `arch` complex set to sol-sparc64 is mapped into a host resource with the resource properties `hardwareCpuArchitecture=sparcv9`, `operatingSystemName=Solaris`.

❷ If a host has defined the complex `num_proc` the value if this complex is transformed into a resource property `hardwareCpuCount`. The value of the resource property is the same as the value of the complex.

Execd installation/uninstallation

When ever a resource is assigned to a Grid Engine service the Grid Engine Adapter tries to install an exec daemon on the managed host. When ever a resource is removed from an execdaeomon it uninstalls the execd daemon from the managed host. This section describes how install/uninstall is done.

Per default Grid Engine uses Grid Engines auto installation feature to perform an execd installation/ uninstallation. The installation is done by executing a install script and with configuration file on the executor of the manged host.

GEAdpater expects the following exit value from the install/uninstall scripts:

0 Execd daemon has been successfully installed/uninstalled.

1 Installation/uninstallation was not successful. The host has been modified. In this case the host resource will be set into error state. Administrator has to cleanup the host. It will be reused after a reset (**sdmadm reset_resource**).

2 Installation was not successful. However the host has not been modified. In this case the GEAdapter will reject the resource. This means it tells the Resource Provider that it can not use the resource. Resource provider will assign the resource to a different host

The installation script and the configuration file is generated out of templates. The pathes to the templates can be defined in the `execd` elements of the GEAdapter configuration (see also Example 2.9, “Example for Grid Engine service configuration”). GEAdapter replaces in this templates some placeholder with the installing settings. The following placeholders are used:

`@@@SGE_CLUSTER_NAME@@@`	Name of the Grid Engine Cluster name (content of `$SGE_ROOT/$SGE_CELL/common/cluster_name` file
`@@@SGE_ROOT@@@`	The Grid Engine root directory (`$SGE_ROOT`).
`@@@CELL_NAME@@@`	Name of the Grid Engine cell (`$SGE_CELL`)
`@@@EXEC_HOST@@@`	Name of the host where the executor will be installed
`@@@SGE_EXECD_PORT@@@`	Port of the execd daemon of the cluster
`@@@SGE_QMASTER_PORT@@@`	Port where qmaster is listening for incoming requests.
`@@@EXEC_HOST_LIST@@@`	Contains the name of the execd which will be installed (only set for installation).
`@@@EXEC_HOST_LIST_RM@@@`	Contains the name of the execd which will be unnstalled (only set for uninstallation).
`@@@DEFAULT_DOMAIN@@@`	Name of the default domain if the cluster (needed for hostname resolving).
`@@@HOSTNAME_RESOLVING@@@`	If set to `false` hostnames will be resolved by the full qualified host name.
`@@@ADD_TO_RC@@@`	Should the boot time startup scripts be installed (only set for installation, this does not include SMF support).
`@@@REMOVE_RC@@@`	If this value is set to true the boot time startup script will be removed (only set for uninstallation).
`@@@SUBMIT_HOST_LIST@@@`	If the execd should become a submit host the hostname is included in the `@@@SUBMIT_HOST_LIST@@@`.
`@@@EXECD_SPOOL_DIR_LOCAL@@@`	Contains the path of the local spool directory for a exec daemon.

Default install template

```
BASEDIR=`pwd`
SGE_ROOT="@@@SGE_ROOT@@@"

if [ ! -d "$SGE_ROOT" ]; then
    echo "SGE_ROOT directory $SGE_ROOT does not exists"
    exit 2
fi
if [ ! -f "$SGE_ROOT/inst_sge" ]; then
    echo "inst_sge script in directory $SGE_ROOT not found"
    exit 2
fi

if [ ! -x "$SGE_ROOT/inst_sge" ]; then
    echo "inst_sge script in directory $SGE_ROOT is not executable"
    exit 2
fi

if [ ! -f "$BASEDIR/install_execd.conf" ]; then
    echo "auto config file $BASEDIR/install_execd.conf not found"
    exit 2
fi

cd "$SGE_ROOT"
./inst_sge -x -noremote -auto $BASEDIR/install_execd.conf
res=$?
exit $res
```

Executors

Overview

The general overview of Executor component can be found in the section called “ Executor ” and the section called “Executor”

Definition of Executor in Hedeby System

The General configuration of Executor component is specified in Java Virtual Machines section For detailed information about specific fields (see the section called “Java Virtual Machines”).

Example 2.12. Executor configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<executor:executor
                 idleTimeout="60"❶
                 maxPoolSize="10"❷
                 corePoolSize="3"❸
                 keepFiles="false"❹
                 xmlns:executor="http://hedeby.sunsource.net/hedeby-executor"
                 xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
                 xmlns:security="http://hedeby.sunsource.net/hedeby-security"
                 xmlns:resource_provider="http://hedeby.sunsource.net/hedeby-re
                 xmlns:common="http://hedeby.sunsource.net/hedeby-common"
                 xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gridengin
```

1. The executor manages a thread pool. Each thread executes and observes a command. The `idleTimeout` defines how long a thread is kept alive in the thread pool if it is idle.
2. Maximum number of threads in the executor thread pool. This parameter limits the number of concurrently executed commands.
3. The minimum number of thread in the thread pool. This number of thread is always kept alive, even if they are idle.
4. If the `keepFiles` flag is set the executor does not delete the temporary files of the executed commands. This is helpful for debugging purpose.

Note

To be fully operative, the Executor has to be run on JVM that is run as "root" user who has uid equal to 0. Otherwise, the user switching is not working anymore. Executor should be defined on each resource host.

Example 2.13. Typical executor component definition

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<common:global name="mySystem"
               xmlns:executor="http://hedeby.sunsource.net/hedeby-executor"
               xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
               xmlns:security="http://hedeby.sunsource.net/hedeby-security"
               xmlns:resource_provider="http://hedeby.sunsource.net/hedeby-r
               xmlns:common="http://hedeby.sunsource.net/hedeby-common"
               xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gridengi
    <common:jvm port="0"
                user="root"
                name="executor_vm">
        <common:component xsi:type="common:MultiComponent"
                          autostart="true"
                          classname="com.sun.grid.grm.executor.impl.Executor
                          name="executor"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
            <common:hosts>
                <common:include>.*</common:include>
            </common:hosts>
        </common:component>

        ...
    </common:jvm>
    ...
</common:global>
```

Details about implementation of Executor

This section describes some implementation details about Executor. This information explains some parameters that are present in component configuration.

- Executor as JMX Dynamic MBean

  Executor is wrapped and exposed to other components in Hedeby (Services) as JMX Dynamic Mbean, so the Service that wants to execute some command on host with Agent has to connect to MBean Server, obtain MBean and than use attributes and operations that are exposed by this MBean.

  Operations:

  - *setKeepFiles* - set attribute keepFiles

- *isKeepFiles* - read attribute keepFiles
- *execute* - execute command or script
- *cleanup* - makes the cleanup of all temp directories, files that were created during execution of command by Executor

Attributes:

- *keepFiles* - information whether the spool directories have to be kept alive after execution is finished

- How executor executes command/script

  The execution of command/script is encapsulated in Command class. The Executor receives Command class object that contains information about user under which the execution should be processed, this object contains also information about environment (environment variables) and the script/command. The executor has some pool with threads (it's created with ThreadPoolExecutor class for more details see (ThreadPoolExecutor [http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/ThreadPoolExecutor.html]). So each command that was requested to be executed by Executor is executed in some thread from this pool. To create such pool Executor needs to know such parameters as:

  - *corePoolSize*
  - *maxPoolSize*
  - *idleTimeout*

  The meaning of this parameters is as described in the section called “Definition of Executor in Hedeby System” The pool with threads is created during start of Executor. So on object command the execution is called on thread. The command contain information about user under which it should be run, command is responsible of setting environmental variables, transporting some additional files that are neccessary to execute command, setting the right permission of files. After execution the result object containing exitCode and output is returned.

Spare Pool

Overview

The general overview of Spare Pool component can be found in the section called “Spare Pool”

Definition of Spare Pool in Hedeby System

Example 2.14. SparePool configuration

```
    <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <common:componentConfig xsi:type="spare_pool:SparePoolServiceConfig"
                        xmlns:executor="http://hedeby.sunsource.net/hedeby-exec
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        xmlns:spare_pool="http://hedeby.sunsource.net/hedeby-sp
                        xmlns:reporter="http://hedeby.sunsource.net/hedeby-repo
                        xmlns:security="http://hedeby.sunsource.net/hedeby-secu
                        xmlns:resource_provider="http://hedeby.sunsource.net/he
                        xmlns:common="http://hedeby.sunsource.net/hedeby-common
                        xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gr
    <common:slos>(1)
        <common:slo xsi:type="common:PermanentRequestSLOConfig
                    urgency="1"
                    name="PermanentRequestSLO">
            <common:request>type = "host"</common:request>
        </common:slo>
    </common:slos>
</common:componentConfig>
```

(1) The only elements that can be define in spare pool configuration are SLOs. More about SLOs, you can find here the section called “SLO definition”. In this example we defined a PermanentRequestSLO, which is constantly asking for new host type resource. This is common usage for a spare pool.

Note

In the system there could be defined as many spare pool`s components as administrator wants. Here it is an example how to add spare pool component configuration to a JVM

Example 2.15. Typical spare pool component definition

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<common:global name="mySystem"
               xmlns:executor="http://hedeby.sunsource.net/hedeby-executor"
               xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
               xmlns:security="http://hedeby.sunsource.net/hedeby-security"
               xmlns:resource_provider="http://hedeby.sunsource.net/hedeby-r
               xmlns:common="http://hedeby.sunsource.net/hedeby-common"
               xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gridengi
    <common:jvm port="0"
                user="user_name"
                name="sparepool_vm">
        <common:component xsi:type="common:Singleton"
                          classname="com.sun.grid.grm.sparepool.SparePoolSer
                          name="spare_pool"
                          host="foo"
                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
        </common:component>
        ...
    </common:jvm>
    ...
</common:global>
```

Security

Hedeby is secured by:

- Ensure the trust relationship between Hedeby components.
- Ensure communication confidentiality and integrity.
- Control access to the components.

CA component

All communication between Hedeby components is done over SSL sockets. This requires that each component has a private key in its keystore, which is created during the installation of a component. The keysotre contains also certificate, which is trusted in Hedeby system.

There is a CA component running on main host of Hedeby system. During the first installation of Hedeby this component must be installed. There will be trusted certificate and private key created for this CA component.

The CA component manage all private keys and certificates. It is managed component (JMX MBean) that implements interfece GrmCA that can be used by clients to for example renew certificate or create new keystore. Each component has access to certificate of CA and with this certificate it is validated wheter component is trusted or not see the GrmCATrustManager class.

Installation of new hosts require CA component running. JMX MBean that represent CA is used to create certificates private keys and keystores for new daemons.

Example 2.16. Typical CA component configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<common:componentConfig
    xsi:type="security:CAComponentConfig"❶
    adminUser="ca_admin"❷
    xmlns:executor="http://hedeby.sunsource.net/hedeby-executor"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
    xmlns:security="http://hedeby.sunsource.net/hedeby-security"
    xmlns:resource_provider="http://hedeby.sunsource.net/hedeby-resource-provid
    xmlns:common="http://hedeby.sunsource.net/hedeby-common"
    xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gridengine-adapter">
    <security:sgeCaScript>/opt/sge/util/sgeCA/sge_ca</security:sgeCaScript>❸
</common:componentConfig>
```

❶ The CA component configuration is stored as special component configuration. The type must be `sec:CAComponentConfig`.

❷ Name of the user which is the administrator of the CA. If the CAComponent is running in a root JVM all commands will be executed under this user account.

❸ Path to the `sge_ca` scripts. The Hedeby security uses the Grid Engine certificate authority script for managing certificate and private keys. It is included into the Grid engine distribution (`$SGE_ROOT/util/sgeCA/sge_ca`).

Example 2.17. Typical CA component definition

```
<?xml version="1.0" encoding="UTF-8"?>
<common:global name="new_hedeby_system"
    xmlns:executor="http://hedeby.sunsource.net/hedeby-executor"
    xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
    xmlns:security="http://hedeby.sunsource.net/hedeby-security"
    xmlns:resource_provider="http://hedeby.sunsource.net/hedeby-resource-pr
    xmlns:common="http://hedeby.sunsource.net/hedeby-common"
    xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gridengine-adapter
    <common:jvm name="root_jvm" user="root"❶ port="0">
        <common:component xsi:type="common:Singleton"
            host="ge5" ❷
            autostart="true"
            classname="com.sun.grid.grm.security.ca.impl.GrmCAServiceDelega
            name="ca"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>

            ...
    </common:jvm>
    ...
</common:global>
```

❶ To be fully functional the CA component must run as privileged user, it should be instantiated in the root jvm. However running the CA component as non privileged user is possible but not recommended. In this case the created ownership of the created keystores are not changed. This can be a security problem.

❸ The CA component in implemented in class `com.sun.grid.grm.security.ca.impl.GrmCAServiceDelegate`.

❷ A Hedeby system can only have one instance of the CA component. It is normally running on the master host.

Authentication

For authenticating communication sides in Hedeby we use JAAS. Communication between Hedeby components requires authentication of the SSL layer for both communication partners. To lower the effort of certificate distrubution Hedeby component do not require an authentication of a CLI client on the SSL layer. Instead the clients has to provide credentials on higher communication layer (JMX authentication). Each Hedeby component provides a JMX interface, that is running in a MBeanServer. Client which want's to manage the component has to provide some credentials to gain access to the component. The credential are passed through the JMXAuthenticator mechanism to the MBeanServer. The MBeanServer uses JAAS to authenticate the user. Generally a client has to provide username and password to gain access to a component. It will be possible that a component gives the authenticated user a challenge. This challenge has a limited lifetime and can be used for passwordless authentication. The challenge needs to be stored on a save place (e.g keystore of the user). The authentication itself is delegated to JAAS LoginModules. LoginModules can be stacked. The final LoginModule must be the Hedeby RoleLoginModule which assigns the role to an authenticated user (e.g admin, user). For details about the JAAS system see http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/JAASRefGuide.html.

- Password less authentication - For password less authentication the communication partner need a valid certificate which is signed by the CA. This certificate is obtained by the SSL communication layer. For password less authentication the Hedeby system uses the GrmCATrustManagerLoginModule. It authenicates a coummnication partner if the SSL layer provides a X509 certificate which is trusted by the CA (used for password less authentication).

- Username/Password authentication - While daemon always uses passwordless authentication, for users it is possible to authenticate with a username/password pair. The remote component validates these credentials using JAAS login modules. The client application is responsible for providing username and password.

- LoginModules for Username/Password authentication

  - com.sun.security.auth.module.JndiLoginModule - Username/Password authentication against NIS and LDAP. Included in the Sun Java implementation. Needs additional jar files containing the JNDI providers.

  - com.sun.grid.security.login.UnixLoginModule - Username/Password authentication against PAM and system specify authentication system (e.g. passwd/shadow). Included in the Grid Engine distribution.

Authorization

Authorization is performed by the JAAS system base on java permissions. After authenticating a communication partner the JAAS LoginModule adds Principals to the java security Subject. All actions triggered by the communication partner are executed under this security subject. The java system checks wether the security subject has permissions to execute critical code.

- Roles - Each role defines a set of permission. Permissions are not directly granted to users, groups or daemons, instead the comunication partner is a member of a role. The Hedeby system provides a special role login module which adds a RolePrincipal to the subject of an authenticated users. Necessary permissions are granted to the RolePrincipal. The RoleLoginModule must be the last in the authentication stack. It searches in the role definition configuration wether a Prinicipal of the subject is member of a role and adds the corresponding RolePrincipal to the subject. Currently the Hedeby system knows only two roles:

  - administrator - All users which should have access the Hedeby system should be in the adminstrator roles.

  - daemon - All Hedeby daemons (JVMs) are in the daemon roles.

- JAAS configuration - JAAS for the Hedeby system is configured in the file <local_spool>/security/jaas.config.

```
HedebySystem {

 com.sun.grid.grm.security.GrmCATrustManagerLoginModule requisite;
 com.sun.security.auth.module.JndiLoginModule requisite
       user.provider.url="nis://<nis server>/<nis domain>/user"
       group.provider.url="nis://<nis server>/<nis domain>/system/group";

 com.sun.grid.security.login.UnixLoginModule requisite
       sge_root="${com.sun.grid.grm.bootstrap.sgeroot}"
       auth_method="pam" pam_service="su";
 com.sun.grid.grm.security.role.RoleLoginModule required
      roleFile="${com.sun.grid.grm.bootstrap.shared}/security/roles.config";
};
```

- Defining permissions - Hedeby deliveres a standard java policy file which defines the set of permissions for the roles of the grm system. The installation will copy this file into <local_spool>/security/java.policy. All JVMs in the Hedeby system uses this file a default policy file.

Example 2.18. Typical Hedeby security configuration

```
<?xml version="1.0" encoding="UTF-8"?>

<security:security xmlns:executor="http://hedeby.sunsource.net/hedeby-executor"
             xmlns:reporter="http://hedeby.sunsource.net/hedeby-reporter"
             xmlns:security="http://hedeby.sunsource.net/hedeby-security"
             xmlns:resource_provider="http://hedeby.sunsource.net/hedeby-resour
             xmlns:common="http://hedeby.sunsource.net/hedeby-common"
             xmlns:ge_adapter="http://hedeby.sunsource.net/hedeby-gridengine-ad
    <security:caCertificate>(1)
<![CDATA[
-----BEGIN CERTIFICATE-----
MIIENzCCA6CgAwIBAgIJANpU7eBw4MMhMA0GCSqGSIb3DQEBBAUAMIHEMQswCQYD
...
9Ac8O04k16QJt2K7EtVm/rkB2BKrwEu66Pid
-----END CERTIFICATE-----
]]>
    </security:caCertificate>

    <security:role name="admin">(2)
        <security:principal classname="com.sun.grid.security.login.UserPrincipa
            <security:name>grm_admin</security:name>
        </security:principal>
    </security:role>
    <security:role name="daemon">(4)
        <security:principal classname="com.sun.grid.grm.security.GrmDaemonPrinc
            <security:regexp>.*</security:regexp>
        </security:principal>
    </security:role>
</security:security>
```

(1) The security configuration contains the certificate authority certificate as a PEM encoded string. The CA component will update this information when renewing the CA certificate.

This information must be public available in the configuration service.

(2) Definition of the admin roles. All members of this roles have administration permissions on the Hedeby system.

(3) In this example the user grm_admin is a administrator. The principal definition defines the classname of the principal class and the name of the principals object.

(4) Beside the admin role the Hedeby system knows the daemon role. All JVMs of a Hedeby System are in the daemon role. They must have the GrmDaemonPrincipal.

Index