THE
DATA
PROTECTION
COMPANY

SafeNet HSM

TECHNICAL NOTE

Overview of Luna High Availability and Load Balancing

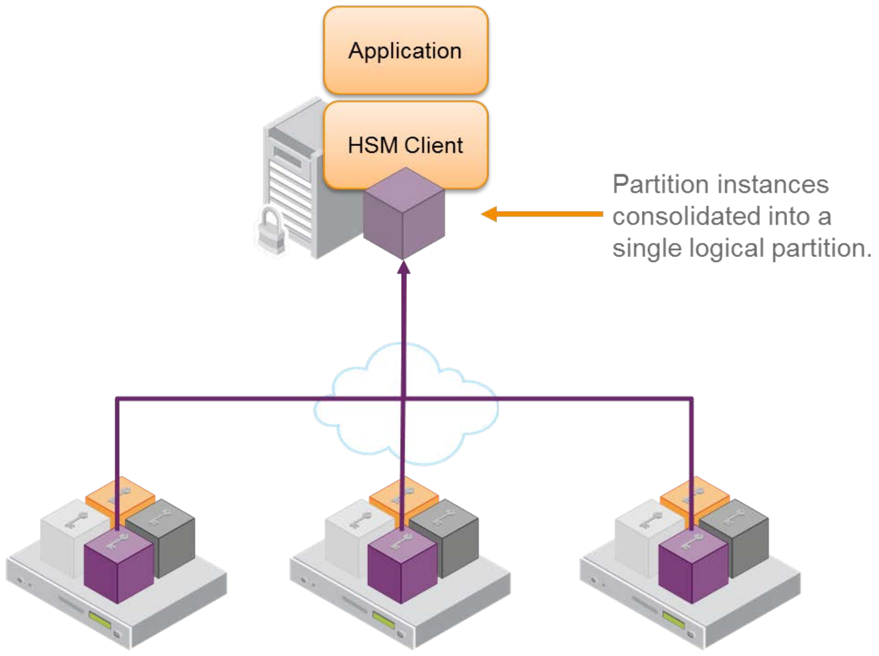
Contents

Introduction

Luna HSM products include availability and scalability capabilities for mission critical applications that require uninterrupted uptime. The feature allow the grouping of multiple devices into a single logical group – known as an *HA group*. When an HA group is defined, cryptographic services remain available to the consuming applications while at least one member in the group remains functional and connected to the application server. In addition many cryptographic commands are automatically distributed across the HA group to enable linear performance gains for many applications. This paper describes these features and the available configuration options in detail to help a user understand how to best configure the HA groups for their application and environment.

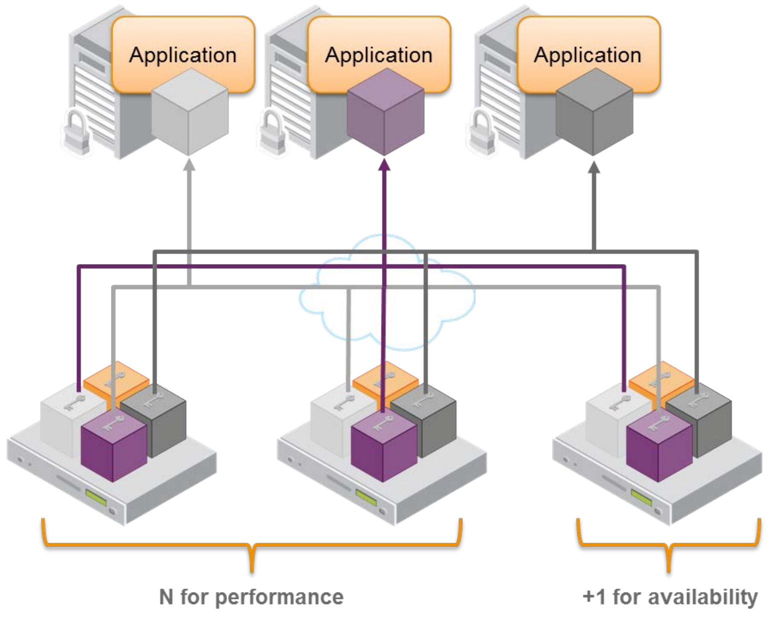
Overview

The Luna high-availability (HA) and load balancing (LB) functionality is implemented in the HSM client libraries. The HSMs and appliances are not involved and are unaware that they may be configured in an HA group. This allows the user to configure HA on a per-application basis. On each application server, the user defines an HA group by first registering the server as normal clients to all the desired HSMs. Then, client-side administration commands are used to define the HA group and set any desired configuration options. The user can configure several settings including: automatic or manual recovery mode; set some HSMs as standby members; and perform various manual synchronization and recovery operations.   Once defined, the library presents to the application a *virtual* HSM that is a consolidation of all the physical HSMs in the HA group. From this point on the library distributes operations and automatically synchronizes key material transparently to the application.

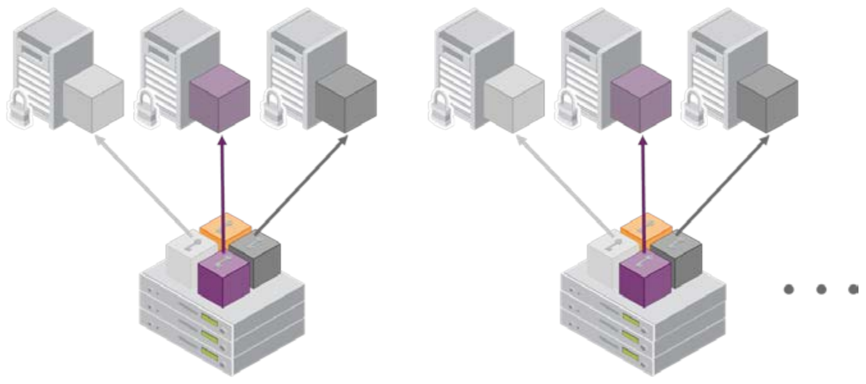

Application
HSM Client
Partition instances consolidated into a single logical partition.

High Availability

The Luna high availability function supports the grouping of up to sixteen members. However, there is a trade-off between performance and the cost of replicating key material across the entire group and this tradeoff will drive the maximum practical group size for your application.   A common practice is to set the group size to N+1 where N is defined by the desired performance per application server(s). As depicted below, this solution gives the desired performance with a single extra HSM providing the availability requirement. The number of HSMs per group of application servers varies based on the application use case, but as depicted groups of three are typical.



As performance needs grow beyond the performance capacity of three HSMs, it often makes sense to define a second independent[1] group of application servers and HSMs to further isolate applications from any single point of failure. This has the added advantage of facilitating the distribution of HSM and application sets in different data centers.



[1] With some applications is makes sense to make these partially independent using the Standby capability (see later in the document). It depends on the overall application and solution architecture.

Whenever an application creates key material, the HA functionality transparently replicates the key material to all members of the HA group before reporting back to the application that the new key is ready. The HA library always starts with what it considers its primary HSM (initially the first member defined in a HA group). Once the key is created on the primary it is automatically replicated to each member in the group. If a member fails during this process the key replication to the failed member is aborted after the fail-over time out[2]. If any member is unavailable during the replication process (i.e. the unit failed before or during the operation), the HA library keeps track of this and will automatically replicate the key when that member rejoins the group[3]. Once the key is replicated on all active members of the HA group a success code is returned to the application.

Load Balancing

The default behavior of the client library is to attempt to load-balance the application's cryptographic requests across the entire set of devices in the HA group. At the top level the algorithm used is a round-robin scheme that is modified to favor the least busy device in the set. As each new command is processed the Luna client looks at how many commands it has scheduled on every device in the group. If all devices have an equal number of outstanding commands the new command is scheduled on the next device in the list – creating a round-robin behavior. However, if the devices have a different number of commands outstanding on them, the command is scheduled on the device with the fewest commands queued – creating a least-busy behavior. This modified round-robin has the advantage of biasing load away from any device currently performing a lengthy-command. In addition to this least-busy bias, the type of command also affects the scheduling algorithm.

Single-part (stateless) cryptographic operations are load-balanced. However, multi-part (stateful) and key management commands are not load-balanced. Multi-part operations carry cryptographic context across individual commands. The cost of distributing this context to different HA group members is generally greater than the benefit. For this reason multi-part commands are all targeted at the primary[4]. Multi-part operations are either not used or infrequent actions, so most applications are not affected by this restriction. Key management commands affect the state of the keys stored in the HSM. As such, these commands are targeted at *all* HSMs in the group. That is the command is performed on the primary HSM and then the result is replicated to all members in the HA group. Key management operations are also an infrequent for most applications[5].

It is important to understand that the least-busy algorithm uses the number of commands outstanding on each device as the indication of its busyness. When an application performs a repeated command set, this method works very well. However, when the pattern is interrupted, the type of command can have an impact. For example, when doing signing and an atypical asymmetric key generation request is issued some number of the application's signing commands will get scheduled on the same device (behind the key generation). Commands *queued* behind the key generation will therefore have a large latency driven by the key generation. However, the least-busy characteristic will automatically schedule more commands to other devices in the HA group minimizing the impact of the key generation.

It is also important to remember that the load-balancing algorithm operates independently in each application process. Multiple processes on the same client or on different clients do not share their "busyness" information while making their scheduling choice. In most cases this is reasonable, but some mixed use cases may cause certain applications to hog the HSMs.

Finally, when an HA group is shared across many servers different initial members can be selected when defining the HA group on each server. The member first assigned to each group becomes the primary. Taking this approach will optimize an HA group to distribute the key management and/or multi-part cryptographic operation load more equally.

In summary, the load-balancing scheme used by Luna is a combination of round-robin and least-busy for most operations. However, as required the algorithm adapts to various conditions and use cases so it may not always emulate a round-robin approach.

---

[2] See fail-over discussion later in this document

[3] This functionality relies on the process creating the key material remaining active until after the failed member recovers. If the creating process has terminated, a manual synchronization will be required.

[4] Future releases will distribute multi-part commands across members through a "work load distribution" scheme.

[5] Work load distribution" will improve the scalability of these operations also.

Failover

When an HA group is running normally the client library continues to schedule commands across all members as described above. The client is continuously monitoring the health of each member at two different levels. First, the connectivity with the member is monitored at the networking layer. Disruption of the network connection invokes a fail-over event within a twenty second timeout[6]. Second, every command sent to a device is continuously monitored for completion. Any command that fails to complete within twenty[7] seconds also invokes a fail-over event. Most commands are completed within milliseconds. However, some commands can take extended periods to complete – either because the command itself is time-consuming (eg. key generation); or because the device is under extreme load. To cover these events the HSM automatically sends “heartbeats” every two seconds for all commands that have not completed within the first two seconds. The twenty second timer is extended every time one of these heartbeats arrives at client, thus preventing false fail-over events.

A fail-over event involves dropping a device from the available members in the HA group. All commands that were pending on the failed device are transparently rescheduled on the remaining members of the group. So when a failure occurs, the application experiences a latency stall on some of the commands in process (on the failing unit) but otherwise will see no impact on the transaction flow[8][9]. Note that the least-busy scheduling algorithm automatically minimizes the number of commands that stall on a failing unit during the twenty second timeout.

When the primary unit fails, clients automatically select the next member in the group as the new primary. Any key management or single-part cryptographic operation will be transparently restarted on a new group member. In the event that the primary unit fails, any in progress multi-part cryptographic operations will need to be restarted by the application as the operation will return an error code.

As long as one HA group member remains functional, cryptographic service is maintained to an application no matter how many other group members fail. As discussed in the Recovery section below, members can also be put back into service without restarting the application.

Recovery

After a failure, the recovery process is typically straight-forward. Depending on the deployment, an automated or manual recovery process may be appropriate. In either case there is no need to restart an application! With automatic recovery, the client library automatically performs periodic recovery attempts while a member is failed. The frequency of these checks is adjustable and can be limited on the number of re-tries. Each time a reconnection is attempted there is a slight delay in one application command while the library attempts to recovery. As such, the retry frequency cannot be set any faster than once per minute. Even if a manual recovery process is selected the application does not need to be restarted. Simply run the client recovery command and the recovery logic inside the client makes a recovery attempt the next time the application uses the HSM. As part of recovery any key material created while the member was offline is automatically replicated to the recovered unit[10].

Sometimes a failure of a device is permanent. In this event, the only solution is to deploy a new member to the group. Only in this case will the application will need to be restarted. Remove the failed unit from the HA group, add a new device to the group and then restart your application. The new application instance will immediately pick up all the members in the newly formed group. You may also need to perform a manual synchronization before the restart so that permanent key objects are replicated to the new device.

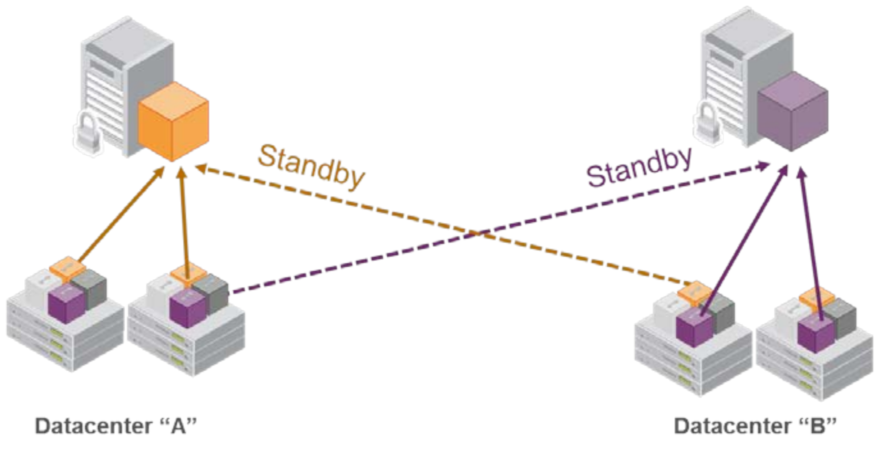
Finally, sometimes both an HSM and application fail at the same time. If, while an HSM was offline, no new key material was created the recovery is still straightforward: simply return the HSM to service and then restart the

[6] Network or device failures that close the socket will fail over immediately.
[7] Customers may modify this value to their application needs.
[8] Exception to this is if the primary fails and the application performs multi-part crypto. In this event the application must re-start the multi-part operation.
[9] Luna 5.2 has an exception where errors can be returned if key generation commands are in process while recovering a previously failed primary member.
[10] This functionality relies on the process creating the key material remaining active until after the failed member recovers. If the creating process has terminated, a manual synchronization will be required.

application. However, if new key material was created after an HSM failed but before the application failed, a manual re-synchronization may be required[11]. Confirm which member or members have the current key material (normally the unit(s) that was online at the time the application failed). Put them back in service with the application. Then, for each member that has stale key material (a copy of an object that was deleted; or an old copy of an object who's attributes were changed), delete all their key material after first making sure they are not part of the HA group. Be particularly careful that the member is not part of the HA group or the action may destroy active key material by causing an accidental synchronization during the delete! After the HSM is cleared of key material, rejoin it to the group and the synchronization logic will automatically repopulate the device's key material from the active units.

Standby Mode

By default all members in an HA group are treated as active. That is, they are kept both current with key material and used to load-balance cryptographic services. There are some deployment scenarios where it makes sense to define some members as standby. Standby members are registered just like active members except after adding them to the HA group they are defined as "standby". As depicted below applications can be deployed in geographically dispersed locations. In this scenario, use Luna's standby capability to use the HSMs in the remote datacenter to cost effectively improve availability. In this mode, only the local units (non-standby) are used for active load-balancing. However, as key material is created they are automatically replicated to both the active (local) units and standby (remote) unit. In the event of a failure of all local members the standby unit is automatically promoted to active status.. The primary reason for using this feature is to reduce costs while improving reliability and this approach allows remote HSMs that have high latency to be avoided when not needed. However, in the worst case scenario where all the local HSMs fail, the remote member automatically activates itself and keeps the application running.



Notes and More

It is important that all members in an HA group have the same configuration and version. Running HA groups with different versions is unsupported. The Luna User Guide will help ensure that HSMs are configured identically to ensure smooth high availability and load balancing operation. Luna HSMs come with various key management configurations: cloning mode, key-export mode or SIM[12]-mode. HA functionality is supported with both cloning and SIM variants – provided all members in the group have the same configuration. Clients automatically and transparently use the correct secure key replication method based on the group's configuration.

[11] This issue is application specific. Applications that reuse previous names for new *versions* of a key may end up with both the old and new key replicated. Applications that use unique key names for every key may end up with an old (possibly deleted) key restored, but there will be no confusion as to which key is the correct key. Consult your application supplier.

[12] Secure Identity Management

It is also critical that all members in an HA group share the same Security Domain role (Red PED key for Trusted Path authentication devices and security domain password for password authenticated devices). The Security Domain defines which HSMs are allowed to share key material. Because HA group members are by definition intended to be peers they need to be in the same Security Domain.

By default the client library present both physical slots and virtual slots for the HA group. Directing applications at the physical slots will bypass the high availability and load balancing functionality. An application must be directed at the virtual slots to activate the high availability and load balancing functionality. There is a configuration setting referred to as "HAonly" that hides the physical slots. SafeNet recommends using this setting to prevent incorrect application configurations it also simplifies the PKCS #11 slot ordering given a dynamic HA group

Application developers should be aware that the PKCS #11 object handle model is fully virtualized with the Luna HA logic. As such, the application must not assume fixed handle numbers across instances of an application. A handle's value will remain consistent for the life of a process; but it may be a different value the next time the application is executed.

The network topography of the HA group is generally not important to the proper functioning of the group. As long as the client has a network path to each member the HA logic will function. Keep in mind that having a varying range of latencies between the client and each HA member will cause a command scheduling bias towards the low-latency members. It also implies that commands scheduled on the long-latency devices will have a larger overall latency associated with each command. In this case, the command latency is a characteristic of the network, to achieve uniform load distribution ensure that there are similar latencies to each device in the group (or use standby mode).

The Luna HA and load-balancing feature works on a per-client and per-partition bases. This provides a lot of flexibility. For example, it is possible to define a different sub-set of HSMs in each client and even in each client's partitions (in the event that a single client uses multiple partitions). SafeNet recommends to avoid these complex configurations and keeping the HA topography uniform for an entire HSM. That is, treat HSM members at the HSM level as atomic and whole. This will simplify the configuration management associated with the HA feature.

When a client is configured to use auto recovery the manual recovery commands must not be used. Invoking them can cause multiple concurrent recovery processes which will result in error codes and possible key corruption[13].

Most customers should enable auto-recovery in all configurations. The only reason a customer may wish to choose manual recovery is if the customer does not want to impart the retry time to periodic customer transactions. That is, each time a recovery is attempted a single application thread will experience an increased latency while the library uses that thread to attempt the re-connection (the latency impact is a few hundred milliseconds).

Example: Database Encryption

This section walks through a specific sample use case of some of the HA logic with a specific application – namely a transparent database encryption.

Typical Database Encryption Key Architecture

Database engines typically use a two-layered key architecture. At the top layer is a master encryption key that is the root of data protection. Losing this key is equivalent to losing the database, so it obviously needs to be highly durable. At the second layer are table keys used to protect table-spaces and/or columns. These table keys are stored with the database as blobs encrypted by the master encryption key. This architecture maps to the following operations on the HSM:

1. Initial generation of master key for each database.
2. Generation[14] and encryption of table keys with the master key.
3. Decrypting table keys when the database needs to access encrypted elements.

[13] This will be made more robust in a future release.
[14] Some databases do this in software.

4. Generation of new master keys during a re-key and then re-encrypting all table keys with it.
5. Generation and encryption of new table keys for storage in the database (often done in a software module).

The HSM is not involved in the use of table keys. Instead it provides the strong protection of the MEK which is used to protect the table keys. Users must follow backup procedures to ensure their MEK is as durable as the database itself. Refer to the HSM User Guide for proper backup procedures.

HSM High Availability with Database Encryption

When the HSMs are configured as a HA group the database's master key is automatically and transparently replicated to all the members when the key is created; and each time it is re-keyed.   If a HSM group member was offline or fails during the replication it will not immediately receive a copy of the key. Instead the HA group will proceed after replicating to all of the active members. Once a member is re-joined to the group the HSM client will automatically replicate the new master keys to the recovered member.

With this in mind, before every re-key event the user should ensure the HA group has sufficient redundancy. A re-key will succeed so long as one HA group member exists, but proceeding with too few HSMs will result in an availability risk. For example, proceeding with only one HSM means the new master key will be at risk since it exists only on a single HSM. Even with sufficient redundancy, SafeNet recommends maintaining an offline backup of a database's master key.

HSM Load Balancing with Database Encryption

While a database is up and running the master key exists on all members in the HA group. As such, requests to encrypt or decrypt table keys are distributed across the entire group. So the load-balancing feature is able to deliver improved performance and scalability when the database requires a large number of accesses to the table keys. That said most deployments will not need much load-balancing as the typical database deployment results in a small number of table keys.

While re-keying the table keys new keys are generated in the HSM and encrypted for storage in the database. Within a HA group, these keys will be generated on the primary HSM and then, even though they will exist on the HSM for only a moment, they are replicated to the entire HSM group as part of the availability logic.   These events are infrequent enough that this extra replication has minimal impact.

Conclusion

The Luna high availability and load balancing features provide an excellent set of tools to scale applications and manage availability of cryptographic services without compromising the integrity of cryptographic keys. They do not need to be copied out of an HSM and stored in a file to achieve high levels of availability. Indeed recovery from many failures is much more rapid with Luna's keys-in-hardware approach since each HSM maintains its own copy of all keys directly inside it. A broad range of deployment options are supported that allow solution architects to achieve the availability needed in a manner that optimizes the cost and performance without compromising the assurance of the solution.