SafeKit 7
Course Handbook
SafeKit 7
Course Handbook

Contents

Course Objective

This course handbook presents Evidian SafeKit, a 100% software based high availability solution. With this course handbook, you will be able:

- To understand the product
- To install, configure and test
- To administer and troubleshoot
- To support the solution

Contents

This handbook includes several slideshows and exercises:

- Introduction
  Introduction + Customers + Videos + Comparison
- Configuration
  Mirror Module + Farm Module + Checkers
- Setup
  Package + Web Management Console + Exercise: Setup and Tests
- Administration
  Command Line + Troubleshooting + Access to Support + Exercise: Advanced Configuration
- 3 Nodes Replication
  3nodesrepli module

This course handbook refers to the following guide:

# SafeKit Training Planning

<table>
<thead>
<tr>
<th>Modules</th>
<th>Pages</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Introduction</td>
<td>12p</td>
<td>0h20</td>
</tr>
<tr>
<td>2. Customers</td>
<td>10p</td>
<td>0h15</td>
</tr>
<tr>
<td>3. Videos</td>
<td>4p</td>
<td>0h30</td>
</tr>
<tr>
<td>4. Comparison – Distinctive Advantages</td>
<td>8p</td>
<td>0h15</td>
</tr>
<tr>
<td><strong>Configuration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Mirror Module Configuration</td>
<td>34p</td>
<td>1h</td>
</tr>
<tr>
<td>6. Farm Module Configuration</td>
<td>16p</td>
<td>0h30</td>
</tr>
<tr>
<td>7. Checkers Configuration for Mirror and Farm Modules</td>
<td>18p</td>
<td>0h30</td>
</tr>
<tr>
<td><strong>Setup</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Package Installation</td>
<td>8p</td>
<td>0h15</td>
</tr>
<tr>
<td>9. Web Management Console</td>
<td>30p</td>
<td>0h55</td>
</tr>
<tr>
<td>10. Exercise - Setup and Tests</td>
<td>2p</td>
<td>2h</td>
</tr>
<tr>
<td><strong>Administration</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Command Line Interface</td>
<td>24p</td>
<td>0h40</td>
</tr>
<tr>
<td>12. Problems - Troubleshooting</td>
<td>16p</td>
<td>0h30</td>
</tr>
<tr>
<td>13. Access to Evidian Support</td>
<td>14p</td>
<td>0h30</td>
</tr>
<tr>
<td>14. Exercise - Advanced Configuration</td>
<td>6p</td>
<td>2h</td>
</tr>
<tr>
<td><strong>Option</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15. 3 Nodes Replication</td>
<td>22p</td>
<td>0h30</td>
</tr>
</tbody>
</table>

## Class and Exercises

<table>
<thead>
<tr>
<th>Modules</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1+2+3+4) + (5+6+7) + (8+9+10) + (11+12+13+14)</td>
<td>9h50</td>
</tr>
</tbody>
</table>

## Class only

<table>
<thead>
<tr>
<th>Modules</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1+2+3+4) + (5+6+7) + (8+9) + (11+12+13)</td>
<td>5h50</td>
</tr>
</tbody>
</table>

## Option Class

<table>
<thead>
<tr>
<th>Modules</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15)</td>
<td>0h30</td>
</tr>
</tbody>
</table>

## Documentation

Evidian SafeKit
Simple and Economical High Availability Software

SafeKit Product Line Manager - bruno.rochat@evidian.com

SafeKit inside Evidian

- Govern, manage and control user identities and access rights
- Manage strong and business-driven authentication
- Secure, policy-based authorization, cloud access, Web SSO and federation
- Secure access to applications from PC, tablets and mobiles
- Sustainable compliance, risk analysis and advanced analytics for Identity and Access
- Analytics and Intelligence
- High-end Directory Server for enterprise and e-Business environments
- Directory Server
- High availability with load balancing, replication and failover
- Business Continuity
- Identity Governance and Administration
- Authentication Management
- Web Access Management
- Enterprise SSO
- Analytics and Intelligence

- http://www.evidian.com/
SafeKit Software High Availability

Business Continuity
Application essentials
24x7

File Replication
mirror

SafeKit

Load Balancing
farm

Fail-Over
mirror farm

Air Traffic Control
Media Broadcasting
Video Surveillance
Banking
...

use cases

Robust   Cost efficient   Easy   Ideal for Partners

free trial, demos for web services, databases on Windows, Linux

https://www.evidian.com/products/high-availability-software-for-application-clustering/cluster-configuration/


https://www.evidian.com/products/high-availability-software-for-application-clustering/#t3

IdS-SK-A 01 Rev18
© Evidian

SafeKit Training
Introduction
The Two Basic Architectures

<table>
<thead>
<tr>
<th>The farm cluster</th>
<th>The mirror cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network load balancing and failover</td>
<td>Real-time replication and failover</td>
</tr>
</tbody>
</table>

- **The farm cluster**
  - Network load balancing and failover
  - **The mirror cluster**
  - Real-time replication and failover

- **Mirror demo**
- **Farm demo**

Advanced Architectures

Multiple mirror modules
Each server is active

1 farm module + 1 mirror module
3-tiers architecture

Architecture of Evidian Web Access Manager

Hyper-V Cluster with SafeKit

Real-time replication, failover, load balancing of full virtual machines

► 2 Windows Hyper-V 2012 R2 physical servers
► 1 mirror module = replication and failover of 1 full VM
► N mirror modules for N VMs (limited to 25 VMs)
► Active-active cluster: VMs are distributed on the two servers
► A centralized web console to manage failover of all VMs

3-Nodes Replication

A Simple Integration Process

<table>
<thead>
<tr>
<th>1 - Design</th>
<th>2 - Integration</th>
<th>3 - Deployment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Farm module</td>
<td>Write 3 files per module</td>
<td>Deploy a module on standard servers</td>
</tr>
<tr>
<td>UP UP UP</td>
<td>user config.xml</td>
<td>Deploy a module on standard servers</td>
</tr>
<tr>
<td>Mirror module</td>
<td>start script</td>
<td>Deploy a module on standard servers</td>
</tr>
<tr>
<td>PRIM SECOND</td>
<td>stop script</td>
<td>Deploy a module on standard servers</td>
</tr>
<tr>
<td>Choose a module according application</td>
<td></td>
<td>Deploy a module on standard servers</td>
</tr>
</tbody>
</table>

Free trial on evidian.com

## Save Costs: 3 Products in 1

<table>
<thead>
<tr>
<th>Hardware solutions</th>
<th>Our 3 in 1 software product</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Network load balancing boxes</td>
<td>Simplify</td>
</tr>
<tr>
<td>2. Replicated storage</td>
<td><img src="" alt="SafeKit" /></td>
</tr>
<tr>
<td>3. Enterprise editions for failover</td>
<td>Save costs</td>
</tr>
</tbody>
</table>

- existing standard servers
- no additional hardware
- standard editions of Windows and Unix
- standard editions of database

Our Business Model for SafeKit

- Reseller of SafeKit
- Reseller
- Make SafeKit integration with their applications/projects
- Software publishers
- System Integrators
- Receive a turn-key high availability solution
- Customer
- Customer
- Customer

All materials are on web to address the market (white papers, free trial, user's guide, training)

- http://www.evidian.com/products/high-availability-software-for-application-clustering/#t2
The 3 Best Use Cases of SafeKit

- **Software Publisher**: High availability OEM software solution
- **Enterprise**: No specific IT skills to deploy software cluster
- **Data Center**: Disaster recovery solution without replicated SAN

Conclusion – The key messages

- SafeKit is a very simple high availability product on the market
- Only SafeKit brings load balancing + replication + failover
- Zero hardware extra cost (replicated SAN, network boxes)
- Zero software extra cost (Enterprise editions of OS and DB)
- Zero specific skill for the deployment
- The ideal product for a software publisher
- Ready for the Cloud – Amazon AWS, Microsoft Azure, Google Cloud
- All materials are on web: free trial, user’s guide, training

http://www.evidian.com/products/high-availability-software-for-application-clustering/
Harmonic
- Specialist of broadcasting and video equipments
- Customers = TV operators
- + 80 clusters deployed worldwide (plug&play deployment)

Viaccess
- + 45 Linux clusters worldwide
- Plug&play deployment of HA solutions

"SafeKit is the ideal application clustering solution for a software publisher. We have deployed more than 80 SafeKit clusters worldwide."

A large market for SafeKit with products:

- Milestone
- Hanwha SSM (ex Samsung)
- Siemens SiPass
- …

Examples:

- Prisons, Stadiums, Buildings in France, UK, Scotland, Middle East, South Korea, Morocco…
- +140 licenses worldwide

“SafeKit is appreciated because the product is easy to install and very quickly deployed.

The SSM application does not need to be modified.

It can be installed on the default C: drive and there is no need to configure a separate disk volume.”

Supervision of Paris metro line 1

- 20 Windows and Linux clusters (prod + preprod)
- protect front-end applications with web services (farm cluster)
- protect back-end applications with database (mirror cluster)
- Windows and Linux: a single solution
- servers in 2 remote rooms for disaster recovery

"This clustering solution is homogeneous for our Windows and Unix platforms.
SafeKit provides the three functions that we needed: load balancing between servers, automatic failover after an incident and real time data replication."

Air Traffic Control (ATC)
- Copperchase – Software editor for ATC
- 32 airports worldwide with SafeKit
- 32 Windows clusters

And also
- DFS (German ATC): 27 Linux clusters
- ADB SafeGate: + 14 Windows clusters

“We have found with SafeKit a simple and complete clustering solution for our needs. With load balancing, real time data replication (no data loss) and automatic failover.”

Defense
- ERP of French army
- 10 billion Euros of orders to industry
- 14,000 employees
- Deploying tens of SafeKit clusters does not pose problems

“Our production team implemented the SafeKit solution without any difficulty on 14 Windows and Linux clusters”

Emergency phone calls

► + 40 call centers with Systel

"SafeKit perfectly meets the needs of a software vendor. It brings in high availability through a software option that is added to our own software suite."

ERP for Industry

- Fives Syleps
- ERP Sydel, +20 enterprises, agri-business
- Deployment on standard servers without specific equipment
- Supports servers in remote computer rooms

And also

- Alstef, Stockware, Bagware in industry and airports
- +30 Linux clusters

“It is not conceivable that our ERP is out of service due to a computer failure.
Otherwise the whole business activity of the Enterprise stops.”

SafeKit is a powerful product that monitors our business applications and restarts them in case of hardware or software failure.

Natixis

- + 30 clusters Windows, AIX, Linux
- With the Axway financial exchange gateway (EAI & EDI)
- And with other critical financial applications

And also

- Credit union software, + 20 banks, with Wellington IT (Ireland, UK)

Demonstration of a mirror module

Demonstration of a farm module

TCP sessions are load balanced between node 1 and node 2

Demonstration of the Hyper-V module

Evidian SafeKit

Comparison

Distinctive Advantages
Hardware Cluster vs SafeKit

Hardware cluster

- Special hardware with bay of disks
- Complex deployment - need for IT skills
- Servers in remote sites complex and expensive

Software cluster

- No special hardware
- Plug-and-play deployment
- Servers in remote sites simple - no extra cost

Cluster of Virtual Machines vs Application Cluster

Cluster of virtual machines

- VMware HA
- Hyper-V Cluster
- Reboot of OS on server 2 if crash of server 1
- Risk of downtime when fixing OS or App
- Shared disk to configure
- Servers in remote sites complex and expensive

Application cluster

- Quick restart of App on OS2 if crash of OS1
- Smooth upgrade of OS1 or OS2 or App
- No shared disk
- Servers in remote sites simple - no extra cost

© Evidian

Hyper-V cluster vs VMware HA vs SafeKit VM HA

Hyper-V Cluster & VMware HA

- Shared disk and specific bay of disk
- Remote sites = replicated bays across a SAN
- Enterprise Editions of Windows or VMware
- Specific IT skills to configure the system
- Large number of VMs

SafeKit VM HA

- No shared disk
- Remote sites = no replicated SAN
- Enterprise Editions of Windows or VMware
- No specific IT skill to configure the system
- Limited number of VMs (25 VMs)

Hyper-V 2012 R2 / 2016

OS
App
VM

OS
App
VM

OS
App
VM

OS
App
VM replica

Asynchronous vs Synchronous File Replication

Asynchronous replication

Double-Take - do not wait acknowledgement of replicated IO. Data buffer is lost in case of failure

Data loss on failure with asynchronous replication

Synchronous replication

SafeKit - wait acknowledgement of replicated IO

No data loss on failure with synchronous replication

Disk Replication vs File Replication

**Block-level disk replication**

- Data must be isolated in a special disk and file system

<table>
<thead>
<tr>
<th>☹ Replication of a full disk</th>
<th>☹ Replication only of selected directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>☹ Prerequisite on disk organization</td>
<td>☹ No prerequisites on disk organization</td>
</tr>
<tr>
<td>☹ Need for IT skills to configure disks</td>
<td>☹ No need for IT skills</td>
</tr>
<tr>
<td>☹ Complex deployment</td>
<td>☹ Plug-and-play deployment</td>
</tr>
</tbody>
</table>

**Byte-level file replication**

- Data can be inside a disk

<table>
<thead>
<tr>
<th>☺ Replication of a full disk</th>
<th>☺ Replication only of selected directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>☺ No prerequisite on disk organization</td>
<td>☺ No need for IT skills</td>
</tr>
<tr>
<td>☺ No need for IT skills</td>
<td>☺ Plug-and-play deployment</td>
</tr>
</tbody>
</table>

Hardware Load Balancer vs SafeKit

Hardware load balancing

External network boxes

Network load balancers
Dedicated servers for load balancing

Software load balancing

SafeKit - Installation on the application server side
No network box, no dedicated server for load balancing

😊 Load balancing OK
😊 Automatic restart of applications OK
😊 File replication OK

😊 Load balancing OK
😊 Automatic failover and failback OK
😊 Real time file replication OK

Note: Contrary to Microsoft NLB, SafeKit load balancing is implemented in VMware without special network configuration

Fault Tolerant Cluster vs SafeKit

**Fault tolerant cluster**

- Application executing the same thing at the same time on both servers
- A software exception will happen on both servers at the same time
- CPU usage on both servers for the same application

**Software cluster**

- SafeKit - Application is restarted in case of failure
- Software exception recovery
- Application restarted in a different OS environment
Evidian SafeKit

Mirror Module Configuration
Mirror Module Overview

Mirror module
Real time replication and failover

Application

Console

PRIM  SECOND
STOP  ALONE
Reintegration  PRIM
SECOND  PRIM

ip1@ / app1
files1

ip1@ / app1
files1

ip1@ / app1
files1

Video - SQL Server
Templates

Generic template for integrating an application in a mirror module

- mirror.safe for Linux and Windows  Generic template

Demonstration templates

- sqlserver.safe for Windows  Microsoft SQL Server integration
- mysql.safe for Linux  MySQL integration
- oracle.safe for Windows and Linux  Oracle integration
- milestone.safe for Windows  Milestone integration
- hyperv.safe for Windows  Hyper-V integration

Templates are under SAFE/Application_Modules/ and
https://support.evidian.com/solutions/downloads/safekit/Version_7.x/
Application_Modules/
Internal Files of a Mirror Module

- **mirror**
  - **conf**
    - **userconfig.xml** User XML configuration file
    - **userconfig.xml.template** Internal use only
  - **bin**
    - **prestart** User script executed on module start
    - **start_prim** User script to start the application on the primary
    - **stop_prim** User script to stop the application
  - **poststop** User script executed on module stop
  - **web**
    - **index.html** Configuration file of the web console
## Mirror Module Configuration

### Configuration in userconfig.xml

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;heart&gt;</code></td>
<td>Synchronize failover through monitoring networks (heartbeats)</td>
</tr>
<tr>
<td><code>&lt;vip&gt;</code></td>
<td>Set a virtual IP address on the primary</td>
</tr>
<tr>
<td><code>&lt;user&gt;</code></td>
<td>Enable user scripts to start/stop the application</td>
</tr>
<tr>
<td><code>&lt;vhost&gt;</code></td>
<td>Set a virtual hostname</td>
</tr>
<tr>
<td><code>&lt;rfs&gt;</code></td>
<td>Configure directories to replicate</td>
</tr>
<tr>
<td><code>&lt;errd&gt;, &lt;check&gt;, &lt;failover&gt;</code></td>
<td>See « Checkers Configuration » slides</td>
</tr>
</tbody>
</table>

### Setting encryption of internal communications for the module

```bash
safekit module genkey -m <module name>
```

Details of configuration are in the User’s Guide
Overview of cluster.xml for the SafeKit cluster

- Definition of all the SafeKit servers that belong to the same cluster and that implement one or more modules
- Definition of networks used for the internal communications and by the web console
- Abstraction of the network topology for configuring modules

Example of cluster.xml

```xml
<cluster>
  <lans>
    <lan name="default" framework="on" console="on"> <!-- configuration of the main network that must define all the cluster nodes -->
      <node name="server1" addr="172.24.199.107"/>
      <node name="server2" addr="172.24.199.108"/>
      <node name="server3" addr="172.24.199.109"/>
    </lan>
    <lan name="private" framework="on" console="off"> <!-- configuration of other networks -->
      <node name="server1" addr="10.0.0.107"/>
      <node name="server2" addr="10.0.0.108"/>
    </lan>
  </lans>
</cluster>
```
Overview of userconfig.xml for a Mirror Module

```
<service mode="mirror">
  <heartbeat>
    <heartbeat name="LAN_NAME_TO_BE_DEFINED" />
  </heartbeat>
  <vip>
    <interface_list>
      <real_interface>
        <virtual_addr addr="IP_VIRTUAL_TO_BE_DEFINED" where="one_side_alias" check="on"/>
      </real_interface>
    </interface_list>
  </vip>
  <rfs>
    <replicated dir="DIRECTORY_PATH_TO_BE_DEFINED" mode="read_only"/>
  </rfs>
  <user/>
</service>
```

← Monitoring Network Configuration -->
← name of the network that connects the 2 servers (name set into cluster.xml) -->

← Virtual IP Configuration -->

← File Replication Configuration -->

← User Scripts Configuration -->
Heartbeat Overview (1/7)

- heartbeat between 2 servers to synchronize states and actions
- automatic failover when all heartbeats are lost
- main states of a module on a server:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALONE</td>
<td>Module is primary without secondary and has the up-to-date replicated files. Application is started on ALONE.</td>
</tr>
<tr>
<td>PRIM</td>
<td>Module is primary with a secondary and replicated files are mirrored on the secondary. Application is started on PRIM.</td>
</tr>
<tr>
<td>SECOND</td>
<td>Module is secondary with a primary and replicated files are mirrored from the primary.</td>
</tr>
<tr>
<td>WAIT</td>
<td>Module waits for the start of the remote server because its files are not up-to-date. Application is stopped (no remote ALONE server).</td>
</tr>
</tbody>
</table>
Server 1 (PRIM) stops or fails

- Server 2 goes ALONE because server 1 stops
- Server 1 may be stopped by an administrator or by a checker

Server 1 (PRIM) fails

- Server 2 goes ALONE because all heartbeats are lost
- It may be a crash of server 1
- It may be a network isolation between server 1 and server 2
- Multiple heartbeats are better to avoid split brain

Restart of application on server 2 ALONE
Server 1

Server 2 (SECOND) stops or fails

- Server 2 goes ALONE because server 2 stops
- Server 2 may be stopped by an administrator or by a checker

Server 2 (SECOND) stops

- Server 1 goes ALONE because all heartbeats are lost
- It may be a crash of server 2
- It may be a network isolation between server 1 and server 2
- Multiple heartbeats are better to avoid split brain

Server 2 (SECOND) fails

Application is not stopped and continues to run on server 1 ALONE
**Heartbeat - Restart (4/7)**

**Swap server 1 - server 2**

- **swap** from an administrator for reversing PRIM-SECOND roles
- restart from an administrator or from a checker, for restarting the application locally on the primary

**Restart on primary server**

- **swap** restart from an administrator or from a checker, for restarting the application locally on the primary

- **stop application on server 1**
- **start application on server 2**
Start-up of the Primary as ALONE (5/7)

- **start** - start from an administrator or at boot
- **stop** - stop from an administrator or from a checker
- **wait** - wait when a resource is set to down by a checker
- **wakeup** - wakeup when a resource is set to up by a checker

- **STOP** (red) - module and application stopped, ready for starting
- **WAIT** (magenta) - module running user scripts **prestart** or **poststop**
- **ALONE** (magenta) - module running user scripts **start_prim** or **stop_prim** to start/stop the application
- **ALONE** (green) - module and application started
- **WAIT** (red) - module waiting for a mandatory resource. Application is stopped
Heartbeat - Start-up as Secondary (6/7)

Server 2: not up to date
- **WAIT (magenta)**: module running user scripts `prestart` or `poststop`
- **WAIT (red)**: replicated files are not up-to-date. Module waits for the start of the other server for reintegrating its files
- **WAIT (red)**: module waits for a mandatory resource
- **SECOND (magenta)**: secondary is reintegrating files from the primary. If necessary, there are two user scripts `start_second` and `stop_second`

Server 1:
- **ALONE (green)**
- **PRIM (green)**

**Start**

**Wait**
- **start**
- **prestart**
- **poststop**
- **stop**
- **uptodate**
- **wait**
- **wakeup**
The network name is the name set into the SafeKit cluster configuration file (cluster.xml). The IP addresses of the servers that implement the module are dynamically retrieved based on the content of cluster.xml and on servers where the module is configured.

heartbeat ident is used in the failover rules (optional)

as many <heartbeat> tags as network connections to use between servers (at least 2 to avoid split brain)

<heartbeat>
  <heartbeat name="network name" [ident="name of the heartbeat"] />
  … as many <heartbeat> tags as network connections to use between servers (at least 2 to avoid split brain)
</heartbeat>
Virtual IP @ associated with the MAC @ of primary server

- ARP clients associate the virtual IP @ with the mac address of the primary server
- virtual IP address is automatically configured on primary server (alias)
- in case of failover, arpreroute daemon reroutes IP-v4 clients with gratuitous ARP to mac2 @
- virtual IP address works with Ethernet interfaces in teaming, bonding, VLAN
VIP - Configuration in userconfig.xml (2/2)

userconfig.xml

```xml
<vip>
  <interface check="on" arpreroute="on">
    <real_interface>
      <virtual_addr addr="192.168.1.50" where="one_side_alias" check="on"/>
      ...as many <virtual_addr> tags as there are virtual @ to configure on 192.168... network
    </real_interface>
  </interface>
</vip>
```

- Puts the module in 🔄 WAIT (red) state until the interface is repaired.
- Reroutes IP-v4 clients with gratuitous ARP.
- Detects virtual IP address conflict and address removal. Runs stopstart on the faulty module to return to normal operation in the cluster.

IP-v4 or IP-v6 address configured as an alias on the primary server.

... as many <interface> tags as there are network interfaces to configure with virtual IP addresses.
Prerequisites for application integration

1. Application is installed and can start on both servers
2. Application offers a script interface to start it and stop it
3. Automatic start of the application at boot has been removed (will be replaced by automatic start of the module at boot `safekit boot -m <module name> on`)

Integration in user scripts `start_prim` and `stop_prim`

- On Linux, start/stop application in its environment: `su -user "appli-cmd"
- On Windows, start/stop application services: `net start|stop "service"`
## Edit User Scripts (2/3)

There are 2 logs per module:
- the stdout/stderr log of scripts (`start_prim`, `stop_prim`)
- the event log inside the module

<table>
<thead>
<tr>
<th><code>start_prim</code></th>
<th><code>stop_prim</code></th>
</tr>
</thead>
</table>
| rem echo in the *stdout log*  
echo "Running start_prim %*"  
rem Start your application here  
net start...  
rem %res% = result of start-up  
if %res% == 0 goto end  
rem message in the *event log*  
%SAFE%\safekit printe "start_prim failed"  
:end | rem echo in the *stdout log*  
echo "Running stop_prim %*"  
rem option force means forcestop  
if "%1" == "force" goto end  
rem Stop your application here  
net stop...  
rem net stop is not synchronous...  
%SAFEBIN%\sleep 10  
:end |

**Restart command on a module run** `stop_prim` and then `start_prim` - thus, it is better to have `stop_prim` waiting for stop of services.
Scripts - Configuration in userconfig.xml (3/3)

userconfig.xml

```xml
<user nicestoptimeout="300"> [forcestoptimeout="300"] [userlogsize="2048"]>
  <var name="MYVAR" value="MYVALUE"/>
  ... as many environment variables <var> as needed
</user>
```

**userlogsize**
- size of the stdout/stderr user scripts log (by default 2048 KB)
- log stored in SAFEVAR/modules/<module name>/userlog.ulog
- reset of stdout/stderr log size when it is superior to **userlogsize**

Predefined environment variables passed to user scripts
- **SAFE**: root installation directory
- **SAFEUSERBIN**: user scripts execution directory of the module
- **SAFEUSERVAR**: variables directory of the module
- **SAFEMODULE**: module name (safekit commands do no require –m <module name> parameter in user scripts)
Virtual Hostname Configuration

**userconfig.xml**

```xml
<vhost>
  <virtualhostname
    name="virtualname"
    when="prim"
    envfile="vhostenv"
  />
</vhost>

virtual hostname = virtualname
activate the virtual hostname when the server is primary
environment file generated by SafeKit to be sourced in start_prim

See vhost.safe template for a complete example, including Windows services

**start_prim**

# Set virtual hostname on Linux
. $SAFEUSERBIN/vhostenv
rem Set virtual hostname on Windows
CALL %SAFEUSERBIN%\vhostenv.cmd
...here, hostname=virtualname
Replication on Linux - How It Works? (1/4)

Linux Application

write

File system

NFS client

nfsbox

nfsbox

ok1

ok2

Synchronous replication: no data loss

acknowledge write when ok1+ok2

Performance?
Works as a remote NAS

Video - Synchronous vs asynchronous replication
Replication on Windows - How It Works? (2/4)

Windows Application

write

File system
RFS mini-filter

acknowledge write when ok1+ok2

Synchronous replication: no data loss

File system
RFS mini-filter

PRIM

SECOND

green

Performances?
Works as a remote NAS

nfsbox

ok1

File system
RFS mini-filter

nfsbox

ok2

Video - Synchronous vs asynchronous replication
Asynchronous Write on Secondary (3/4)

Windows/Linux Application

write

File system
RFS/NFS

acknowledge write when ok1+ok2

write

nfsbox

green
PRIM

green
SECOND

ok1

ok2

Asynchronous on secondary

Write to disk will be made later

acknowledge write when ok1+ok2
Special Case - ALONE server (4/4)

- **Linux Application**
  - read/write
  - NFS client
  - NFSbox

- **Windows Application**
  - read/write
  - RFS mini-filter

**Linux file system**

**Windows file system**

**ALONE**

**green**

**nfsbox**
Reintegration - How It Works? (1/2)

Server 2 is stopped - its data must be reintegrated from server 1

**Phase 1 - reintegration**
- update of trees under replicated directories

**Phase 2 - reintegration**
- if the bitmaps are safe, copy modifications set in bitmaps during the stop
- if the bitmaps are not safe, copy files modified during the stop

To have a safe reintegration in case of crash, reintegration takes also files modified before the crash according to a grace period of about one hour.
Reintegration policy

- the module was properly stopped on the server: reintegration per bitmap

- primary or secondary crashes (ex.: power off) or nfsbox replication process stops on exception: full copy of files modified during the stop (+ grace period)

- files have been modified on primary or secondary while the module was stopped: full copy of all replicated directories
Server 1

- Server 1 is started - it will become primary ALONE
- Replicated directories are up
date on server 1

- Linux
  - start-up of NFS client and start-up of nfsbox
  - NFS mount on 127.0.0.1 of replicated directories
- Windows
  - activation of RFS file system mini-filter and start-up of nfsbox

- Linux
  - nfsbox intercepts access and follows them to local file system
- Windows
  - RFS mini-filter intercepts access and follows them to local file system

- Server 1 is primary without secondary and has the up-to-date files
- Server 1 does not replicate write requests

Server 1
Replication - Start-up as SECOND (2/3)

Server 2

- Server 2 is started - it will become SECOND
- Replicated data are not up to date

- Linux
  - start-up of NFS client and start-up of nfsbox
  - NFS mount on 127.0.0.1 of replicated directories

- Windows
  - activation of RFS file system mini-filter and start-up of nfsbox
  - ❗️ WAIT (red) (not up to date) : waits for start-up of the other server

Server 2

- Server 2 reintegrates modified files from server 1
- Server 1 is ✅ ALONE (green) and replicates write

Linux

- NFS mount on 127.0.0.1 of replicated directories
- RFS file system mini-filter and start-up of nfsbox

- Windows

Server 2 is SECOND

- Server 2 becomes secondary and its data are up to date
- Server 1 becomes ✅ PRIM (green)
- Data are replicated from primary to secondary
- The application availability is more critical than the application data
- Server 2 is started - it will become ALONE
- Replicated data are not uptodate
- Server 1 is uptodate but not responding

- Default start and failover policy is relaxed
- `syncdelta` configuration in `<rfs>` allows the not uptodate server to become primary but only if the elapsed time since the last synchronization is `<syncdelta` (in min)

- Server 2 becomes secondary and its data are uptodate

- When Server1 gets back, it will reintegrate modified files from Server 2 and become SECOND
Replication Configuration Prerequisites (1/4)

Prerequisites

1. Replicated directories at the same location on both servers
2. On Linux, same uid/gid on both servers for replicated files
3. Align clocks of both servers (NTP)

First start-up of a module: synchronize data in the right direction

- Start the server with the uptodate replicated directories as primary with:
  safekit prim -m AM
- Start the other server as secondary with:
  safekit second -m AM
Replication Configuration in userconfig.xml (2/4)

```xml
userconfig.xml

  <rfs>
    <replicated dir "/safedir" mode="read_only">
      <notreplicated path="/safedir/file1" />
      <notreplicated path="/safedir/subdir1/subdir2" />
      <notreplicated regexpath=".*\.tmp" />
    </replicated>
  ...
  </rfs>
```

- absolute path of the directory to replicate
- read-only access on the secondary to avoid corruption
- as many `<replicated>` tags as there are directories to replicate
- notreplicated
  - the file "/safedir/file1" is not replicated
  - the subdirectory "/safedir/subdir1/subdir2" is not replicated
  - all ".tmp" files in "/safedir" and its subdirectories are not replicated (regular expression on Linux only)
Configure a dedicated replication network (ex.: crossed Ethernet cable)

- Add the network into the SafeKit cluster configuration (cluster.xml)
- Add a heartbeat with `ident="flow"` for this network (the network is used for both replication and for the heartbeat)

```xml
<heartbeat name="network name" ident="flow"/>
```

- If no heartbeat with `ident="flow"`, replication network is the first configured heartbeat
Replication Advanced Configuration (4/4)

**userconfig.xml**

```xml
<rfs [acl="on"|"off"] [async="none"|"second"]
 [nbrei="3"] [reitimeout="150"]
 [reiallowedbw="20000"] [syncdelta="nb minutes"] >
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl=&quot;on&quot;</td>
<td>&quot;off&quot;</td>
</tr>
<tr>
<td>nbrei=&quot;3&quot;</td>
<td>Number of reintegration threads and timeout in seconds for running reintegration requests (adapt on big replicated tree or heavy load)</td>
</tr>
<tr>
<td>reitimeout=&quot;150&quot;</td>
<td>Default value &quot;second&quot;: asynchronous writes on the secondary server</td>
</tr>
<tr>
<td>async=&quot;none&quot;</td>
<td>&quot;second&quot;</td>
</tr>
<tr>
<td>reiallowedbw=&quot;20000&quot;</td>
<td>When &gt; 1, it allows a not up to date server to run as primary or a failover when the up to date server is not responding but only if the elapsed time, in minute, since the last synchronization is lower than the syncdelta value</td>
</tr>
</tbody>
</table>
Evidian SafeKit

Farm Module Configuration
Farm Module Overview

Farm module
Network load balancing and failover

Application

Console

Video - Apache farm
Templates

Generic template for integrating an application in a farm module

- farm.safe for Linux and Windows  
  generic template

Demonstration templates

- apache_farm.safe for Linux and Windows  
  Apache integration
- iis_farm.safe for Windows  
  Microsoft IIS integration

Templates are under SAFE/Application_Modules/ and
https://support.evidian.com/solutions/downloads/safekit/Version_7.x/Application_Modules/
### Internal Files of a Farm Module

**farm**

- **conf**
  - userconfig.xml: User XML configuration file
  - userconfig.xml.template: Internal use only

- **bin**
  - prestart: User script executed on module start
  - start_both: User script to start the application on all servers
  - stop_both: User script to stop the application
  - poststop: User script executed on module stop

- **web**
  - index.html: Configuration file of the web console

---

**Internal Files of a Farm Module**

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>farm</td>
<td></td>
</tr>
<tr>
<td>conf</td>
<td></td>
</tr>
<tr>
<td>userconfig.xml</td>
<td>User XML configuration file</td>
</tr>
<tr>
<td>userconfig.xml.template</td>
<td>Internal use only</td>
</tr>
<tr>
<td>bin</td>
<td></td>
</tr>
<tr>
<td>prestart</td>
<td>User script executed on module start</td>
</tr>
<tr>
<td>start_both</td>
<td>User script to start the application on all servers</td>
</tr>
<tr>
<td>stop_both</td>
<td>User script to stop the application</td>
</tr>
<tr>
<td>poststop</td>
<td>User script executed on module stop</td>
</tr>
<tr>
<td>web</td>
<td></td>
</tr>
<tr>
<td>index.html</td>
<td>Configuration file of the web console</td>
</tr>
</tbody>
</table>
### Farm Module Configuration

#### Configuration in userconfig.xml

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;farm&gt;</td>
<td>Networks used by the farm for synchronizing servers</td>
</tr>
<tr>
<td>&lt;vip&gt;</td>
<td>Configure virtual IP address and load-balancing rules</td>
</tr>
<tr>
<td>&lt;user&gt;</td>
<td>Enable user scripts to start/stop the application</td>
</tr>
</tbody>
</table>

<i>errd>, <check>, <failover>  See « Checkers Configuration » slides</i>

### Setting encryption of internal communications for the module

```
safekit module genkey -m <module name>
```

Details of configuration are in User’s Guide.
Overview of cluster.xml for the SafeKit cluster

- Definition of all the SafeKit servers that belong to the same cluster and that implement one or more modules
- Definition of networks used for the internal communications and by the web console
- Abstraction of the network topology for configuring modules

Example of cluster.xml

```xml
<cluster>
  <lans>
    <lan name="default" framework="on" console="on">
      <!-- configuration of the main network that must define all the cluster nodes -->
      <node name="server1" addr="172.24.199.107"/>
      <node name="server2" addr="172.24.199.108"/>
      <node name="server3" addr="172.24.199.109"/>
    </lan>
    <lan name="private" framework="on" console="off">
      <!-- configuration of other networks -->
      <node name="server1" addr="10.0.0.107"/>
      <node name="server2" addr="10.0.0.108"/>
    </lan>
  </lans>
</cluster>
```
Overview of userconfig.xml for a Farm Module

```xml
<service mode="farm">
    <farm>
        <!-- Definition of the networks used -->
        <lan name="LAN_NAME_TO_BE_DEFINED"/>
        <!-- name of the network that connects the servers (name set into cluster.xml) -->
    </farm>

    <vip>
        <!-- Virtual IP Configuration -->
        <interface_list>
            <interface check="off" arpreroute="on" arpelapse="60" arpininterval="5">
                <virtual_interface type="vmac_directed">
                    <virtual_addr addr="VIRTUAL_IP_ADDR_TO_BE_DEFINED" where="alias" check="on"/>
                </virtual_interface>
            </interface>
        </interface_list>

        <loadbalancing_list>
            <!-- Load Balancing Configuration -->
            <group name="FarmProto">
                <rule port="PORT_TO_BE_DEFINED" proto="tcp" filter="on_addr"/>
            </group>
        </loadbalancing_list>

    </vip>

    <user/>

</service>
```
Start-Stop-Restart of a Farm Module

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>start</strong></td>
<td>start from an administrator or at boot</td>
</tr>
<tr>
<td><strong>stop</strong></td>
<td>stop from an administrator or from a checker</td>
</tr>
<tr>
<td><strong>restart</strong></td>
<td>restart from an administrator or from a checker. application is restarted</td>
</tr>
<tr>
<td><strong>wait</strong></td>
<td>wait when resource is set to down by a checker</td>
</tr>
<tr>
<td><strong>wakeup</strong></td>
<td>wakeup when resource is set to up by a checker</td>
</tr>
<tr>
<td><strong>STOP</strong></td>
<td>module and application stopped, ready for starting</td>
</tr>
<tr>
<td><strong>UP</strong></td>
<td>module running user scripts prestart or poststop</td>
</tr>
<tr>
<td><strong>UP</strong></td>
<td>module running user application scripts start_both or stop_both</td>
</tr>
<tr>
<td><strong>WAIT</strong></td>
<td>module and application started</td>
</tr>
<tr>
<td><strong>WAIT</strong></td>
<td>module waiting for a mandatory resource. application is stopped</td>
</tr>
</tbody>
</table>

Note: The diagram illustrates the states and transitions of a farm module, showing how the module and application are started, stopped, and restarted in response to various status changes and administrator actions.
The network name is the name set into the SafeKit cluster configuration file (cluster.xml). The IP addresses of the servers that implement the module are dynamically retrieved based on the content of cluster.xml and on servers where the module is configured.
VIP - Virtual IP Address in a Farm (1/4)

Virtual IP @ associated with a MAC @

- the virtual IP @ is configured on all servers of the farm (alias)
- the vip driver is performing load balancing by accepting or forwarding incoming packets
- only one vip driver accepts the packet according to the hash function
- hash tables in all vip drivers are coherent thanks to the synchronization protocol
- no need of promiscuous mode on Ethernet card and switches broadcast

CLIENT 1
vip @ at mac1 @

CLIENT 2
vip @ at mac2 @

Application

VIP - Virtual IP

Virtual IP Address

Address in a Farm (1/4)

VIP - Virtual IP Address in a Farm (1/4)

Virtual IP Address

Address in a Farm (1/4)
VIP - Virtual IP Configuration (2/4)

```
userconfig.xml

<vip>
  <interface check="on" arpreroute="on">
    <virtual_interface type="vmac_directed">
      <virtual_addr addr="192.168.1.50" where="alias" check="on" />
      ...as many <virtual_addr> tags as there are virtual @ to configure on 192.168... network
    </virtual_interface>
    </interface>
  </vip>

... as many <interface> tags as there are network interfaces to configure with virtual IP addresses
```

- puts the module in ✋ WAIT (red) state until the interface is repaired
- When farm membership change, arp reconfiguration packets are sent.
- The mac address used to exchange packets with the farm is one of the farm network card's address
- virtual IP-v4 or IP-v6 address configured as an alias on all servers of the farm
- When farm membership change, arp reconfiguration packets are sent.
- The mac address used to exchange packets with the farm is one of the farm network card's address
- virtual IP-v4 or IP-v6 address configured as an alias on all servers of the farm
VIP - Load Balancing Configuration (3/4)

TCP ports on which the application is listening

userconfig.xml

```xml
<loadbalancing_list>
  <group name="FarmProto">
    <rule port="9000" proto="tcp" filter="on_port"/> <!-- 9010 = safewebserver -->
    <rule port="23" proto="tcp" filter="on_port"/>  <!-- 23 = telnet -->
    <rule port="80" proto="tcp" filter="on_addr"/>  <!-- 80 = http -->
  </group>
</loadbalancing_list>
```

- **filter="on_port"**
  - load-balance on client **TCP port** - **stateless application**
  - **no session affinity** - different TCP sessions coming from the same client are load balanced in the farm

- **filter="on_addr"**
  - load-balance on client **IP address** - **stateful application**
  - **session affinity** - a same IP client is always connected on the same server over several sessions
Power per server

- The load balancing hash table is distributed among servers **according powers**
- Thus, the traffic on each server is distributed according powers
- Without `<cluster>` definition, each server has a power of 1

The host name is the name set into the SafeKit cluster configuration file (`cluster.xml`). Set as many `<host>` as servers configured with the module.
Prerequisites for application integration

1. Application is installed and can start on both servers

2. Application offers a script interface to start it and stop it

3. Automatic start of the application at boot has been removed (will be replaced by automatic start of the module at boot: `safekit boot -m <module-name>` on)

Integration in user scripts **start_both** and **stop_both**

- On Linux, start/stop application in its environment: `su -user "appli-cmd"

- On Windows, start/stop application services: `net start|stop "service"`
## Edit User Scripts (2/3)

There are 2 logs per module:
- the stdout/stderr log of scripts `start_both`, `stop_both`
- the event log with events inside the module

<table>
<thead>
<tr>
<th>start_both</th>
<th>stop_both</th>
</tr>
</thead>
<tbody>
<tr>
<td>rem echo in the <em>stdout log</em></td>
<td>rem echo in the <em>stdout log</em></td>
</tr>
<tr>
<td>echo &quot;Running start_both %*&quot;</td>
<td>echo &quot;Running stop_both %*&quot;</td>
</tr>
<tr>
<td>rem Start your application here</td>
<td>rem option force means forcetstop</td>
</tr>
<tr>
<td>net start...</td>
<td>if &quot;%1&quot; == &quot;force&quot; goto end</td>
</tr>
<tr>
<td>rem %res% = result of start-up</td>
<td>rem Stop your application here</td>
</tr>
<tr>
<td>if %res% == 0 goto end</td>
<td>net stop...</td>
</tr>
<tr>
<td>rem message in the <em>event log</em></td>
<td>rem net stop is not synchronous...</td>
</tr>
<tr>
<td>%SAFE%\safekit printe &quot;start_both failed&quot;</td>
<td>%SAFEBIN%\sleep 10</td>
</tr>
<tr>
<td>:end</td>
<td>:end</td>
</tr>
</tbody>
</table>

The `restart command on a module runs stop_both and then start_both` thus, it is better to have `stop_both` waiting for stop of services.
Scripts - Configuration in userconfig.xml (3/3)

userconfig.xml

```xml
<user>
    [nicestoptimeout="300"]
    [forcestoptimeout="300"]
    [userlogsize="2048"]

    <var name="MYVAR" value="MYVALUE"/>
    ... as many environment variables <var> as needed
</user>
```

- **timeouts in seconds for executing** `stop_both` and `stop_both force` *scripts*
- **userlogsize**
  - size of the stdout/stdlib user scripts log (by default 2048 KB)
  - log stored in SAFEVAR/modules/<module name>/userlog.ulog
  - reset the stdout/stdlib log size when it is superior to `userlogsize`

**Predefined environment variables passed to user scripts**

- **SAFE**: root installation directory
- **SAFEUSERBIN**: user scripts execution directory of the module
- **SAFEUSERVAR**: variables directory of the module
- **SAFEMODULE**: module name (SafeKit commands do not require –m <module name> parameter in user scripts)
Evidian SafeKit

Checkers Configuration
for Mirror and Farm Modules
Checkers Configuration Overview (1/2)

```xml
<errd>
  <proc> process checker (restart or stopstart or stop) </proc>
  ... as many <proc> tags as needed
</errd>

<check>
  <intf> interface checker generated when <interface check="on"> (wait) </intf>
  <ip> virtual ip checker generated when <virtual_addr check="on"> (stopstart) </ip>
  <tcp> tcp checker (restart) </tcp>
  <ping> ping checker (wait) </ping>
  <custom> custom checker (restart or wait) </custom>
  <module> module checker (wait) </module>
  ... as many <intf>, <ip>, <tcp>, <ping>, <module>, <custom> tags as needed
</check>
```

(action in case of failure)
userconfig.xml

```xml
<service
    [maxloop="3"]
    [loop_interval="24"]
>
```

- After 3 unsuccessful restart or stop/start initiated by checkers on a server, a module stops
- Each 24 hours, the loop counter is reset
- When `loop_interval="0"`, the loop counter is disabled
- Note: administration command (stop, start…) resets the loop counter
**Process Checker (1/3)**

**process running**
- Continue the test

**errd daemon checks processes**

**process not running**
- Action on the module: `<restart | stop | stopstart>`

**Action on the module**
- **restart**
  - Restarts application locally
- **stop**
  - Stops application locally
- **stopstart**
  - Stops the module and starts the module
  - If PRIM, restarts application on SECOND
  - If ALONE or UP, restarts application locally
Process Checker Configuration (2/3)

restart module and application if myprocess.exe is not running

userconfig.xml in Linux and Windows

```xml
<errd>
  <proc name="myprocess.exe" atleast="1" action="restart" class="prim" />
</errd>
```

Test that at least one process myprocess.exe is running

userconfig.xml in Windows

```xml
<errd>
  <proc name="myservice" service="yes" atleast="1" action="restart" class="prim" />
</errd>
```

Test that the Windows or Linux service myservice is started

Other values: action="stop"|"stopstart"

class="both" for a farm

Test that myprocess.exe is not running

mirror

restart module and application if myprocess.exe is not running

PrIM

ALONE

magenta

stop_prim

restart

ALONE

UP

UP

start_prim

restart

userconfig.xml in Linux and Windows

```xml
<errd>
  <proc name="myprocess.exe" atleast="1" action="restart" class="prim" />
</errd>
```

Test that at least one process myprocess.exe is running

userconfig.xml in Windows

```xml
<errd>
  <proc name="myservice" service="yes" atleast="1" action="restart" class="prim" />
</errd>
```

Test that the Windows or Linux service myservice is started

Other values: action="stop"|"stopstart"

class="both" for a farm
Process Checker - Advanced Configuration (3/3)

userconfig.xml

```xml
<errd>
  <proc name="oracle" argregex=".*Base1.*" atleast="1" action="restart" class="prim"/>
  <proc name="oracle" nameregex="oracle_.*" atleast="1" action="restart" class="prim"/>
</errd>
```

- **Regular expression on command name and arguments** (argregex)
- **Example with 2 Oracle instances:**
  ```
  ps -e -o pid,comm,args
  20236 oracle ora_pmon_Base1
  20238 oracle ora_pmon_Base2
  ```
  Only death of process 20236 leads to the restart of the module
- **Regular expression on command name only** (nameregex)
- **You can list running processes with command names and arguments with**
  ```
  SAFE/safekit -r processtree list all
  ```
Principle of Other Checkers

Failover rule

```python
if (resource.ident==down) then
    <restart() | stopstart() | stop() | wait()>
```

test a resource / action on the module
- A farm is running an application listening on TCP port 80
- The checker checks that local connections work on TCP port 80
- **restart** locally the farm module when `tcp.Web_80` is down

**userconfig.xml**

```xml
<tcp ident="Web_80" when="both">
    <to addr="192.168.1.50" port="80" [interval="10"] [timeout="5"] />
</tcp>
```

TCP connection made on the local virtual IP address `192.168.1.50`

Test made every 10 seconds and with a timeout of 5 seconds

Default rule: `tcp_failure`: if (`tcp.? == down`) then restart();
A module must **wait** if a router is not accessible.

- wait if `ping.router_id` is down
- **wakeup** if `ping.router_id` is up

**userconfig.xml**

```xml
<ping ident="router_id" when="pre">
  <to addr="router_addr" [interval="10"] [timeout="5"]/>
</ping>
```

Default rule: `ping_failure`: if (ping.==down) then wait();
- `myexec` is a loop
- it tests a mandatory component for running the application
- it sets up/down `custom.id1`
- `wait` if `custom.id1` is down
- `wakeup` if `custom.id1` is up

**Custom Checker Configuration (wait)**

```
userconfig.xml

<custom ident="id1" when="pre" exec="myexec" [arg="..."]/>

<failover> <![CDATA
id1_failure: if (custom.id1==down) then wait();]]>

</failover>
```

- `myexec` is in `bin/` directory of the module (can be a script)
- `myexec` calls `SAFE/safekit set -r custom.id1 -v [up|down]`
Custom Checker Configuration (restart)

### myexec
- is a loop
- it checks the application
- it sets up/down custom.id2
- restart if `custom.id2` is down

#### userconfig.xml

```xml
<custom ident="id2" when="prim" exec="myexec" [arg="..."]/>
<failover><![CDATA
id2_failure: if (custom.id2==down) then restart();]]>
</failover>
```

- `myexec` is a loop
- it checks the application
- it sets up/down `custom.id2`
- restart if `custom.id2` is down
- `myexec` is in `bin/` directory of the module (can be a script)
- `myexec` calls `SAFE/safekit set -r custom.id2 -v [up|down]`
Module Checker Principle (wait)

Apache module waits for SQL Server module availability before starting.

If SQL Server module is restarted, Apache module makes a stopstart.
Module Checker Configuration (wait)

apache farm module depends on SQLserver module

wait of apache when [module.sqlserver_192.168.1.54] is down
wakeup of apache when [module.sqlserver_192.168.1.54] is up

userconfig.xml of apache module

```
<module name="sqlserver">
  <to addr="192.168.1.54" [interval="10"] [ timeout="5"]/>
</module>
```

module name to check: sqlserver
virtual IP address 192.168.1.54 of the sqlserver module

default rule - module_failure: if (module.?==down) then wait();
Module Checker - Master/Slaves

Slave Modules

- One module per application
- Light module (<service mode="light">)
- Individual start/stop of each application in start_prim/stop_prim of each light module
- Only <errd>, <check> and <user> sections + dependency on master module in userconfig.xml

Master Module

- Resources shared between all slaves (<vip>,<rfs>)
- Start-stop of slaves in start_prim/stop_prim of the master

See master.safe and slave.safe templates

Video
Failover Rule Configuration

userconfig.xml

```xml
<failover [period="5000"] [handle_time="15000"]>
<![CDATA
/* Failover rules for wait and restart custom checkers */
id1_failure : if (custom.id1 == down) then wait();
id2_failure: if (custom.id2 == down) then restart();
]]>
</failover>
```

- **period** (milliseconds): time between two evaluations of failover rules
- **handle_time** (milliseconds): time during which the failover action must stay stable (the same) before applying the action
### Checkers Summary

<table>
<thead>
<tr>
<th>Checker with <strong>wait</strong> action</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>- module stays in <img src="image" alt="WAIT (red)" /> while the resource is down</td>
<td>- interface checker</td>
</tr>
<tr>
<td>- checker is started at the start of the module (<strong>prestart</strong>)</td>
<td>- ping checker</td>
</tr>
<tr>
<td>- checker is stopped at the stop of the module (<strong>poststop</strong>)</td>
<td>- module checker</td>
</tr>
<tr>
<td>- type of checker: when=&quot;pre&quot;</td>
<td>- custom checker</td>
</tr>
</tbody>
</table>

| Checker with **restart|stop|stopstart** action                                                                 | Examples                      |
|----------------------------------------------------------------------------------------------|-------------------------------|
| - the module is ![green checkmark](image) (magenta) during application restart              | - process checker             |
| - checker is started after the application (**start_prim/both**)                            | - virtual ip checker          |
| - checker is stopped before the application (**start_prim/both**)                          | - tcp checker                 |
| - type of checker: when="prim"|"both"                          | - custom checker              |
Local action on the module

- **restart | stopstart**
  - restarts the application locally

- **stop | wait**
  - stops the application locally

- **wakeup**
  - implicit action when no wait rule applies
Local action on the module

- **restart**
  - restarts the application locally
- **stop | wait**
  - stops the application locally
- **wakeup**
  - implicit action when no wait rule applies
- **stopstart**
  - if PRIM, restarts the application on SECOND
  - if ALONE, restarts the application locally
Evidian SafeKit

Setup

Package Installation
Installation Procedure Overview

1 - Download and install the SafeKit package on your cluster

Download package
- at https://support.evidian.com/safekit
- at http://www.evidian.com/safekit (free trial, same package)

2 - Deploy mirror / farm modules with the web console or with the command line interface

For step 2, see slides on "Web Console" or "Command Line Interface"
Mirror Module Prerequisite

- Same OS on both servers
- For support, align clocks of both servers (NTP)
- Same IP network for both servers if using virtual IP
- 2 interconnects recommended to avoid split brain
- 1 dedicated replication network recommended

Application
- Installed and ready to start on both servers
- Remove automatic start of the application at boot

File replication
- Replicated directories at the same location on both servers
- On Linux, align uids/gids of replicated directories and files
- Write-back disk cache recommended for replication performances
### Farm Module Prerequisite

| Operating System | Same OS on both servers  
|                  | Linux: kernel compilation tools installed for vip kernel module  
|                  | For support, align clocks of both servers (NTP)  
| Network          | All servers in the same IP network  
|                  | Ethernet card with standard Ethernet protocol  
| Application      | Installed and ready to start on both servers  
|                  | Remove automatic start of the application at boot  

```
virtual ip = ip 1.20  ip 1.20  ip 1.20
farm(app2) = app2    app2    app2
```

```bash
ip 1.1
ip 1.2
ip 1.3
```
Install SafeKit Package

**Disk space requirement**

- Installation directory SAFE: 60 MB
- SAFE=C:safekit on Windows and /opt/safekit on Linux
- Variables directory SAFEVAR: 3 GB per module
- SAFEVAR=SAFE\var on Windows and /var/safekit on Linux

**Install on Windows as administrator**

- Double click on the package `safekitwindows_7_x_y_z.msi` *(silent installation: msiexec /qn /i safekitwindows_7_x_y_z.msi)*
- Double click on the package `safekitwindows_7_x_y_z.exe` for Windows 2012 R2

**Install on Linux as root**

- Run `safekitlinux_7_x_y_z.bin` *(extraction of the package and the safekitinstall script)*
- Run `safekitinstall` to install the package (-q for a silent installation)
License

- Without license, SafeKit starts but stops every 3 days
- Download a one-month trial key "any hostname"/"any OS" at http://www.evidian.com/safekit/
- After a purchase order, get a permanent key based on hostname/OS at https://support.evidian.com

License installation

- Save license into SAFE/conf/license.txt

License checking

- Check license with the command line SAFE/safekit level
How to Fix / Migrate to Last Version

- Go to https://support.evidian.com/safekit for reading the Software Release Bulletin (list of fixes) and the Release Notes on major upgrade (new features and migration instructions)

- Stop all running modules (safekit shutdown)

- Uninstall SafeKit package (but modules are not uninstalled)
  - Control Panel - Add/Remove Programs on Windows
  - safekit uninstall on Linux

- Install the new SafeKit package

- Before using the web console, clear the web browser’s cache (open the browser to any page, hold CTRL and SHIFT while tapping the DELETE key and clear all entries)

- Reconfigure modules (safekit config) before restarting them
Continue with the web console for configuration and monitoring of an application module
Evidian SafeKit

Management Console - Web
Tutorials

- First use of SafeKit
- Edit failover scripts
- Monitoring a high availability cluster
- Set a replication network
- Troubleshooting with the cluster management console
- Set a heartbeat timeout
- Add a ping checker

➢ http://www.evidian.com/products/high-availability-software-for-application-clustering/high-availability-cluster-management-tutorial/#first-use
Web console start

(1) Start a web browser
- Internet Explorer
- Firefox
- Chrome

(2) Set the URL http://servername:9010
- `servername` is the name or the IP address of a SafeKit server
- do not use localhost and 127.0.0.1
- check the configuration of the firewall, web proxy, trusted zones for IE

(3) Define the clusters inventory for the web console
The web console can administer one or more SafeKit clusters, the one defined into the clusters inventory

Note
Click on icon to get contextual help or open the User’s Guide. Use Ctrl-F or F3 to find words
Definition of the clusters inventory

Right-click on the entry to open the menu
- add a cluster to inventory
- edit or delete the entry from the inventory

Add a cluster to inventory
- set the name for the cluster displayed in the console
- enter the IP address or DNS name of one the servers that makes up the SafeKit cluster and press the Tab key to check the server availability
- click on Confirm

Note
The clusters inventory is stored into the cache of the browser: it must be set after the cache clean or the use of another browser
Administration of all the clusters of the inventory

(1) Click on “Clusters inventory” entry for selecting all the clusters

(2) Restricted administration panel for all the clusters

Administer in the same window, all the modules installed on each SafeKit cluster set in the inventory

- Control tab: modules control (start/stop, …)
- Monitoring tab: modules status monitoring
Administration of one cluster of the inventory

(1) Click on the name of the cluster you want to administer

(2) Full administration panel for the selected cluster

- Administer only the selected SafeKit cluster and the modules installed on it
- Cluster configuration panel: definition of the servers that make up the cluster
- Configuration tab: quick install and configuration of modules
- Control tab: modules control (start/stop, …)
- Monitoring tab: modules status monitoring
- Advanced configuration tab: modules advanced configuration and management
Configuration of a SafeKit cluster (1/3)

Simple edit mode
Define all the servers that make up the SafeKit cluster
➢ allows consistency checking of modules installed on the SafeKit cluster

Advanced edit mode
Define the network topology
➢ facilitates dynamic reconfiguration of modules on the SafeKit cluster

Note
- it is recommended to fully configure the cluster before configuring modules
- for a correct behavior, it is required to apply the same cluster configuration on all the nodes
Configuration of a SafeKit cluster (2/3)

Click on ▶️ to save and apply the configuration on all the nodes.

Add a new node in the cluster

1. Fill in the DNS name of IP address of the server and press the Tab key to check the server availability.

2. Check the name that uniquely identifies the server. It is also the name displayed into the console.

Note

You have to include into the cluster configuration, the one that is set in the inventory as connection address for the cluster.
Click on to save changes and apply the configuration on all the nodes.

The console network: it must contain all the nodes that form the cluster.

Add a new network:

1. Fill in the network name. This name is used for configuring networks used by the modules.

2. Fill in the IP address for the cluster nodes that are connected to this network.

3. Check or uncheck to set the network type:
   - web console communication
   - internal framework communication
Quick configuration and basic control

Click to open (close) the panel
- List of module templates (stored on each SafeKit cluster nodes)
- Quick install and configuration of a new module based on a template

Click to open (close) the panel
- Quick re-configuration of modules (installed on SafeKit cluster nodes)
- Basic control and monitoring of installed modules

Note: the same Configuration Wizard is used for configuring new modules and re-configuring installed modules
Quick configuration of a new module

(1) Click on the name
   - choose `mirror.safe` for real-time file replication and failover
   - choose `farm.safe` for network load-balancing and failover

(2) Enter the name of the new module
   It can be your application name

(3) Click Confirm to open the Configuration wizard
**Configuration wizard (1/5)**

Select the nodes to configure and the networks for the module

<table>
<thead>
<tr>
<th>Module Nodes</th>
<th>Select the nodes on which you want to apply the module configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module Networks</th>
<th>Select at least one network for synchronizing the module nodes and detecting its failures</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Check/uncheck the boxes
   - Check (uncheck) the box for configuring (unconfiguring) the module on the node
   - Check (uncheck) the box for using this network for internal communications of the module

2. Click on **Validate**
   - Save changes and go to the next tab

Note: Adding new node or new lan is equivalent to change the SafeKit cluster configuration
Edit the configuration of the module

(1) Fill in the form
Quick configuration of the module (main parameters)

Note: if you need advanced configuration of userconfig.xml or scripts, complete the configuration and go to Advanced Configuration tab

(2) Click on the name of the script
Start the editor for inserting the start/stop of your application

Secure the module communications
Check the box “Generate keys” (“Delete Keys”) for creating (removing) cryptographic key for the module communication

(3) Click on Validate
Save changes and go to the next tab

Tutoriel - Edit failover scripts
Configuration wizard (3/5)

Apply the configuration of the module

(1) Check the module state
Note: when “not configured” go to (3)

(2) Click on for stopping the module and then wait for the STOP (red) state on all nodes

(3) Click on Configure to apply the configuration on all the nodes
Note: if you do not want to re-configure all the nodes, use instead the Advanced Configuration Wizard available in the Advanced configuration tab
Check the result of the configuration

(1) Read the configuration result

- **success**: successful configuration on the node
- **connection error**: connection failure with the node
- **failure**: failed configuration on the node. See (2) for reading the output and find out the error

(2) Click to open (close) the panel
It contains the output of the configuration on this node

(3) Click on **Next**

Note: on error, fix the problem and **Configure** again
(1) For the 1st start, you must choose the node with the up-to-date replicated data

(2) Click on the mirror node for starting this node as primary (prim command)

(3) Close the Configuration Wizard

(4) When ALONE (green) start the other node (see next slide)
Basic control and monitoring of installed modules

Click to open the menu of commands on the module
- global start
- global stop
- re-configuration with the Configuration Wizard
- deconfiguration
- module uninstall

Click to open the menu of commands on the node (mirror modules)
- local start
  - start as primary if up to date
  - start as secondary if not up to date
- local stop
- Expert / Force start / as primary
- Expert / Force start / as secondary

Click to open the menu of commands on the node (farm modules)
- local start
- local stop

Tutoriel - Set a replication network
Monitoring of modules

Monitor modules installed on SafeKit cluster nodes

Choose the display format

Module name - cluster name

Node name

Replicated data status for mirror modules

State and color

- (green) available
- (magenta) transient
- (red) unavailable

Tutoriel –
Monitoring a high availability cluster
Full control of modules

Control and detailed status of modules installed on SafeKit cluster nodes

(1) Click on the node to display, into the right panel, the detailed status of the module on this node

(2) Browse the right panel to check
- the resources status
- the module event log
- the log of output messages from scripts
- the log of safekit commands
- informations on the server and on the last applied configuration for the module

Click to display the detailed status into a new window

Click to open the full menu of commands on the node
Advanced configuration and management of modules

On the selected SafeKit cluster node

List of installed modules
Right-click on entries to open the menu of actions

Click to open (close) the folder
- repository of modules templates
- backup storage
- workspace for implementing new modules
Right-click on entries to open the menu of actions

Click to open (close) the log of safekit commands

Click on the module to display, into the right panel, the control panel of the module
Advanced configuration of a module (1/2)

Edit the configuration files

1. Click on the module to open (close) the folder
   - Configuration files for the module

2. Click on a file to open the editor
   - scripts: insert the start/stop of the application
   - userconfig.xml: customize IP @, replication, checkers...

3. Save
   - The configuration is modified but not active (you have to apply it)

Tutoriel - Add a ping checker
Advanced configuration of a module (2/2)

Apply the new configuration

Module icon color
configuration files not modified ( - blue) or modified ( - purple) compared to the latest applied configuration

(1) Right-click on the module name to open the menu

(2) Click on Apply the configuration to open the Advanced Configuration wizard

3 latest configurations applied with success
Right-click to open the menu and restore a saved configuration
(1) Select the nodes
Check the box to apply the configuration on the node; uncheck for not applying it.
Note: by default, all the nodes on which the module is installed are selected.

(2) Click on Validate
Go to the next tab.
Advanced configuration wizard (2/2)

1. Check the module state
   - Only the nodes selected in the previous tab are displayed

2. Click on (optional)
   - If the module is not stopped, there is an attempt of dynamic configuration. It is successful only if:
     - the module is in the ALONE (green) or WAIT (red) state
     - in userconfig.xml, you have modified only parameters that are allowed to be dynamically changed

3. Click on Configure to apply the configuration on the nodes

4. Check the result of the configuration
Uninstall a module

In **Configuration** or **Advanced Configuration** tabs

(1) Click to open the menu of commands on the module

(2) Click on **Uninstall**

It opens a dialog for selecting nodes

(3) Check the box for uninstalling the module on the node; uncheck for not uninstalling

(4) Click on **Confirm**
Mirror is working well with your appli. You want to make your own module template: appli.safe

Before Pack, you may impersonalize userconfig.xml in appli

Now, your application module can be easily configured by anyone
Troubleshooting

Get snapshots of a module for the support

1. Click to open the menu of commands on the node

2. Click on **Snapshot** into the **Support** submenu

3. **Save** the snapshot on your workstation

4. Repeat and get snapshot of other node

5. Send the snapshots to support (see Evidian support slides)
### Restricting access to the web console

#### 3 roles are defined for users of the web console

- **Admin role:** 🛠️ Configuration, 📰 Control, ⬛️ Monitoring and ⚙️ Advanced Configuration tabs
- **Control role:** ⚱️ Control and ⬛️ Monitoring tabs
- **Monitor role:** ⬛️ Monitor tab only

#### Restrictions based on URL

- **Admin role:** connection to http://servername:9010
- **Control role:** connection to http://servername:9011
- **Monitor role:** connection to http://servername:9012

#### Restrictions based on basic user authentication

See the SafeKit User’s Guide

#### Restrictions based on SSL and client’s certificates for the 3 roles

Certifies access of users and secures communication. See the SafeKit User’s Guide
Refer to the User's Guide or to the Troubleshooting slides

7... Troubleshooting

7.1..... How to Read Logs?
7.2..... Stable module ✅ (green) and ✅ (green)
7.3..... Degraded module ✅ (green) and ⚠️ (red)
7.4..... Out of service module ⚠️ (red) and ⚠️ (red)
7.5..... Module ⚠️ STOP (red): restart the module
7.6..... Module ⚠️ WAIT (red): repair the resource="down"
7.7..... Module oscillating from ✅ (green) to 🔴 (magenta)
7.8..... Message on stop after maxloop
7.9..... Module ✅ (green) but non-operational application
7.10... Module ✅ (green) but problem of load balancing in a farm
7.11... Problem after Boot
7.12... Still in Trouble
4.2..... Tests of a Mirror Module

4.2.1  Test start of a mirror module on 2 servers ☠ STOP (red)
4.2.2  Test stop of a mirror module on the server ☑ PRIM (green)
4.2.3  Test start of a mirror module on the server ☠ STOP (red)
4.2.4  Test restart of a mirror module on the server ☑ PRIM (green)
4.2.5  Test swap of a mirror module from one server to the other
4.2.6  Test virtual IP address of a mirror module
4.2.7  Test file replication of a mirror module
4.2.8  Test mirror module shutdown on the server ☑ PRIM (green)
4.2.9  Test mirror module power-off on the server ☑ PRIM (green)
4.2.10 Test split brain with a mirror module
4.2.11 Continue your mirror module tests with checkers

4.3.9  Test farm module power-off of a server ☑ UP (green)
4.3.10 Continue your farm module tests with checkers
10. Exercise - Setup and Tests

During this exercise, you will install, configure and test SafeKit on two physical or virtual servers and you will use the SafeKit web console.

If you have no physical servers, you can create 2 virtual servers on your laptop with the free following tools.

<table>
<thead>
<tr>
<th>Packages</th>
<th>What?</th>
</tr>
</thead>
</table>
| VMware Player                   | [http://www.vmware.com/products/player](http://www.vmware.com/products/player)  
No license                           |
After the 90-day trial, the OS is rebooted each hour (OK for demo) |

**EXERCISE:**

- Download the SafeKit package at [http://support.evidian.com/safekit](http://support.evidian.com/safekit)
- Follow the instructions displayed during installation
- Implement with the web console a mirror and a farm module
- Pass "Tests of a Mirror Module" and "Tests of a Farm Module" described in the *SafeKit User's Guide*

**IMPORTANT:**

Always refer to the *SafeKit User's Guide* for help
Evidian SafeKit

Command Line Interface
Command Line Interface Overview

Rich interface for administrating a cluster

- Commands to manage the SafeKit cluster
- Commands to deploy modules (module install…)
- Commands to control modules (start, stop…)
- Commands to monitor modules (state…)
- Distributed commands on a cluster

Several Usage

- Operators can use the commands in a system console
- Scripts can combine several commands
- Integrators include SafeKit administration in Nagios, Microsoft SCOM, Patrol or specific consoles
**SafeKit Cluster Management**

**In the same system console, configure and monitor the SafeKit cluster**

All operations will be made in the same Windows cmd console on server 1

1 – Configure the SafeKit cluster on server 1

2 – Deploy the cluster configuration from server 1 on server 2

3 – Check the cluster configuration on server 1 and server 2

4 – Display the cluster state

Note: commands presented in this use case are the same on Linux and Windows
Prepare the SafeKit cluster configuration on server 1 in a Windows cmd console

1. Start cmd as administrator on server 1
2. cd c:\safekit

Only administrator has the right to read/write in c:safekit

3. Edit cluster.xml file
   - console="on"
     network for the web console communications
   - framework="on"
     network for the framework internal communications
   - Console network and at least one framework network
     must contain all the nodes that make up the cluster
   - The network name is set in the module configuration to designate networks used

4. safekit cluster confcheck c:safekit\var\cluster\cluster.xml

Check the cluster configuration without applying it
Apply the SafeKit cluster configuration on server 1 and export it on server 2

```bash
safekit -H "[http],[*]" -G  or  safekit -H "[https],[*]" -G
```

- Create new cryptographic key (for the global communication between cluster nodes) and apply the configuration `c:safekit\var\cluster\cluster.xml` on server 1
- Export the cluster configuration and cryptographic key on all cluster nodes
- Prefer HTTPS protocol for securing this command

Check the SafeKit cluster configuration

1. `safekit cluster confinfo`

Display the signature of the cluster configuration and the date of the configuration, for all the SafeKit nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>server1</td>
<td>e892184b2eb6f6ebf0c84b9b8c8f16c01c07b227ae6 2015-09-17T11:03:31</td>
<td></td>
</tr>
<tr>
<td>server2</td>
<td>e892184b2eb6f6ebf0c84b9b8c8f16c01c07b227ae6 2015-09-17T11:03:31</td>
<td></td>
</tr>
</tbody>
</table>

Note: the SafeKit cluster does not properly work if signatures are not identical on all nodes

2. `safekit cluster state`

Display the list of modules installed on each cluster nodes (empty at this step)
Module Management

In the same system console, deploy / control / monitor a mirror module on a cluster

All operations will be made in the same Windows cmd console on server 1

1 - Install a mirror module on server 1
2 - Configure the module on server 1
3 - Start and stop the module on server 1
4 - Install and configure the same module on server 2 from server 1
5 - Global start of the module on server 1 and server 2
6 - Global stop of the module on server 1 and server 2
7 - Set automatic start of the module at boot on both servers

Note: commands presented in this use case are the same on Linux and Windows
Install a mirror module on server 1 in a Windows cmd console

1. **Start cmd** as **administrator** on server 1

2. **cd** `c:\safekit`
   Only administrator has the right to read/write in `c:\safekit`

3. **safekit module install** `-m` `mirror1` `c:\safekit\Application_Modules\generic\mirror.safe`
   Install module, from template `mirror.safe`, under `c:\safekit\modules\mirror1`

5. **safekit module getports** `-m` `mirror1`
   List all the ports that will be used for internal communications of the module. If necessary, use this list to setup the firewall rules or use the command `firewallcfg` when it is possible (see the User’s Guide)
### Module Management (2/7)

Configure the module on server 1

1. **Edit** `c:safekit\modules\mirror1\conf\userconfig.xml`
   
   Set in this file the configuration settings for virtual IP, heartbeats, directories to replicate, checkers.

2. **Edit** `c:safekit\modules\mirror1\bin\start_prim.cmd`
   
   Start the application in this script.

3. **Edit** `c:safekit\modules\mirror1\bin\stop_prim.cmd`
   
   Stop the application in this script.

4. **Manage encryption of internal communications for the module (optional)**
   
   - Create new cryptographic key with: `safekit module genkey -m mirror1`
   - Delete the cryptographic key with: `safekit module delkey -m mirror1`

4. **safekit config -m mirror1**
   
   - apply the configuration
   - must be executed each time `userconfig.xml`, `start_prim.cmd`, `stop_prim.cmd`, or cryptographic key are modified
   - the module must be stopped before executing this command
Module Management (3/7)

Start and stop of the module on server 1

1. **safekit prim -m mirror1**
   - we suppose that replicated directories are up-to-date on server 1
   - use prim command only the first time; next time, use start command

2. **safekit logview -m mirror1**
   - check that you reach the message "Local state ALONE green" in the log
   - normally, application integrated in start_prim is started - check it

3. **safekit state -m mirror1**
   returns the state of the module on the local server; normally, ALONE(green)

4. **safekit stop -m mirror1**

5. **safekit logview -m mirror1**
   - check that you reach the message "Local state STOP red" in the log
   - normally, application integrated in stop_prim is stopped - check it
Install and configure the module on server 2

1. `safekit -H http://server2:9010 -E mirror1`
   - export the mirror1 module configuration (including cryptographic key if exists) to server 2 (it’s a distributed command with a `-H` parameter)
   - the mirror1 module is installed and configured on server 2
   - make sure that the mirror1 module is stopped locally on server1 before executing this command
   - note that a list of servers can be used to export a farm module on several servers

2. `safekit cluster state`
   
   List the modules installed on each cluster nodes and display informations on the module configuration

<table>
<thead>
<tr>
<th>Node</th>
<th>Module</th>
<th>Mode</th>
<th>Id</th>
<th>Date</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>server1</td>
<td>mirror1</td>
<td>mirror 1</td>
<td>2015-09-17T11:52:23</td>
<td>3265107898a2db05535a954e91d98c628f18d2e0</td>
<td></td>
</tr>
<tr>
<td>server2</td>
<td>mirror1</td>
<td>mirror 1</td>
<td>2015-09-17T11:55:25</td>
<td>3265107898a2db05535a954e91d98c628f18d2e0</td>
<td></td>
</tr>
</tbody>
</table>

Note: the module does not properly work if the configuration (mode, id, signature) is not identical on all the nodes that implement the module
Module Management (5/7)

Global start of the module on server 1 and server 2

1. `safekit -H "[http],server2,server1" start -m mirror1`
   
   This distributed command starts on server1 and server2 (node names into cluster.xml)

2. `safekit -H "[http],*" log -m mirror1`
   
   Log of module on all nodes (set into cluster.xml)

3. `safekit -H "[http],server1,server2" state -m mirror1`
   
   state of module on server1 and server2 - PRIM(green) and SECOND(green)

Notes:

- server1 with the up-to-date replicated directories is PRIM (was started with prim command)
- data are reintegrated from server 1 to server 2
Module Management (6/7)

Global stop of the module on server 1 and server 2

```shell
```

be careful, you may have an application failover if server 1 is stopped before server 2

The best is to stop first the secondary and then the primary

1. ```shell
   safekit -H http://server2:9010 stop -m mirror1
   ```

2. ```shell
   safekit -H http://server2:9010 state -m mirror1
   ```
   wait for the state STOP red

3. ```shell
   safekit stop -m mirror1
   ```
   stop mirror1 locally on server 1

4. ```shell
   safekit state -m mirror1
   ```
   server 1 goes from ALONE to STOP
Module Management (7/7)

Set automatic start of the module at boot

1. `safekit -H "[http],*" boot -m mirror1 on`
   
   Sets automatic start at boot

2. `safekit -H "[http],*" boot -m mirror1 status`
   
   Gives state of modules at boot

Notes:

- at each boot of server1 or server2, mirror1 module will be started
- thus, application in mirror1 will be automatically started on primary server and not on secondary
- that’s why, automatic start of the application at boot must be removed and replace by the module start
### More (1/10) - Installation Directories

<table>
<thead>
<tr>
<th>Installation</th>
<th>Linux</th>
<th>/opt/safekit</th>
<th>/var/safekit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Windows</td>
<td>C:safekit</td>
<td>C:safekit\var</td>
</tr>
</tbody>
</table>

| Templates (.safe)  | SAFE/Application_Modules/ |

<table>
<thead>
<tr>
<th>Installed modules</th>
<th>SAFE/modules/&lt;module-name&gt;/conf/userconfig.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SAFE/modules/&lt;module-name&gt;/bin/start_prim and stop_prim for a mirror</td>
</tr>
<tr>
<td></td>
<td>SAFE/modules/&lt;module-name&gt;/bin/start_both and stop_both for a farm</td>
</tr>
</tbody>
</table>

| Scripts log        | SAFEVAR/modules/<module-name>/userlog.ulog (stdout/stderr of scripts) |

| Snapshot           | SAFEVAR/snapshot/modules/<module-name>/ |

| Commands log       | SAFEVAR/commandlog (safekit commands that have been executed on the server) |
## More (2/10) – Services and Processes

### Service safeadmin - useful for local commands
- Automatically started at boot, mandatory service
- **Windows**: `net [start | stop] safeadmin`
- **Linux**: `service safeadmin [start | stop]`

### Service safewebserver - useful for distributed commands, web console, `<module>` checkers
- Automatically started at boot (httpd process)
- `safekit boot [webon | weboff | webstatus]` *(default is automatic start at boot)*
- `safekit webserver start | stop | restart`

### Processes per module
- **Heart**: manages the recovery procedures
- **Circolog**: manages the circular module’s log
- **Nfsbox, reintegre**: file replication / reintegration
- **Vipd**: synchronizes a farm of servers
- **Checkers** (`ipcheck`, `intcheck`, …)
- **Errd**: manages detection of process death

Note: there is also a SNMP agent - `safeagent`
### More (3/10) - Boot and Shutdown of Modules

#### Boot - automatic start of a module

```
safekit boot -m <module-name> [on | off | status]  (default module start at boot is off)
```

#### Shutdown - clean stop of all running modules

```
safekit shutdown - command which stops all running modules
```

<table>
<thead>
<tr>
<th>OS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>Modules are automatically stopped during the stop of the <code>safeadmin</code> service</td>
</tr>
<tr>
<td>Windows</td>
<td><strong>Modify the Windows shutdown script in the Microsoft Management Console</strong></td>
</tr>
<tr>
<td></td>
<td>1. Start cmd as administrator and type <code>mmc</code></td>
</tr>
<tr>
<td></td>
<td>2. File - Add/Remove Snap-in – Add - &quot;Group Policy Object Editor&quot;</td>
</tr>
<tr>
<td></td>
<td>3. Under &quot;Console Root&quot;/&quot;Local Computer Policy&quot;/&quot;Computer Configuration&quot;/&quot;Windows Settings&quot;/&quot;Scripts (Start-up/Shutdown)&quot;, double click on &quot;Shutdown&quot;</td>
</tr>
<tr>
<td></td>
<td>4. Click on Add then set for &quot;Script Name&quot;: &quot;c:safekit\private\bin\safekitshutdown.cmd&quot;</td>
</tr>
</tbody>
</table>

Page 16 © Evidian
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>safekit module install -m &lt;module-name&gt;</td>
<td>Installs the template as a new module</td>
</tr>
<tr>
<td>SAFE/Application_Modules/template.safe</td>
<td></td>
</tr>
<tr>
<td>safekit config -m &lt;module-name&gt;</td>
<td>To be applied after modifications of userconfig.xml, start_prim/both, stop_prim/both in &lt;SAFE&gt;/modules/&lt;module-name&gt;</td>
</tr>
<tr>
<td>safekit confcheck -m &lt;module-name&gt;</td>
<td>Check, without applying it, the configuration of the module in &lt;SAFE&gt;/modules/&lt;module-name&gt;</td>
</tr>
<tr>
<td>safekit module getports -m &lt;module-name&gt;</td>
<td>List ports used for internal communications of the module</td>
</tr>
<tr>
<td>safekit module genkey -m &lt;module-name&gt;</td>
<td>Create (delete) cryptographic key for securing the internal communications of the module. Require the module re-configuration.</td>
</tr>
<tr>
<td>safekit module delkey -m &lt;module-name&gt;</td>
<td></td>
</tr>
</tbody>
</table>
## More (5/10) - Commands to Control a Module

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>safekit start -m &lt;module-name&gt;</code></td>
<td>Starts a module</td>
</tr>
<tr>
<td><code>safekit waitstart -m &lt;module-name&gt;</code></td>
<td>Waits for the end of the module start</td>
</tr>
<tr>
<td><code>safekit stop -m &lt;module-name&gt;</code></td>
<td>Stops a module</td>
</tr>
<tr>
<td><code>safekit waitstop -m &lt;module-name&gt;</code></td>
<td>Waits for the end of the module stop</td>
</tr>
<tr>
<td><code>safekit stopstart -m &lt;module-name&gt;</code></td>
<td>Stop/start with failover on the other server if the module is PRIM</td>
</tr>
<tr>
<td><code>safekit forcestop -m &lt;module-name&gt;</code></td>
<td>Forces the stop when some processes are frozen</td>
</tr>
<tr>
<td><code>safekit restart -m &lt;module-name&gt;</code></td>
<td>Restart of the application locally with no failover</td>
</tr>
<tr>
<td><code>safekit swap -m &lt;module-name&gt;</code></td>
<td>Swaps the roles of primary and secondary (mirror)</td>
</tr>
<tr>
<td><code>safekit prim -m &lt;module-name&gt;</code></td>
<td>Special - Forces the module to start as primary (mirror)</td>
</tr>
<tr>
<td><code>safekit second -m &lt;module-name&gt;</code></td>
<td>Special - Forces the module to start as secondary (mirror)</td>
</tr>
</tbody>
</table>
### Control a started module in stable state (ALONE, UP, PRIM, SECOND, WAIT)

#### Update of DNS names

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>safekit -H &quot;[http],*&quot; -G</code></td>
<td>Redo a DNS name resolution for all names specified into cluster.xml and userconfig.xml without stopping the modules (when possible).</td>
</tr>
</tbody>
</table>

#### Checkers and failover control for maintenance

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>safekit checker off -m &lt;module-name&gt;</code></td>
<td>Stops/starts checkers (interface, TCP, IP, custom, etc ...).</td>
</tr>
<tr>
<td><code>safekit checker on -m &lt;module-name&gt;</code></td>
<td>▪ resource usersetting.checker=&quot;on&quot;</td>
</tr>
<tr>
<td><code>safekit errd suspend -m &lt;module-name&gt;</code></td>
<td>Suspends/resumes the process death detection.</td>
</tr>
<tr>
<td><code>safekit errd resume -m &lt;module-name&gt;</code></td>
<td>▪ resource usersetting.errd=&quot;on&quot;</td>
</tr>
<tr>
<td><code>safekit failover off -m &lt;module-name&gt;</code></td>
<td>Used to dynamically set the failover attribute to on or off.</td>
</tr>
<tr>
<td><code>safekit failover on -m &lt;module-name&gt;</code></td>
<td>▪ must be ran on the 2 servers of the mirror module</td>
</tr>
<tr>
<td></td>
<td>▪ resource usersetting.errd=&quot;on&quot;</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><code>safekit state</code></td>
<td>Displays the status of all modules</td>
</tr>
<tr>
<td><code>safekit state -m &lt;module-name&gt;</code></td>
<td>Displays the status of a module</td>
</tr>
<tr>
<td><code>[\(-v\)]</code></td>
<td><code>\(-v\)</code> : status of all resources are also listed</td>
</tr>
<tr>
<td><code>safekit logview -m &lt;module-name&gt;</code></td>
<td>A loop which displays I and E messages of a module</td>
</tr>
<tr>
<td>`[(-l)] [(-A)] [(-l) en</td>
<td>fr]`</td>
</tr>
<tr>
<td>`[(-l) en</td>
<td>fr]`</td>
</tr>
<tr>
<td><code>safekit log -m &lt;module-name&gt;</code></td>
<td>Displays I and E messages in the last &lt;nb&gt; messages of the log</td>
</tr>
<tr>
<td>`[(-s) nb] [(-l)] [(-A)] [(-l) en</td>
<td>fr]`</td>
</tr>
<tr>
<td><code>safekit logsave -m &lt;module-name&gt;</code></td>
<td>Saves the log in <code>/tmp/f.txt</code> (absolute path mandatory)</td>
</tr>
<tr>
<td>`[(-l)] [(-A)] [(-l) en</td>
<td>fr] /tmp/f.txt`</td>
</tr>
</tbody>
</table>

Indicates the version of SafeKit and the license.
More (8/10) - Command for Support

To analyze a problem, the support asks for snapshots of a module on all servers

- Locally on a server, takes a snapshot of a module in /tmp/snapshot_xx.zip
- A snapshot creates a dump on the server
- The command puts in .zip file last 3 dumps and last 3 configurations of the module
- Dumps and configurations are in SAFEVAR/snapshot/modules/<module-name>/

```
safekit snapshot
-m <module-name>
/tmp/snapshot_xx.zip
```

- To solve a problem in real time on a server, the support may ask to make a dump of a module when the problem occurs
- A dump creates a directory "dump_<date>_<hour>" on the server side
- The "dump_<date>_<hour>" directory contains the module logs and information on the system state and SafeKit processes at the time of the dump
- The directory is created under SAFEVAR/snapshot/modules/<module-name>/

```
safekit dump
-m <module-name>
```

absolute path mandatory
### More (9/10) - Command for the SafeKit Cluster

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>safekit cluster config &lt;path of xml file&gt;</code></td>
<td>Apply the new SafeKit cluster configuration with the content of the file passed as argument and create new cryptographic key for the global communications</td>
</tr>
<tr>
<td><code>safekit cluster confinfo</code></td>
<td>Return, for each cluster nodes, the date and the digital signature of the last cluster configuration. Note: 0 and -1 values means that the nodes can not exchange their state. It occurs on failure or when the cluster configuration is not identical on all nodes.</td>
</tr>
<tr>
<td><code>safekit cluster state</code></td>
<td>Display for all cluster nodes, the list of installed modules and information on the module configuration. Note: this list is up to date only if nodes can exchange their state</td>
</tr>
</tbody>
</table>
### More (10/10) - Distributed Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Installs if not yet installed and configures on all servers</td>
</tr>
<tr>
<td><code>safekit -H http://server1:9010 -N /dir -l mirror</code></td>
<td>Imports a module from server1 in /Application_Modules/dir</td>
</tr>
<tr>
<td><code>safekit -H http://127.0.0.1:9010 -N /dir -E mirror</code></td>
<td>Installs and configures locally</td>
</tr>
<tr>
<td><code>safekit -H &quot;*&quot; -G</code></td>
<td>Exports the SafeKit cluster configuration on all the servers set into the cluster.xml</td>
</tr>
</tbody>
</table>
### Reminder (1/2) - Main States of a Module

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALONE</strong></td>
<td>module is primary without secondary and has the up-to-date replicated files. <em>Application is started on the ALONE server.</em></td>
<td>green</td>
</tr>
<tr>
<td><strong>PRIM</strong></td>
<td>module is primary with a secondary and replicated files are mirrored on the secondary. <em>Application is started on the PRIM server.</em></td>
<td>green</td>
</tr>
<tr>
<td><strong>SECOND</strong></td>
<td>module is secondary with a primary and replicated files are mirrored from the primary.</td>
<td>green</td>
</tr>
<tr>
<td><strong>UP</strong></td>
<td>module is in a farm. <em>Application is started on the UP server.</em></td>
<td>green</td>
</tr>
</tbody>
</table>
| **WAIT** | - may be because of a not uptodate server  
  - may be because of a resource set to down by a checker | red |

**PRIM**
- **uptodate**
- **up to date**
- module is primary with a secondary and replicated files are mirrored on the secondary. *Application is started on the PRIM server.*

**SECOND**
- **uptodate**
- module is secondary with a primary and replicated files are mirrored from the primary.

**WAIT**
- may be because of a not uptodate server  
- may be because of a resource set to down by a checker
Reminder (2/2) - 2 Logs per Module

**Script log: output messages**
(SAFEVAR/modules/<module-name>/userlog)

```
 2014/01/27 14:11:32 stop_prim "force"
"Running stop_prim force WAIT STOP"
 2014/01/27 14:13:08 prestart "start"
"Running stop_prim *
"Running stop_prim force"
 2014/01/27 14:14:47 start_prim
"Running start_prim WAIT ALONE;"
 2014/01/27 15:26:04 stop_prim
"Running stop_prim PRIM WAIT"
 2014/01/27 15:26:19 stop_prim "force"
"Running stop_prim force WAIT STOP"
```

**Module: event log** (safekit logview -m <module-name>)

stdout / stderr of scripts
<table>
<thead>
<tr>
<th>Module state</th>
<th>What to do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIM/ALONE/UP</td>
<td>A checker tries to restart the application</td>
</tr>
<tr>
<td>PRIM/ALONE/UP</td>
<td>But each time, it detects an error</td>
</tr>
<tr>
<td>PRIM/ALONE/UP</td>
<td>Find in the event log the checker making the oscillation</td>
</tr>
</tbody>
</table>

- By default in `userconfig.xml`, `maxloop=3`
- At the 4th unsuccessful restart, the module stops on the server: it goes to **STOP** (red) state
- In a farm module, the server is no more load balanced
- In a mirror module with a PRIM server, there is a failover of the application on the SECOND server
- In a mirror module with an ALONE server, the application is stopped
<table>
<thead>
<tr>
<th>Module state</th>
<th>What to do?</th>
</tr>
</thead>
</table>
| green PRIM/ALONE/UP but non-operational application | - Look for scripts errors in the script log  
- Run the restart command  
- Check if application is operational  
- If not, run the stopstart command  
- If it was a PRIM server, failover of the application on the SECOND server  
- Check if application is operational |
## Mirror or Farm (3/3) – Nodes communication failure

<table>
<thead>
<tr>
<th>Module state</th>
<th>What to do?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For a farm module</strong></td>
<td>Nodes can not communicate because:</td>
</tr>
<tr>
<td><img src="#" alt="green UP" /> <img src="#" alt="green UP" /> and each node got 100% load-balancing</td>
<td>▪ network failure: check the network connection</td>
</tr>
<tr>
<td></td>
<td>▪ Firewall rules: check the firewall settings</td>
</tr>
<tr>
<td></td>
<td>▪ Bad module or SafeKit cluster configuration</td>
</tr>
<tr>
<td>Check that the signature for the module and the SafeKit cluster configuration are identical on all nodes with the commands:</td>
<td>When different, re-apply the configuration on all nodes.</td>
</tr>
<tr>
<td>✓ safekit confinfo –m &lt;nom du module&gt;</td>
<td></td>
</tr>
<tr>
<td>✓ safekit cluster confinfo</td>
<td></td>
</tr>
</tbody>
</table>

**For a mirror module**

![green ALONE](#) ![red WAIT](#) and each node got state UNKNOWN for the remote node
## Farm (1/1) - Problem of Load Balancing

<table>
<thead>
<tr>
<th>Module state</th>
<th>What to do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>![green]</td>
<td>1. Find in userconfig.xml the virtual IP address of the farm</td>
</tr>
<tr>
<td>![green]</td>
<td>2. Find one server (green) in the farm with established connections on the virtual IP address:</td>
</tr>
<tr>
<td>![green]</td>
<td>- Windows: `netstat -an</td>
</tr>
<tr>
<td>![green]</td>
<td>- Linux: `netstat -an</td>
</tr>
<tr>
<td>![green]</td>
<td>3. Stop the module on all servers except the one with established connections</td>
</tr>
<tr>
<td>![green]</td>
<td>4. Check that the single (green) server manages all TCP connections on the virtual IP address</td>
</tr>
</tbody>
</table>

**Load balancing is not working**

but

![green] UP

© Evidian
Mirror (1/6) – First Start of a Module

At first start, synchronize mirrored data in the right direction

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Prim on server1</th>
<th>Second on server2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A new directory to replicate. Where are up-to-date data?</td>
<td>Start the server with the up-to-date data as primary</td>
<td>Start the other server as secondary</td>
</tr>
<tr>
<td>server1</td>
<td>server1</td>
<td>server1</td>
</tr>
<tr>
<td>?</td>
<td>uptodate</td>
<td>uptodate</td>
</tr>
<tr>
<td>red</td>
<td>green</td>
<td>green</td>
</tr>
<tr>
<td>STOP</td>
<td>ALONE</td>
<td>PRIM</td>
</tr>
<tr>
<td>server2</td>
<td>server2</td>
<td>server2</td>
</tr>
<tr>
<td>?</td>
<td>not uptodate</td>
<td>uptodate</td>
</tr>
<tr>
<td>red</td>
<td>red</td>
<td>green</td>
</tr>
<tr>
<td>STOP</td>
<td>STOP</td>
<td>SECOND</td>
</tr>
</tbody>
</table>

Use **prim** command only the first time
After use **start** command

Use **second** command only the first time
After use **start** command
A not up-to-date server cannot start automatically as primary

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Stop on server1</th>
<th>Start on server2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server1 primary ALONE.</td>
<td>Stop server1 with the up-to-date data</td>
<td>Server2 waits for server1 for resynchronizing data</td>
</tr>
<tr>
<td>Server2 STOP, not up-to-date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>server1</td>
<td>server2</td>
<td></td>
</tr>
<tr>
<td>uptodate</td>
<td>not uptodate</td>
<td></td>
</tr>
<tr>
<td>green</td>
<td>red</td>
<td></td>
</tr>
<tr>
<td>ALONE</td>
<td>STOP</td>
<td></td>
</tr>
</tbody>
</table>

- safekit state -v: rfs.uptodate = down on server2
- In this case: `start` on server1
- Or sacrifice up-to-date data: `stop; prim` on server2
### Mirror (3/6) - Degraded

**Degraded mode on primary when failure of replication process**

<table>
<thead>
<tr>
<th>Stable state</th>
<th>Failure of replication process on server1 (nfsbox)</th>
<th>Restart replication process on server1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application is running on server1 and replication works</td>
<td>Application continues on server1 without replication</td>
<td>Administrator stop / start the module on server1</td>
</tr>
<tr>
<td>server1 uptodate</td>
<td>server1 degraded</td>
<td>server1 uptodate</td>
</tr>
<tr>
<td>server2 uptodate</td>
<td>server2 not uptodate</td>
<td>server2 uptodate</td>
</tr>
</tbody>
</table>

**safekit state -v : rfs.degraded = up**
userconfig.xml: `<service mode="mirror" failover="on" | "off">

How it works with
failover="on"

Server 1 stops – server 2 becomes ALONE and restarts the application

server1
STOP

server2
ALONE

How it works with
failover="off" (1/2)

Server 1 stops - server 2 does not become primary but waits

server1
STOP

server2
WAIT

How it works with
failover="off" (2/2)

An administrator forces the application to restart on server 2 with `stop` and `prim`

server1
STOP

server2
ALONE

safekit stop; safekit prim
userconfig.xml:<service mode="mirror" defaultprim="server1">

(1/2) Failover
server1 stops - server2 becomes primary

(2/2) Failback without defaultprim
server1 is restarted - server2 remains primary

(2/2) Failback with defaultprim="server1"
server1 is restarted - server 1 returns as primary

Restart of the failed server: the application continues its execution on server 2

Restart of the failed server: the application is automatically stopped on server 2 and started on server 1
A partially reintegrated server has potentially corrupted files

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Stop on server1</th>
<th>Prim on server2</th>
</tr>
</thead>
<tbody>
<tr>
<td>server1 is primary ALONE - server2 is reintegrating files</td>
<td>server1 stops - server2 is partially reintegrated</td>
<td>stop; prim fails because server2 has potentially corrupted files</td>
</tr>
</tbody>
</table>

- server1 uptodate  
  - green ALONE
- server2 not uptodate  
  - magenta SECOND

- server1 uptodate  
  - red STOP
- server2 not uptodate  
  - red WAIT

- server1 uptodate  
  - red STOP
- server2 not uptodate  
  - red WAIT

In this case: start server1 and server 2 will complete reintegration of files
Or restore files from a backup and run sakekit primforce on server2
<table>
<thead>
<tr>
<th><strong>Initial state</strong></th>
<th><strong>Stop on server1</strong></th>
<th><strong>Prim on server2</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>server1 is primary ALONE - server2 is reintegrating files</td>
<td>server1 stops - server2 is partially reintegrated</td>
<td>stop ; prim fails because server2 has potentially corrupted files</td>
</tr>
<tr>
<td>server1 uptodate</td>
<td>server1 uptodate</td>
<td>server1 uptodate</td>
</tr>
<tr>
<td>server2 not uptodate</td>
<td>server2 not uptodate</td>
<td>server2 not uptodate</td>
</tr>
</tbody>
</table>

- In this case: **start** server1 and server 2 will complete reintegration of files
- Or restore files from a backup and run **sakekit primforce** on server2
Log (1/3) - Message on Administrator Command

Want to know if an administrator has stopped the module on server 1?

```
safekit stop
  ▪ Action stop called by web@<IP>/SYSTEM/root

safekit start
  ▪ Action start called by web@<IP>/SYSTEM/root

safekit restart
  ▪ Action restart called by web@<IP>/SYSTEM/root

safekit swap  (mirror module only)
  ▪ Action swap called by web@<IP>/SYSTEM/root
```

Message in event log of the module on server1

`web@<ip>` command called in the console

`SYSTEM` command line on Windows

`root` command line on Linux

Message in event log of the module on server1
Log (2/3) - Message from a Wait Checker

Want to know if a checker has put the module in wait state?

**Checker of a local network interface** - `<interface check="on">` in userconfig.xml
- Resource `intf.<network>` set to down by `intfcheck`
- Action `wait` from failover rule `interface_failure`

**Checker of an external device** - `<ping>` checker in userconfig.xml
- Resource `ping.<id>` set to down by `pingcheck`
- Action `wait` from failover rule `ping_failure`

**Checker of another module** - `<module>` checker in userconfig.xml
- Resource `module.<other_module_ip>` set to down by `modulecheck`
- Action `wait` from failover rule `module_failure`

**Custom checker** - `<custom ident="id" when="pre">` in userconfig.xml
- Resource `custom.<id>` set to down by `<customscript>`
- Action `wait` from failover rule `<customid_failure>`
Want to know if a checker has restarted the module?

**Checker of processes** - `<errd>` in `userconfig.xml`
- event atleast on proc `<appli.exe>`
- Action restart (ou stopstart) called by `<errd>`

**TCP checker of the application** - `<tcp ident="id" when="prim"|"both">` in `userconfig.xml`
- Resource `<tcp <id>>` set to down by `<tcpcheck>`
- Action restart (ou stopstart) from failover rule `<tcp_failure>`

**Custom checker** - `<custom ident="id" when="prim"|"both">` in `userconfig.xml`
- Resource `<custom <id>>` set to down by `<customscript>`
- Action restart (ou stopstart) from failover rule `<customid_failure>`

**Checker of the virtual IP** - `<virtual_addr addr="@" check="on">` in `userconfig.xml`
- Resource `<ip <*>>` set to down by `<ipcheck>`
- Action stopstart from failover rule `<ip_failure>`
Evidian SafeKit

Access to Evidian Support
Welcome to Evidian's support web site...

- **Software Keys**
  Get by e-mail the license keys required to use Evidian products.

- **Subscription Request**
  To be able to access the customer reserved area.

- **Download**
  Get products, patch levels, fixes, service packs and tools.
  Access the "Exchange area".

- **Call Desk**
  Submit new problems to Evidian support.
  Follow-up existing calls.

- **Knowledge Base**
  Search for solutions and technical information using the Knowledge Base

- **Bulletin**
  Evidian Support Bulletin

What is explained in this presentation...
You can get your permanent keys

- **Software Keys**
  Get by e-mail the license keys required to use Evidian products.

- **Subscription Request**
  To be able to get your permanent license keys, you need information which are on the small envelope attached to the parcel.
  The DELIVERY NOTE Nr is written in big on the top of the delivery note. The OFE Nr is situated on the left.

  - First name: [ ]
  - Last name: [ ]
  - Company/Organization: [ ]
  - Mail reply address: [ ]
  - DELIVERY NOTE Nr / BON DE LIVRAISON N°: [ ]
  - OFE Nr / N° COMMANDE: [ ]

  [Continue]
Subscription Request to Create an Account

- **Software Keys**
  Get by e-mail the license keys required to use Evidian products.

- **Subscription Request**
  To be able to access the customer reserved area.

- **Download**
  Get products, patch levels, fixes, service packs and tools.
  Access the "Exchange area".

- **Call Desk**
  Submit new problems to Evidian support.
  Follow-up existing calls.

- **Knowledge Base**
  Search for solutions and technical information using the Knowledge Base

The procedure must be executed only once

Fill in the form with:
- your customer ID
- your customer confidential ID
- your e-mail address

What you will get
- a personal user id & password
After Account Creation

User can log on and has a full access to the web site

Evidian's support web site...

- **Software Keys**
  Get by e-mail the license keys required to use Evidian products.

- **Download**
  Get products, patch levels, fixes, service packs and tools.
  Access the “Exchange area”.

- **Call Desk**
  Submit new problems to Evidian support.
  Follow-up existing calls.

- **Knowledge Base**
  Search for solutions and technical information using the Knowledge Base.

“Bruno Rochat”
log on

No more locked

Page 5 © Evidian
Call Desk to Report Problems

- **Software Keys**
  Get by e-mail the license keys required to use Evidian products.

- **Download**
  Get products, patch levels, fixes, service packs and tools. Access the "Exchange area".

- **Call Desk**
  Submit new problems to Evidian support. Follow-up existing calls.

- **Knowledge Base**
  Search for solutions and technical information using the Knowledge Base

---

Call Desk is the main tool for communication with Evidian support team

Each problem / question

- is stored as a "Call" in the repository
- is referenced by its unique Call Number “EVDxxxxxxxxxxx”

The 2 main operations are

- call creation
- exchange between support/customer inside a call
Call Desk Home Page

**Call Desk**
Submit new problems to Evidian support.
Follow-up existing calls.

Name: Bruno ROCHAT - WEB
Company: Evidian Customer Care - Web
Phone:
E-Mail: Bruno.Rochat@evidian.com

Opened Calls

<table>
<thead>
<tr>
<th>Call #</th>
<th>Status</th>
<th>Type</th>
<th>Priority</th>
<th>Create Do.</th>
<th>Domain</th>
</tr>
</thead>
</table>

- Call creation
- Call search and update
- Remote control
- Report

Submit New Call  Search Calls  Remote control  Create report
Call Creation

General information

Problem summary

Problem detail: scenario date and hour

Problem/Question Summary*
How can I restart my sqlserver module which is WAIT (red) on both servers?

Problem/Question Detail
Our problem is on a sqlserver module.
Yesterday afternoon, May 19th 2010 at 7:00pm, both servers were PRIMARY (green) and SECONDARY (green).
This morning at 9:00 am, both servers are in WAIT (red) and WAIT (red).
How can I restart the sqlserver module in green state?

Attach snapshots

Call creation

Add attachment  Submit  Cancel
Attach Snapshots

**Remark text:**
Please find enclosed the snapshots of sqlserver module on both servers

<table>
<thead>
<tr>
<th>File Name</th>
<th>Max Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>snapshot_sqlserverServer1.zip</td>
<td>4670 KB</td>
</tr>
<tr>
<td>snapshot_sqlserverServer2.zip</td>
<td>3913 KB</td>
</tr>
</tbody>
</table>

**Indicate if you put snapshots here or in your private upload area**

**Attach snapshots here if < 10 MBytes else put them in your private download area**
Add a remark to continue the exchange with support.

Exchange between Evidian support and customer until the call is closed.

To deconfigure the checker in the module, you must put this checker in comment in the file userconfig.xml.

For that:
- add the line userconfig.xml
- remove the definition of the checker: this is defined like that:

```xml
<check>    
    <id="checke name">"/>    
    <address="<IP address>"/>    
</check>
```

For the module:

```
<module name="\"a\"" id="checke name">"/>    
    <address="<IP address>"/>    
</module>
```

More remarks:
- 20/05/2010 15:47:45 CUST: rockit: Closure requested by rockit.
- 20/05/2010 15:47:47 CUST: rockit: Thank you. The aspserver module is restarted in PRIM (green) - SECOND (green).
- 20/05/2010 14:98:58 SUP: Dominique Pires: To deconfigure the checker in the module, you must put this checker in comment in the file source.
- 20/05/2010 14:52:52 CUST: rockit: The event component has been removed last night. How can I deconfigure the checker in the module?
- 20/05/2010 13:56:13 SUP: Dominique Pires: According to the logs, it seems that the 2 servers are in WAIT state, because the ping checker has been removed.
- 20/05/2010 16:19:08 CUST: rockit: Please find enclosed the snapshot of aspserver module on both servers.
Download / Upload Exchange Area

- **Download**
  Get products, patch levels, fixes, service packs and tools.
  Access the "Exchange area".

---

Customer Care

Download and exchange area

- **Product download area**
  This area is accessible to all supported customers. You will find here all supported releases and all Evidian product lines as well as the latest fixes and service packs.

- **Private area [support]**
  Area reserved for exchanging data with the support team. Files in this area are visible only to members of your company and to Evidian Customer Care. You can upload or download files in this area.

- **Private area [internal European customer]**
  Area reserved for exchanging data with the support team. Files in this area are visible only to members of your company and to Evidian Customer Care. You can upload or download files in this area.

---

Get last product package

Private area to upload/download files
Product Download Area

- Product download area
  This area is accessible to all supported customers. You will find here all supported releases and all Evidian product lines as well as the latest fixes and service packs.

Welcome to SafeKit page
- Go to last version 7.2.
- Version 7.2
- Version 7.3

Customer Care

Current SafeKit packages for Windows

Supported versions
- Windows Server 2015 (Intel x64 64-bit kernel)
- Windows Server 2012 R2 (Intel x64 64-bit kernel)
- Windows 10 Enterprise (Intel x64 64-bit kernel)

Go to
- SafeKit Software Release Bulletin for details on this version.
- Documentation for the SafeKit User’s guide, the SafeKit Release Notes, ...

Example of SafeKit Windows package
Private Area for Upload / Download

Upload a file

You can create a directory and upload SafeKit snapshots here
Knowledge Base

Search for solutions and technical information using the Knowledge Base

Customer Care

Enter one or more keywords to search. Note that ‘*’ and ‘?’ wildcards are supported:

Search for: errd
Document types: All
Products: All Products

Run search

Results per page: 10

Search article about SafeKit "errd" component
14. Exercise: Advanced Configuration

During this exercise, you will configure a mirror module, a farm module and checkers on two servers with the same OS:

OVERVIEW OF EXERCISE:

- Mirror Module Configuration Exercise
- Farm Module Configuration Exercise
- Checkers Configuration Exercise

NOTE:

Each step is described in the following pages.

IMPORTANT:

Refer to SafeKit User's Guide for help
14.1 Mirror Module Configuration Exercise

14.1.1 Install a Mirror Module

If not already done, install and configure the generic module **mirror.safe**.

14.1.2 Configure Multiple Virtual IP Addresses

Add a new virtual IP address on the same network interface (modify userconfig.xml of mirror module). And check that this virtual IP address is responding when the primary server is started.

For more information, refer to “<vip> Structure” in *SafeKit User's Guide*.

14.1.3 Configure Heartbeat Timeout

Change the heartbeat timeout to 60 seconds and check what happen when you disconnect the heartbeat network interface.

Refer to “Heartbeats (<heart>, <heartbeat>, <server> tags)” in *SafeKit User's Guide*.

14.1.4 Configure the Replication Flow

If you have two network interfaces, define a dedicated replication flow by two IP addresses or names connecting both servers. Do not forget to set a heartbeat with ident="flow" on this network.

For an example, see “Dedicated Replication Network Example” in *SafeKit User's Guide*.

Refer to “File Replication (<rfs>, <replicated> tags)” in *SafeKit User's Guide*. 
**14.1.5 Configure Non-Replicated Files/Subdirectories**

Configure the module with not replicating a file or sub-directory in your replicated directory. Create/update this file or sub-directory on your primary server and check that there is no replication on the secondary.

For more information, refer to “File Replication (<rfs>, <replicated> tags)” in *SafeKit User's Guide*.

**14.1.6 Configure Several Replicated Directories**

Configure the module for replicating several directories. Update these directories on your primary server and check replication on the secondary.

Refer to “File Replication (<rfs>, <replicated> tags)” in *SafeKit User's Guide*. 
14.2 Farm Module Configuration Exercise

14.2.1 Install a Farm Module

If not already done, install and configure the generic module farm.safe.

14.2.2 Configure Load Balancing Rules

Configure userconfig.xml of the farm module with a load-balancing rule on port="9010" and filter="on_port".

Check the result using the mosaic demonstration (many loads from http://<virtual_ip_address>:9010/safekit/mosaic.html). Refer to “Tests of Farm Module” in the SafeKit User's Guide.

Change the load balancing rule with port="9010" and filter="on_addr" and check the result with mosaic.

For load balancing examples, see “TCP Load Balancing Example” in the SafeKit User's Guide.

For more information, refer to “Virtual IP Address (<vip> tag and subtags)” in the SafeKit User's Guide.

14.2.3 Configuring Load Balancing Powers per Server

In userconfig.xml of your farm module, change again the load balancing rule with port="9010" and filter="on_port". And give a power of 1 for one server and a power of 2 for the other server. Check the result using the mosaic demonstration.

For an example, see “TCP Load Balancing Example” in the SafeKit User's Guide.

For more information, refer to “<vip> Description” in SafeKit User's Guide.

14.2.4 Configure a Farm with Multiple Virtual IP Addresses

Add a new virtual IP address on the same network interface (modify userconfig.xml of the farm module). And run mosaic demonstration with this new virtual IP address.

For more information, refer to “<vip> Structure” in the SafeKit User's Guide.
14.2.5 Configuring a Farm of 3 Servers

If you have a 3rd server available, add it to your existing farm. Install SafeKit on it and install the farm module. Configure the farm module to include the 3rd server in load balancing. Check the result using the mosaic demonstration.

For an example, see “TCP Load Balancing Example” in the SafeKit User's Guide. For more information, refer to “Farm Topology (<farm>, <lan>, <node> tags)” in the SafeKit User's Guide.
14.3 Checkers Configuration Exercise

14.3.1 Configuring Software Error Detection (<errd>)

In userconfig.xml of the mirror module, configure <errd> to monitor a process and check that action is executed if the process is killed (note: you can use the command safekit kill processname). Refer to “Tests of Checkers Common to Mirror and Farm” in the SafeKit User's Guide.

Implement the demonstration module softerrd.safe that demonstrates the software error detection features. See “Software Error Detection Example with softerrd.safe” in the SafeKit User's Guide.

For more information, refer to “Software Error Detection (<errd>, <proc> tags)” in the SafeKit User's Guide.

14.3.2 Configuring a Ping Checker


For an example, see “Ping Checker Example” in the SafeKit User's Guide.

For more information, refer to “Ping Checker (<check>, <ping> tags)” in the SafeKit User's Guide.

14.3.3 Configuring a Custom Checker


For an example, see “Custom checker example” in the SafeKit User's Guide.
Evidian SafeKit

3-Nodes Replication
3-Nodes Replication Architecture

server1 = server2 = serverDR

STOP ≠ ALONE ≠ WAIT

Reintegration ≠ WAIT

SECOND = PRIM = SECOND
- configure the module on 3 servers with the template
  - 3nodesrepli_v2.safe in Windows
  - 3nodesrepli.safe in Linux
- 2 modules are created on 3 servers: 3nodesrepli, 3nodesrepli_spare
- 3nodesrepli = application module with automatic application restart
- 3nodesrepli_spare = replication to the disaster recovery
Click on a module (safe) for launching the wizard that installs and configures the module.

Select Node and Modules

- **Module Nodes**
  - Select the nodes on which you want to apply the module configuration
  - **Configure the module on this node**
    - server1 / Microsoft Windows Server 2012 R2 Standard Evaluation (64-bit) (6.3.9600) / SafeKit 7.2.0.13
    - server2 / Microsoft Windows Server 2012 R2 Standard Evaluation (64-bit) (6.3.9600) / SafeKit 7.2.0.13
    - serverDR / Microsoft Windows Server 2012 R2 Standard Evaluation (64-bit) (6.3.9600) / SafeKit 7.2.0.13
Configuration (4/5)

For a configuration with 3 nodes replication, the disaster recovery node (DR) must be identified:

Select the future DR node: server1 or server2 or serverDR and click here to assign its role.

For the first start, the node with the up-to-date replicated directories must be started as primary:

Select the node that must be started as primary: server1 or server2 and click here to start it.

Prim on server1?
The application module replicates "dir" from server1 to server2.

The spare module replicates "dir" from server2 to serverDR.

The operating mode is Normal (1/4).
server1 Fails: Automatic Failover on server2 (2/4)

stop command by admin or server1 failure

server1 server2 serverDR

STOP ALONE offline

offline offline DR Site

ALONE

offline

server DR Site

WAIT

server1 Fails: Automatic Failover on server2 (2/4)
server1 Restarts: Automatic Failback (3/4)

start command by admin

server1: Reintegration
server2: ALONE
serverDR: offline

DR Site: WAIT

server1: ALONE
server2: offline

by admin

start command
Return to Normal Operation (4/4)

- Use the `stopstart` command on `3nodesrepli/server2/PRIM` to return to (1/4)
- Note that the `swap` command is not working in this architecture
Failover to the Disaster Recovery Site (1/4)

This manual operation is exceptional and happens when server1 and server2 fail

activate the failover wizard in the web console

activate the failover wizard in the web console

This manual operation is exceptional and happens when server1 and server2 fail
Failover to the Disaster Recovery Site (2/4)

- Initiate the application failover on the disaster recovery node (DR).
- Prerequisites:
  - The application module is not reachable on the main nodes
  - The DR node is serverDR

Confirm
Failover to the Disaster Recovery Site (3/4)

Change the role of nodes  Step 1  Finish

Failover the application on the disaster recovery node (DR) set.
Click on Confirm to run: changerole -role FAILOVER -m 3node

Change the role of nodes  Step 1  Finish

SUCCESS
- Be careful, if the application module is restarted automatically at boot on server 1 and server 2, stop it immediately and keep only the DR site running
- The only way to correctly return to the execution on server 1 and server 2 is to execute the manual failback operation
Failback from the DR Site to the Main Nodes (1/5)

_activate the failback wizard in the web console_

server1
STOP

server2
STOP

serverDR
ALONE

Disaster Recovery Site
offline

SECOND
STOP

PRIM

WAIT

SECOND
STOP

PRIM
offline

offline

PRIM

SECOND
### Failback from the DR Site to the Main Nodes (2/5)

**Initiate the application failback to the main nodes.**

**Prerequisites:**
- The application failover procedure on the DR node has been applied before.
- The application module state on the DR node is **ALONE**.
- The application module state on the main nodes is **STOP** or **offline**.

The DR node is **serverDR**

Select the main node that is to become primary: 
- **server1**
- **server2**

**Change the role of nodes**

<table>
<thead>
<tr>
<th>Node</th>
<th>Role</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>server1</td>
<td>STOP</td>
<td>update</td>
</tr>
<tr>
<td>server3</td>
<td>STOP</td>
<td>not update</td>
</tr>
<tr>
<td>serverDR</td>
<td>ALONE</td>
<td>update</td>
</tr>
<tr>
<td>serverDR</td>
<td>offline</td>
<td></td>
</tr>
</tbody>
</table>

**Emergency procedures**
- Drift the cluster recovery node (DR)
- Power off the DR node
- Start the DR node
- Verify the application module state on the DR node
- Access the server DR
- Start the application module on the DR node
- Verify the application module state on the server DR
- Verify the application module state on the main nodes
- Verify the cluster status
- Ensure the cluster is running

**Server configuration**
- **server1**
- **server2**
- **serverDR**

---

© Evidian
Failback from the DR Site to the Main Nodes (3/5)

Set the application module offline on server2.
Click on Confirm to run: changerole -role NONE -m 3nodesrepli

Enable communication between the disaster recovery node and the main nodes.
Click on Confirm to run: changerole -cluster REJOIN
Failback from the DR Site to the Main Nodes (4/5)

Resynchronise data from the disaster recovery node to server1
Click on Confirm to run: changerole -role RESYNCH -m 3nodesrepli

Assign a disaster recovery (DR) node role to serverDR
First wait for the end of the resynchronisation, when the state is SECOND
Click on Confirm to run: changerole -role SPARE -m 3nodesrepli -start
Failback from the DR Site to the Main Nodes (5/5)

Assign a main node role to server2.
Click on Confirm to run: changerole -role MAIN -m 3nodesrepli -start.

Success.
The failover wizard on serverDR when server1 and server2 are down issues:

1. on serverDR: `safekit changerole -role FAILOVER -m 3nodesrepli -start`

The failback wizard from serverDR, with server1 as future primary issues:

1. on server2: `safekit changerole -role NONE -m 3nodesrepli`

2. on serverDR: `safekit changerole -cluster REJOIN`

3. on server1: `safekit changerole -role RESYNCH -m 3nodesrepli`

4. wait PRIM (serverDR 3nodesrepli) / SECOND (server1 3 nodesrepli)

5. on serverDR: `safekit changerole -role SPARE -m 3nodesrepli -start`

6. wait ALONE (server1 3nodesrepli) / offline (serverDR 3nodesrepli)

7. on server2: `safekit changerole -role MAIN -m 3nodesrepli -start`

6 – the system returns to normal operation PRIM-SECOND, PRIM-SECOND
### Command Lines (2/2)

**(AM, AM_spare) deployed: use AM in the commands, not AM_spare**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`safekit changerole role MAIN -m AM [-start [prim</td>
<td>second]]`</td>
</tr>
<tr>
<td><code>safekit changerole role SPARE -m AM [-start]</code></td>
<td>Sets the current node as the disaster recovery (DR) node for AM</td>
</tr>
<tr>
<td><code>safekit changerole role NONE -m AM</code></td>
<td>Sets the current node as not playing any role for AM</td>
</tr>
<tr>
<td><code>safekit changerole cluster ISOLATE</code></td>
<td>Isolates temporarily the current node and all its modules from its current cluster</td>
</tr>
<tr>
<td><code>safekit changerole cluster REJOIN</code></td>
<td>Re-inserts the current node and all its modules in its previous cluster</td>
</tr>
<tr>
<td><code>safekit changerole role FAILOVER -m AM [-start]</code></td>
<td>Performs a failover of the AM module on the DR node and calls ISOLATE command</td>
</tr>
<tr>
<td><code>safekit changerole role RESYNCH -m AM [-start]</code></td>
<td>Initiates a data resynchronization, as part of the failback procedure</td>
</tr>
</tbody>
</table>