New OpenESB console specifications: Draft V.01

Paul Perez 2012 Q3

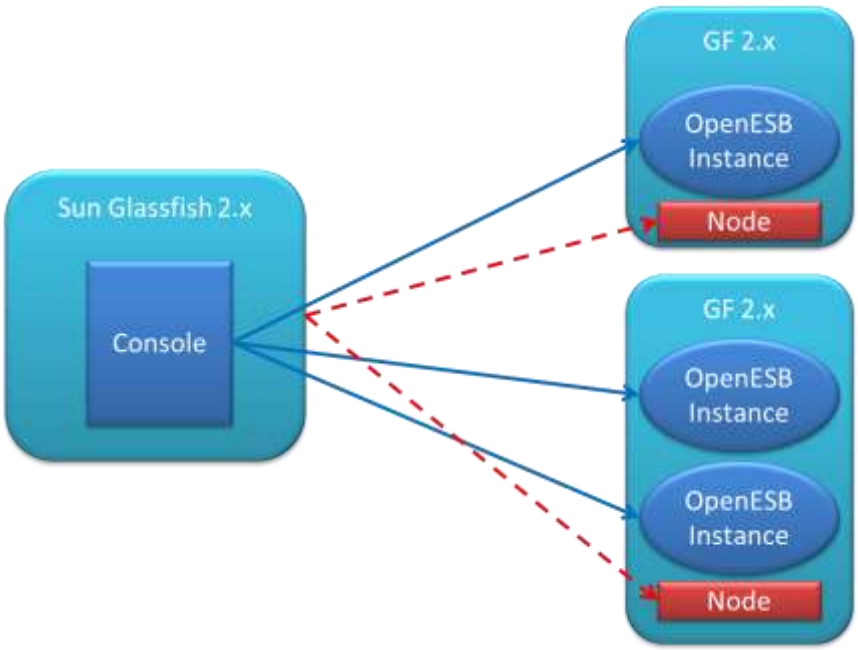
Preamble

Note to the reader: As French speaker, when writing this paper I had the choice between writing in perfect French or in an intermediate English language (with syntax and grammar errors). I chose to write it in English so the document is understandable by all. I apologise for the English faults and I beg my pardon to my English teacher in high-school Mrs Posteck. English reviewers are welcome.

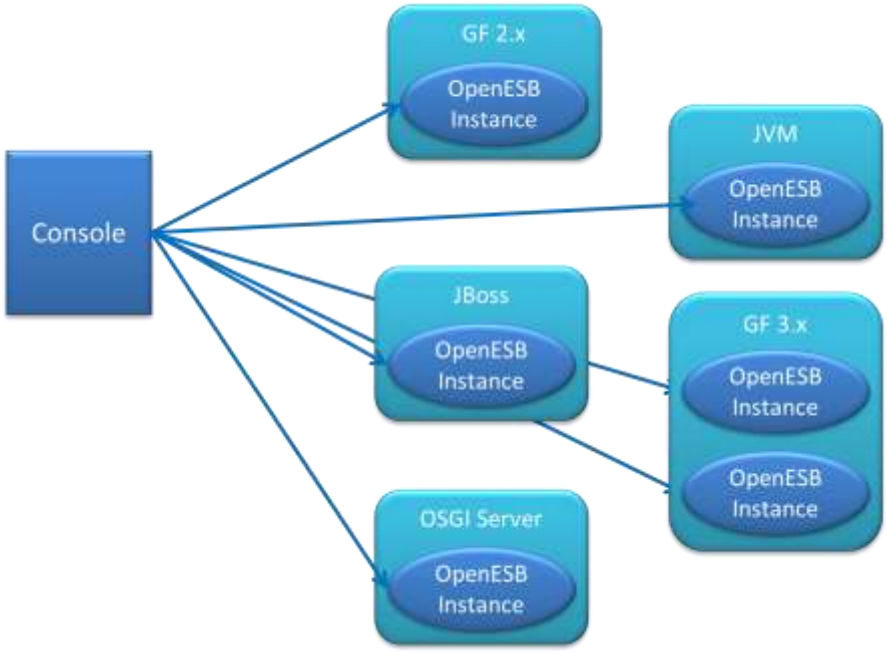
In this document, I tried to list the features required to propose to OpenESB users a reliable useful and professional console available on any platform and for any OpenESB container. The current console provided with OpenESB 2.2 is strongly tied with the application server Glassfish 2.2. This stops OpenESB developments, implementations and migration on other platforms like GFV£, Jboss or else. I'm not expecting compliment for this document but many remarks, comments and propositions. My objective is to start up functional specifications, architecture design and development as soon as possible in order to prepare OpenESB V3 issue. Please, try your best to be constructive and productive.

Introduction

OpenESB is a neutral platform for integration. Theoretically, this means it can work on any Java platform, on any applications server. This is the theory!! In practice, Sun Microsystem linked OpenESB with Glassfish 2.2. The best example is the admin console which is embedded in GF2 console. We can understand wanted to promote its application server through its ESB, however, this context becomes a burden when we wanted to migrate OpenESB to another platform. OpenESB is completely embedded in GF2 console and cluster deployment is relying on Glassfish Cluster infrastructure.



The objective of this document is to outline the main features for a new console that would be autonomous and independent. This new console must target any OpenESB instances regardless the instance container and provides multi-instances deployment and synchronisation.

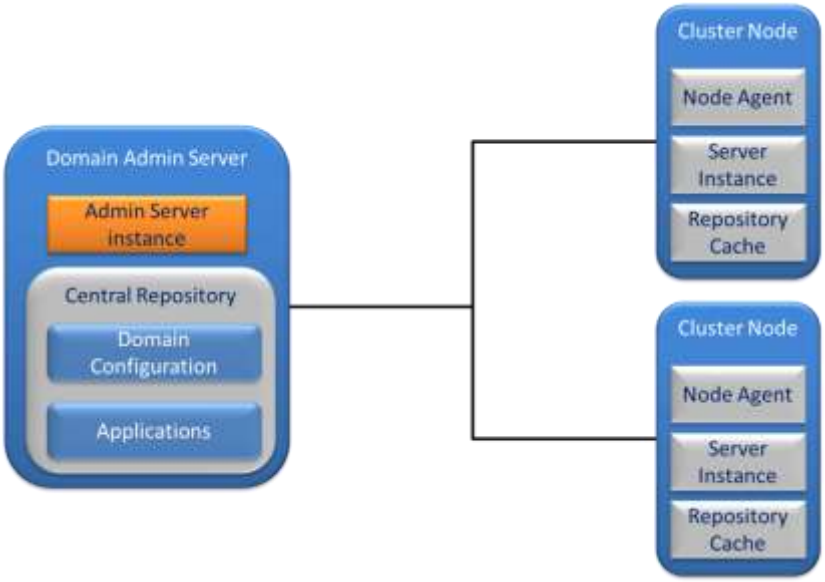


Community's long term roadmap plans to transform OpenESB to a neutral platform tools. So it has to develop a new console able to communicate directly with OpenESB instances regardless their container.

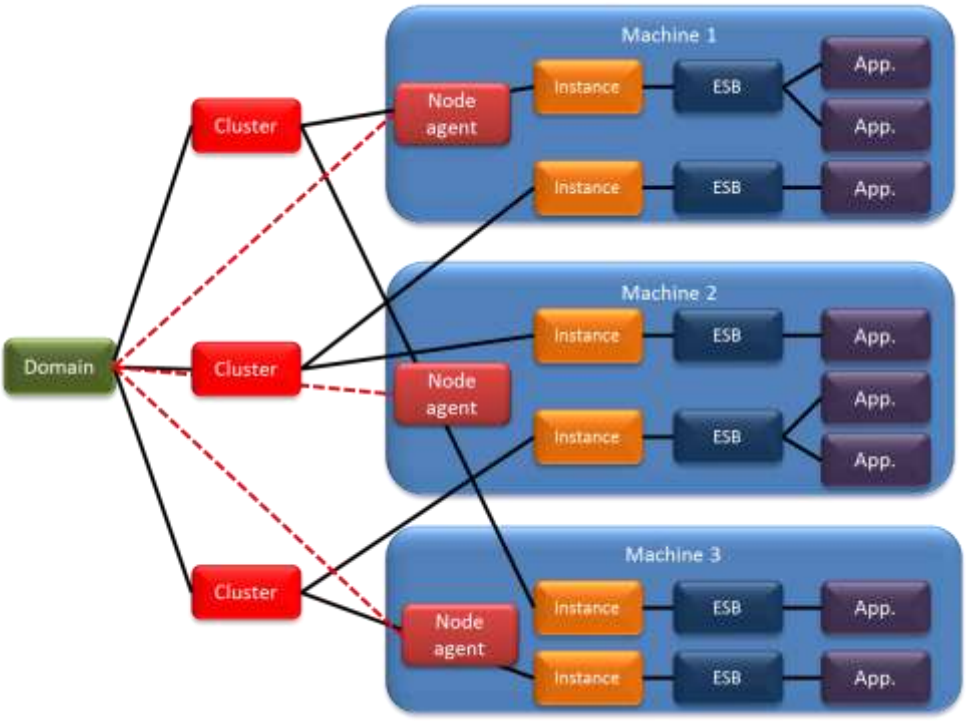
OpenESB JBI components and organisation.

Glassfish 2.x and Glassfish 3.x domains and clusters propose an understandable and easy to use organisation for components and applications management in multi instance context. Production

and support teams take advantage of Glassfish node agents to deploy components and applications on many nodes at once.



Basic Cluster organisation



Multi-node clusters

However, as explained in the introduction, we don't want to rely on Glassfish infrastructure anymore. Consequently, Glassfish domain, Node Agent must be removed from OpenESB world. We must create, design and implement a new deployment framework which provides developers, users, support and production with the same characteristics than with Glassfish V.2.x.

Domain-Group-Instance OpenESB hierarchy

In order to organise OpenESB multi-instances deployment, I propose a new three levels hierarchy which replaces Glassfish hierarchy Domain-Cluster-Node Agent and Instance. This new hierarchy is made up with Domain, Group and Instance.

Instances

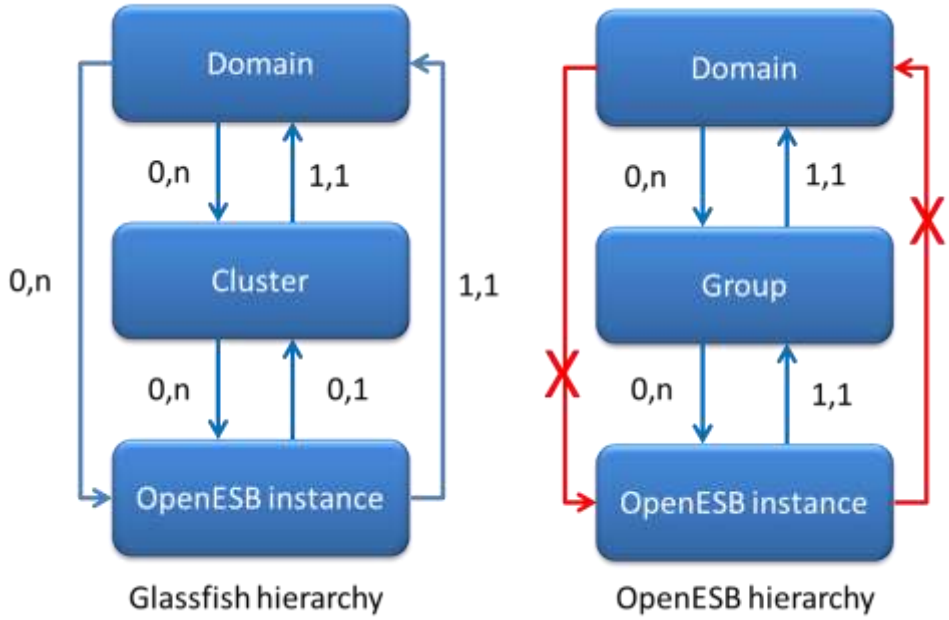
In the new OpenESB hierarchy, instance is the basic element of the new management organisation. There is no difference between GF2 instance and the instance in the new hierarchy except that the former is always linked with GF2 App Server as container. That's not the case in the new hierarchy.

Group

As designed in a Glassfish cluster, OpenESB instances can be put together to scale applications or provide redundancy and high availability. In the new hierarchy, a set of identical OpenESB instance is named a "**Group**". A group contains zero or more OpenESB instances. At the runtime, a single OpenESB instance will be monitored only if it belongs to a group[1]. Consequently, there is no instance without group.

Domain

"**Domain**" is the top entity in the hierarchy. A domain is an administrative entity used to match management organisation with "functional domains. Access to group configuration and monitoring must be validated by security policies defined at the domain level. This means that all groups belonging to a domain follow the security policies defined in the domain.



In the new hierarchies an OpenESB instance belongs to a group. The new console will relies on this hierarchy.

[1] I don't know if we can force users to create first a group before deploying an instance.

Console administrative task

Domain

Create a domain

From my point of view any one can create a domain. There is no entry ticket to do it. But after its creation a domain belongs to its creator or the administrator. During the creation process the future administrator defines:

- Where Domain repository will take place.
- Who is the administrator
- Access detail for the remote access to the admin domain.

From my point of view, the repository contains the following information:

- Users directory (LDAP or Else) I think that one directory per domain is required in order to avoid dependency between domain.
- A database like feature to store groups or domain detail and configuration.

Update a domain

A Domain is an administrative entity and any update is made at an administrative level only (New user, admin, etc…)

Delete a domain

We have to define what does is means to delete a domain. What do we do with instances and group belonging to the domain we want to delete?

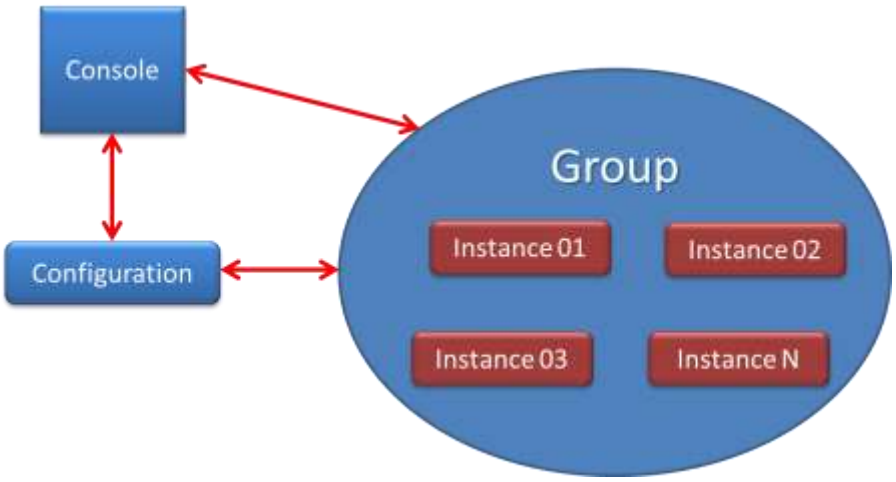
Users hierarchy in a domain

At the first glance I propose the Following roles:

- Domain administrator
  Manage a domain like an administrator. Create new group. New user
- Group administrator
  Manage a Group, Create new instances, configure the group, deploy application, create new viewer
- Viewer
  Allowed to view group configuration

Group

Group is the key concept in OpenESB. A group is the level where instance is configured, new resource, component and application are deployed. A group is defined by a configuration. A configuration represents all the parameters use by an OpenESB instance at the runtime.

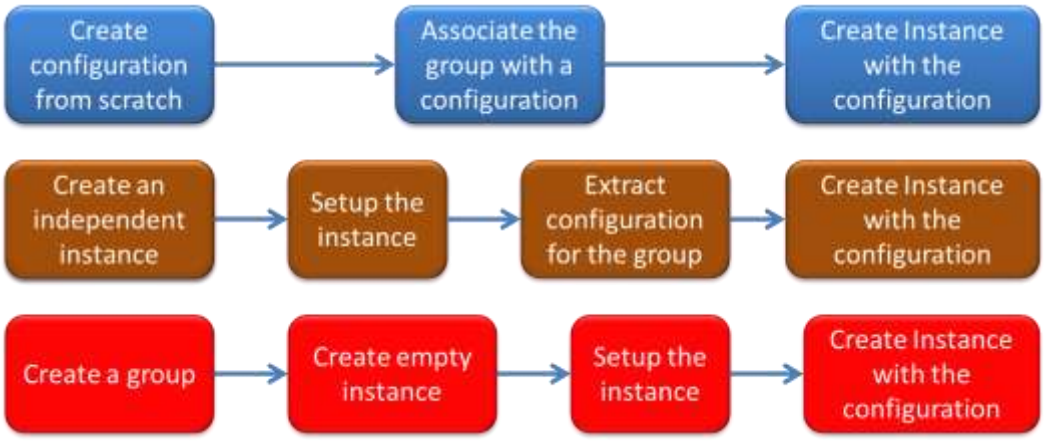


The instances belonging in a group are identical. Parameters defined in the configuration are:

- The components installed in the instance
- Component configuration
- Application installed in the instance
- Application configuration.
- All other parameters required at design time and runtime.

Create a group

How do we create a group is an interesting question we have to think about. At the first glance, I see two-three ways to do it.



1. The first way is to define a configuration from scratch. We associate a group with this configuration. Then Group administrator creates the instances from this configuration. I picture this way to configure the group in a production environment when the final configuration has been clearly defined by development and production teams. The configuration can be generated by a script or an external tool.
2. The second way is a most natural one. An external instance could be defined and setup during the development stage, by another project or and external provider. Then this instance and its configuration serve as model to the instance of group.

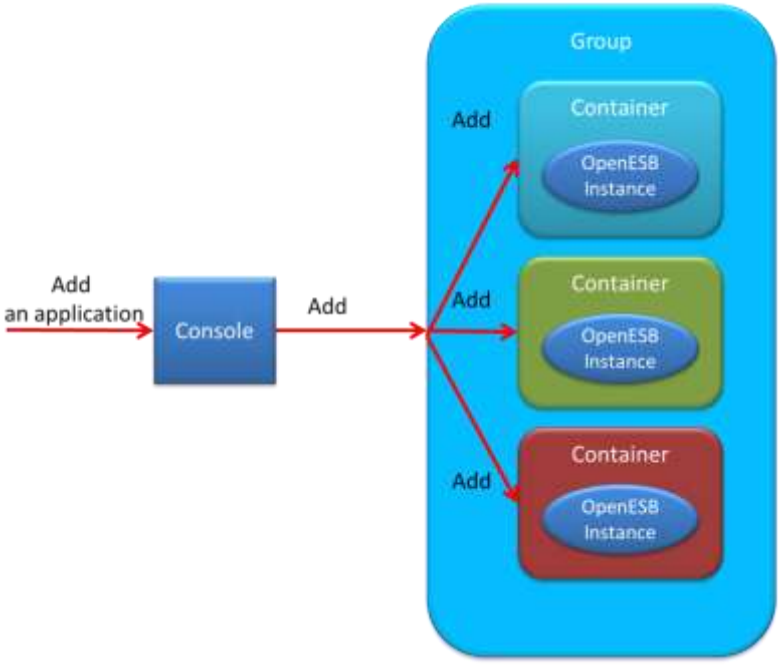
3. The last way starts by creating an empty Group and an instance (We have to define what the default configuration for this instance is). The group administrator sets up the instance. The instance defines the default configuration for the group.

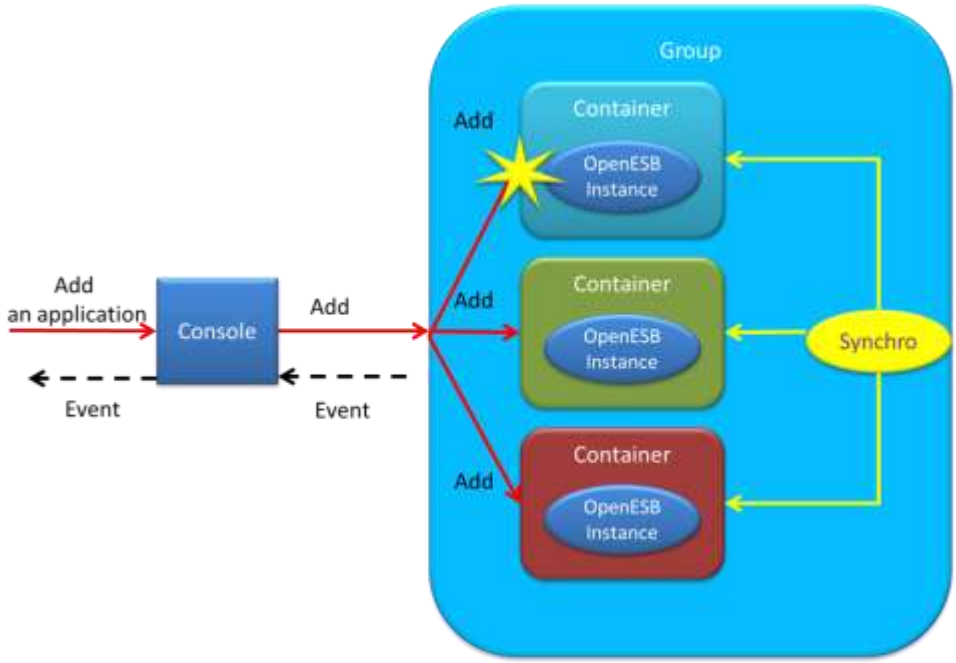
I'm pretty sure that technical constraints will stop us to propose these three different ways at the same time. But we have to see what are the most accurate for the administrator and what the easiest ways are for console designers and developers

Modify a group

Group modifications means adding a new application or component, changing framework, component, application configurations. Since we don't want to use Glassfish Cluster features to synchronise instances of the same group, we have to rely on independent framework for synchronising all the instances of a group.

The same framework MUST be able to fix or to alert the group administrator when an error occurs during the synchronisation.





The most dangerous issue in a group is a desynchronisation between OpenESB instances. After a deployment, a synchronisation control must be done. During the runtime, this synchronisation must be controlled too. If a desynchronisation occurs during the installation and the runtime, an alert must be sent to the group administrator and a manual or automatic fix must be run.

This is the critical point of this monitoring architecture. If we are not able to guaranty this synchronisation between the instances of the same group, this solution will not be reliable and be cancelled.

Delete a group

To be discussed

Instance

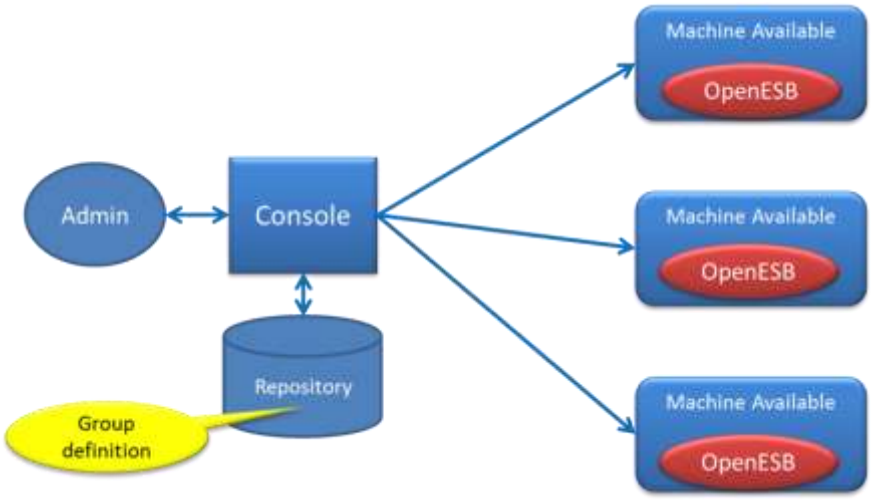
Create an instance

It would be nice if an OpenESB instance could be created directly from a console without using the container infrastructure or may be instance creation means create the instance plus its container. This approach would be very useful in a cloud environment where we can create many OpenESB instances.

Scenario example

1. The administrator select N machine available to deploy OpenESB. (A kind of agent on available machine will certainly be required)
2. He selects the group of the new instances. The group contains the entire configuration of OpenESB instances.
3. The administrator lunches the creation.
4. After a while, the new instances appear in the selected group hierarchy.

For the moment I have no precise idea how to do it. May be tools like Zookeeper, Chief or Puppet or any useful administrative tools for remote deployment can be a candidate.



Regarding Instance creation, I have additional questioning about instance creation. As explained above, OpenESB instances in the same group will be exactly the same, but what about the container. Let's suppose we have two machines where we want to set up and run OpenESB instances (one per machine). The first machine has 32 Gb memory and the second just 8 Gb. How do we take into account this difference? For the moment I have no idea and any input would be welcome.

Modify an instance

Regarding the organisation defined above, a singleOpenESB instance cannot modify.

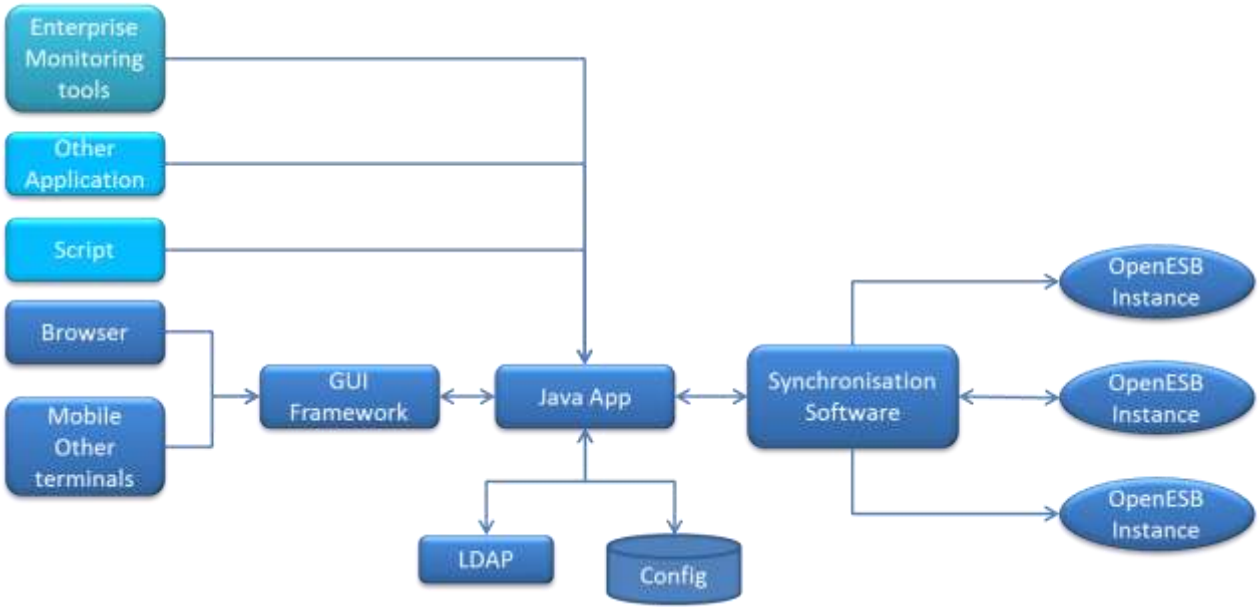
Delete an instance

Deleting an instance during the runtime can impact process consistency and delete message waiting for processing. This issue already exists with OpenESB 2.2. From GF 2,2 console, we cannot delete an instance without shutdown. The same behaviour must be duplicated in the new console.

Additional study on that topic must be made to provide the most consistent process to delete an instance

Console architecture

At the first glance, I picture the console architecture as follows. Our discussion and our comment are here to affine and precise what can be the best architecture for the console



Console architecture at the first glance

Console clients

We can design many clients for the console.

Browser

The main and the most important is the browser used by 99% of production teams and developers. There are many frameworks that can be used to interact with the browser and lot of religious wars when we ask: "Which one is the best". Please feel free to send your feedback on Web framework.

Mobile

In Option: I imagine that shortly, domain and group administrator would be very happy to get a control on the groups and domains through their IPad or any other tab. Just have to think about it

Script

In GF V2 Asadmin is used to execute script. Scripts are mainly used in production. Similar feature must be defined for the new console

Other applications

Administrative command can be sent from other applications. An application to application channel must be defined by using Web services SOAP or Rest based.

Enterprise monitoring tools

My idea is not to allow enterprise monitoring tools like Opsview, Nagios or else to deploy OpenESB instance. However, alert or event like Instances desynchronisation may be catch by monitoring tools.

Console App and Synchronisation (the name is not very good)

It is the core of the project with the Synchronisation. This application can be an independent application running on a simple JVM, or embedded in a WAR or EAR to run on an application server. In my first plan, I propose to separate Console and Synchronisation because each application has its own purpose and scope. Console deals with admin features like access control, group configuration. Synchronisation send configuration to the instances.

Regarding the environment, Synchronisation framework can be different and for obvious reason, it has to be decoupled from the console.

LDAP

A LDAP access to store users' details and rights is required to secure console accesses. Access security must be designed too. Remote communication from remote terminal to the console must be safe. SSL or stronger encryption can be used. (TBD)

Configuration

Domain and Group configuration must persist to a console crash, so they have to be store somewhere. The simple technic would be to use a simple database; other persistence devices like Object DB, persistent cache may be used too.

Conclusion

This 12 pages document is a first draft and a many days work task. Its aim is to start a discussion on the new OpenESB console. I tried to list the main issue we will face. I'm waiting for your comments questions, remarks or other document on the same topic.