RSD GLASS 3.4.4

Multi-Tenant Edition

SaaS Guide

English

RSD-000044-EN-04

Preface

RSD GLASS® is a software package property of RSD SA - Geneva, Switzerland that cannot be used without license.

RSD reserves the right to make any modifications to this product and to the corresponding documentation without prior notice or advice.

Trademarks and abbreviations

All brand and product names quoted in this publication are trademarks or registered trademarks of their respective holders.

Manual: RSD GLASS® SaaS Guide version 3.4.4

Contents

Release Information

This 3.4.4 release of RSD GLASS adds the following functionality to RSD GLASS 3.4:

- The IT admin can now manage Role and Permission definitions by "business family" from RSD Admin Center.
- Role and Permission definitions are now part of a dedicated SQL schema.
- Each tenant is now using a Role and Permission datasource. A Role and Permission datasource can be shared by several tenants.
- RSD Admin Center now supports OAuth and is used to manage Role and Permission definitions in any role schema.
- RSD GLASS now uses Quartz 2.2 for scheduled actions. Migration of data is required.

Known issue that does not prevent the product from working: hibernate error messages in log when starting Policy Manager.

Migration steps

Step 1: Migrate Zookeeper configuration from 3.4.3 to 3.4.4. See How to migrate an RSD GLASS Cloud configuration page 13.

Step 2: Stop applications.

Step 3: Install new version of the applications (war files version 3.4.4).

Step 4: Execute SQL migration scripts.

Step 5: Restart applications.

RSD GLASS® SaaS Solution Architecture

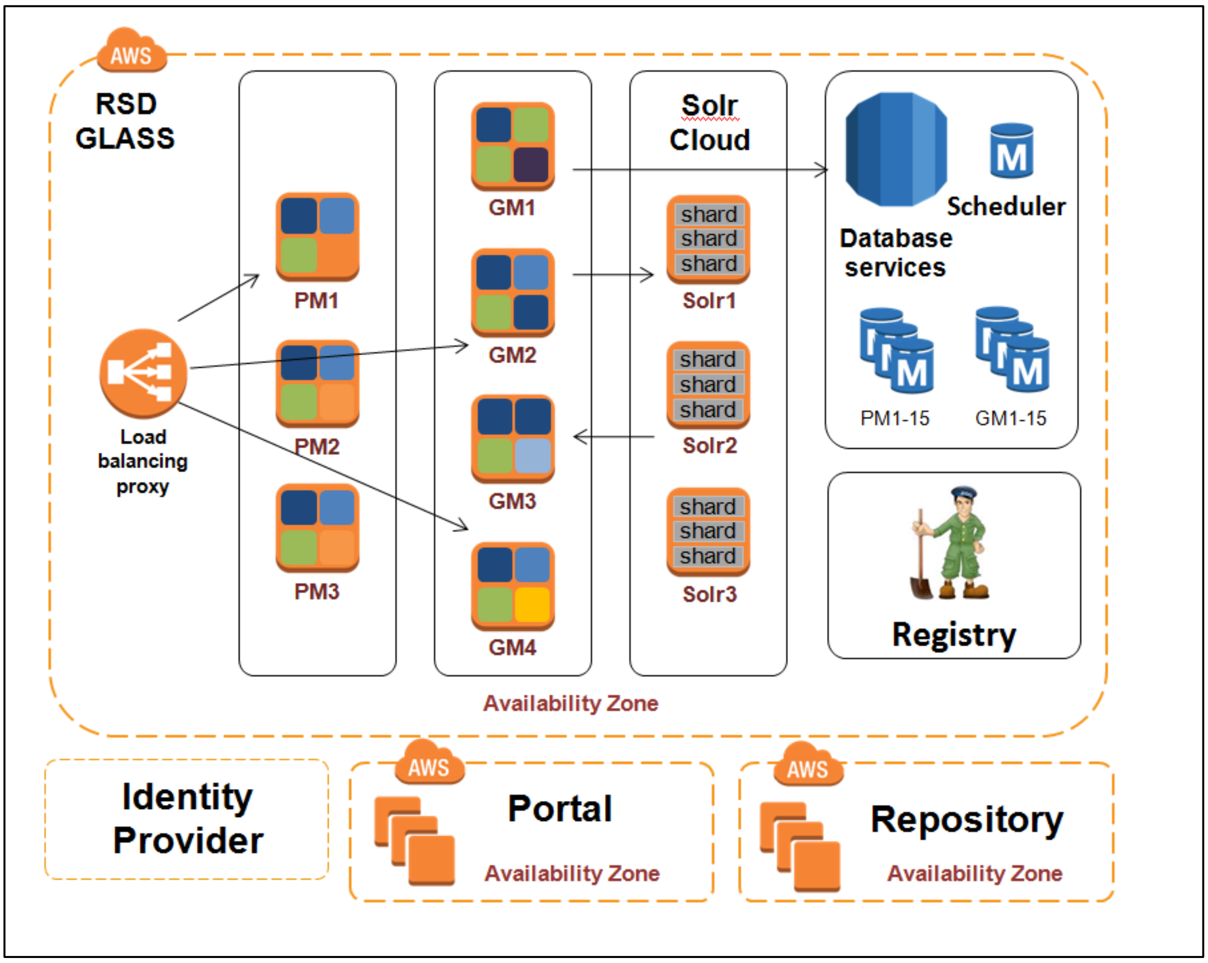
Introduction

Starting with version 3.4, RSD GLASS® offers a SaaS solution and supports multi-tenancy, i.e. the ability for a single instance of the product to support multiple users from several companies ('tenants').

This helps storage service providers reduce the cost of RSD GLASS® deployment for their customers.

Architecture basics

The architecture options that the RSD GLASS® SaaS solution is based on are hereunder described. The main RSD GLASS® components are 'tenant-aware' so that each GLASS instance can support multiple tenants.



Cloud-oriented solution

The RSD GLASS® SaaS solution is especially of interest when RSD GLASS® is installed on a cloud computing platform, such as Amazon Web Services (AWS). All components of RSD GLASS® are installed in the cloud to offer fault-tolerance, high availability and distributed capabilities. Relational Database Services are used to set up, operate, and scale RSD GLASS® relational databases.

Similarly, SolrCloud offers cloud-wide distributed indexing and search capabilities.

Data architecture

The data architecture of the solution implements the **'shared database - separate schemas'** paradigm. Multiple tenants are housed in the same RSD GLASS® database, and each tenant has its own set of tables that are grouped into a schema created specifically for it. Each tenant can access its own data and benefits from logical data isolation.

This architecture enables the database administrator to virtualize database access, since databases are not dedicated to tenants. A small manageable number of databases and database tables are involved, which makes database management easier. This proves especially handy in a cloud environment, where Relational Database Services (RDS) enable the administrator to quickly get new DB instances and control the behavior of the databases that he manages. In this context, a tenant is hosted on exactly one RDS, and a RDS may host multiple tenants.

Web access and load balancing

Tenants access RSD GLASS® data using a unique URL, for example: http://crystal.mycompany.com, or http://mycompany.com/GLASS.

A load balancer (such as HAProxy) spreads requests across multiple PM or GM instances, using simple scheduling algorithms (round robin). It achieves 'sticky sessions': it redirects requests to the same PM or GM instance throughout a user’s session.

Clustering and caching make it easier for PM and GM instances to communicate and share the workload; an in-memory data grid software product (Hazelcast) complies with this purpose.

For SolrCloud, partitioning indexes in shards helps to better spread the workload.

Security

Security is based on an identity provider that delivers authentication, authorization and single sign-on, based on the SAML and OAuth 2.0 standards.

After the end user is authenticated, he is provided with an access token which is revalidated by the PM or GM instance. At the same time, the RSD GLASS® instances obtain the roles and the tenant-ID associated with the end user.

RSD GLASS® instances are linked to the repository platform (e.g. Enterprise File Sync and Share Cloud) whose access control is similarly based on SAML and OAuth protocols.

Provisioning and Registry

Tenants must be provisioned in a dedicated Registry (Apache ZooKeeper) which provides distributed configuration services. RSD GLASS® components are not impacted by provisioning because they only acknowledge roles (not users); from the Registry, they retrieve tenant information (schema, SolrCloud collection...) and general RSD GLASS® configuration parameters (datasources...). The Registry is also used for distributed synchronization (locking, queuing) and cluster topology. It complies with clustering and high availability requirements.

Provisioning consists in setting tenant-related information in the identity provider database and in the Registry.

When GM starts up, it retrieves the solution configuration in the Registry. It also accesses the Registry when required, to get any incoming new tenant's configuration. SolrCloud also interacts with the Registry.

Auditing

RSD GLASS® Audit is not involved in the SaaS support. There is one common and single log file per PM or GM instance. Log entries store the tenant-ID.

Limitations for RSD GLASS 3.4.4

RSD GLASS® for Physical records do not support multiple tenants. Only Policy Manager and Governance Manager are tenant-aware.

Public view and administration of the servlet (PM & GM) do not support multiple tenants (no login mechanism and as a consequence no way to find the right tenant ID).

Tenants cannot be deactivated once defined.

Additional components

This is a list of the components involved in the SaaS solution, in addition to the RSD GLASS® components that are shipped in the mono-tenant case:

- application server: Tomcat 7 with Java JRE 7
- load balancer: **HAProxy**
- in-memory data grid product: **Hazelcast** (embedded in RSD GLASS® as part of the libraries)
- identity provider supporting SAML and OAuth: current version of RSD GLASS® only supports **OpenAM** (a.k.a. OpenSSO)
- registry and distributed configuration services: **Apache ZooKeeper** (recommended version: 3.4.6)
- distributed indexing and search capabilities: **SolrCloud** (recommended version: 4.10)
- multi-tenant job scheduling tool: **Quartz** (embedded in RSD GLASS® as part of the libraries)
- **Relational Database Services** as supplied by the cloud computing platform (e.g., RDS for MySQL 5.6.17+, or RDS for Oracle, etc.)
- repository platform: only **EFSS** from Perceptive Software is supported
- content analysis: **Apache Tika** (recommended version: 3.1.0).

Supported web browsers are:

- **Chrome**: version 39 and above
- **Firefox**: version 33 and above
- **Internet Explorer** 11: December 2013 version and above.

Note: it is not the responsibility of RSD to deliver, install and configure these applications or products. We do however recommend that you use the SolrCloud package delivered by RSD as there are specific configurations that need to be set in the *solrconfig.xml* and *schema.xml* files.

The Registry as a configuration tool

The Registry is used as an administration tool to define the configuration of the solution.

It is assumed that distributed configuration services are provided by the **Apache ZooKeeper** software product.

Except for testing purposes, it is recommended that at least three ZooKeeper servers be clustered for high availability and quorum-based commit voting. On each ZooKeeper instance, you could have an Exhibitor Web App (Netflix), to manage the znodes. Alternatively you could manage the nodes using zkCli.sh: the ZooKeeper commandline tool packaged in the bin directory of each ZooKeeper instance.

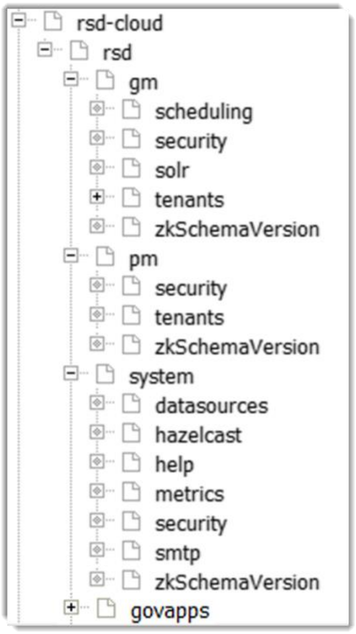
Structure

The ZooKeeper configuration structure for RSD GLASS® is based on 4 levels:

- **system level** (common properties for all application instances)
- **gm level** (define tenants configuration and specific properties for GM instance)
- **pm level** (define tenants configuration and specific properties for PM instance)
- **govapps level** (define tenants configuration and specific properties for Governace Apps instance)
- **adminCenter level** (define tenants configuration and specific properties for RSD Admin Center instance)

As an example, the figure opposite illustrates this structure:

- the root path configured in glass.properties is rsd-cloud/rsd
- the GLASS cloud configuration is located under the rsd directory. Two tenants are defined: TENANT_A and TENANT_B on each application.



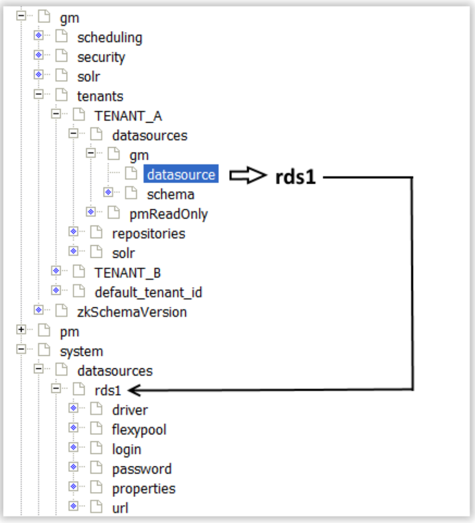
Convention for datasources

Datasources are defined as specific nodes in the **'datasources' subdirectory** of the **'system'** directory. The properties for this datasource are located under this node.

For each new tenant three folders should be created called **gm**, **pm** and **pmReadOnly** that should refer to the datasource and the schema.

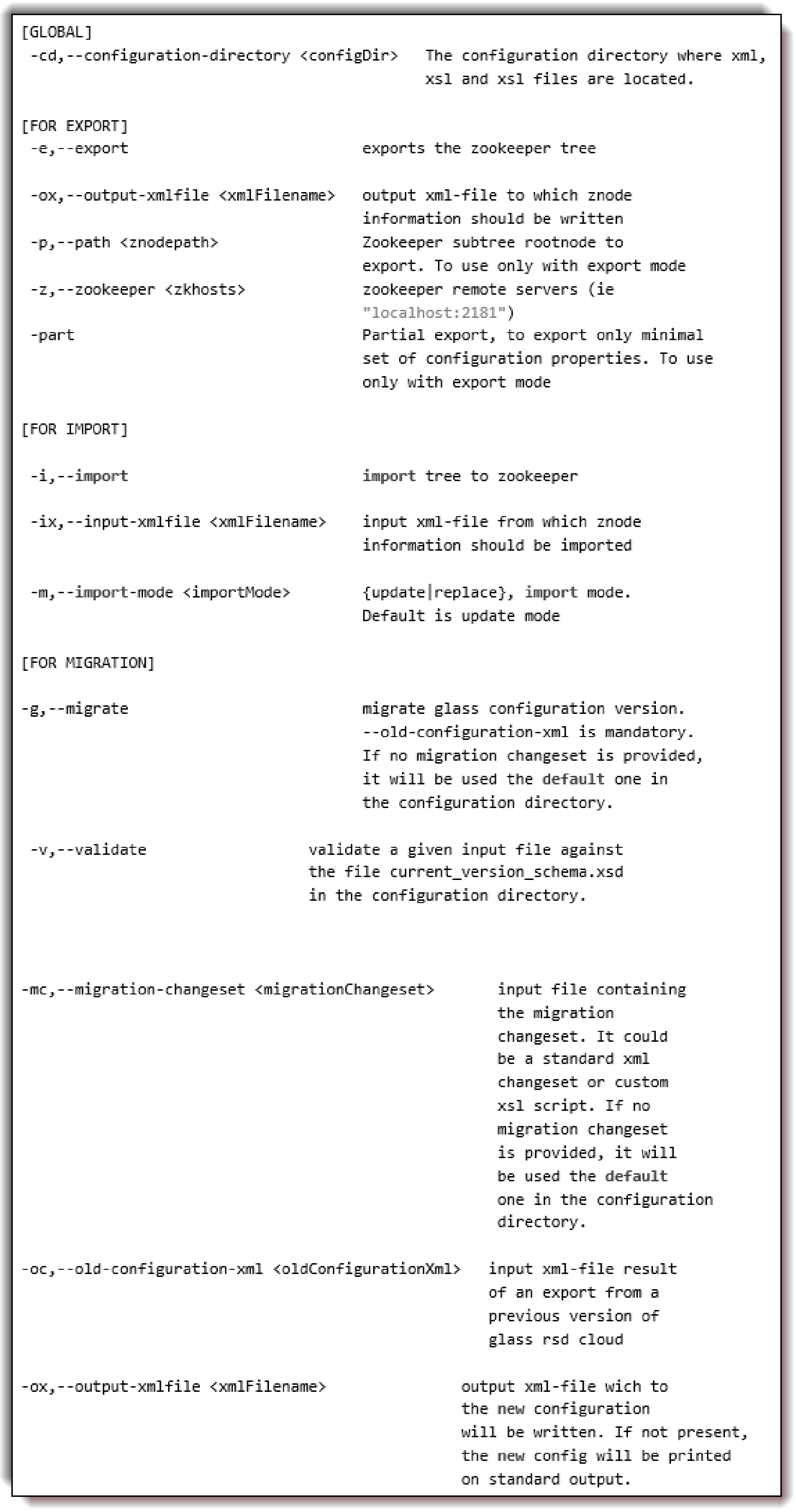
Tenants can refer to an existing datasource by setting the value of their own **'datasource'** node to the name of the datasource node as defined in the **'system'** directory.

In the figure opposite, tenant TENANT_A's datasource value of 'gm' refers to the 'rds1' datasource already defined in system/datasources.



ZooKeeper provisioning using RSDGlassRegUtil

To populate the ZooKeeper configuration from scratch, including adding as many tenants as required, use the RSDGlassRegUtil tool. This tool may also be used to migrate a previous RSD GLASS Cloud configuration. It is contained in a .jar file and is used as follows:



```
[GLOBAL]
 -cd,--configuration-directory <configDir>   The configuration directory where xml,
                                             xsl and xsl files are located.

[FOR EXPORT]
 -e,--export                          exports the zookeeper tree

 -ox,--output-xmlfile <xmlFilename>   output xml-file to which znode
                                      information should be written
 -p,--path <znodepath>                Zookeeper subtree rootnode to
                                      export. To use only with export mode
 -z,--zookeeper <zkhosts>             zookeeper remote servers (ie
                                      "localhost:2181")
 -part                                Partial export, to export only minimal
                                      set of configuration properties. To use
                                      only with export mode

[FOR IMPORT]

 -i,--import                          import tree to zookeeper

 -ix,--input-xmlfile <xmlFilename>    input xml-file from which znode
                                      information should be imported

 -m,--import-mode <importMode>        {update|replace}, import mode.
                                      Default is update mode

[FOR MIGRATION]

-g,--migrate                          migrate glass configuration version.
                                      --old-configuration-xml is mandatory.
                                      If no migration changeset is provided,
                                      it will be used the default one in
                                      the configuration directory.

 -v,--validate                  validate a given input file against
                                the file current_version_schema.xsd
                                in the configuration directory.



-mc,--migration-changeset <migrationChangeset>      input file containing
                                                    the migration
                                                    changeset. It could
                                                    be a standard xml
                                                    changeset or custom
                                                    xsl script. If no
                                                    migration changeset
                                                    is provided, it will
                                                    be used the default
                                                    one in the configuration
                                                    directory.

-oc,--old-configuration-xml <oldConfigurationXml>  input xml-file result
                                                   of an export from a
                                                   previous version of
                                                   glass rsd cloud

-ox,--output-xmlfile <xmlFilename>              output xml-file wich to
                                                the new configuration
                                                will be written. If not present,
                                                the new config will be printed
                                                on standard output.
```

There is also a **regutil.sh** shell script to facilitate using this jar.

Configuration of branches

The layout of the tree is managed separately for each branch.

There is a branch per application (**pm**, **gm**, **govapps**) and for the **system** configuration.

Each branch has its corresponding configuration directory for the --configuration-directory option of RSDGlassRegUtil.

Import, export and migration processes should be performed branch by branch.

Examples

```
#Export /rsd-cloud/DEV/rsd/system Zookeeper namespace:

./regutil.sh --configuration-directory zk/config-system-3.4.3.0
--zookeeper example.com:2181 --export -path /rsd-cloud/DEV/rsd/system
--output-xmlfile system-conf.xml

#Validate

./regutil.sh --configuration-directory zk/config-system-3.4.3.0
--validate --input-xml-file system-conf.xml

#Import updating the existing configuration with dev-conf.xml (Note that
Zookeeper namespace is specified as the root element of the xml document
to import ex: <root path="/rsd-cloud/DEV/rsd/system">)

./regutil.sh --configuration-directory zk/config-system-3.4.3.0
--zookeeper example.com:2181 --import --input-xml-file system-conf.xml

#Import replacing the existing configuration, using the --import-mode
replace command line option

./regutil.sh --configuration-directory zk/config-system-3.4.3.0
--zookeeper example.com:2181 --import --import-mode replace --input-xml-
file system-conf.xml
```

ZooKeeper provisioning

Installing a new branch

(mandatory) full template loading:

- Make a copy of the *current_version_template_full.xml* file that matches the version of the RSD Application that is being deployed (for instance, *zk/config-pm-3.4.3.0-SNAPSHOT/current_version_template_full.xml* for RSD GLASS Policy Manager version 3.4.3).
- Edit the file then review and modify all the properties that contain ##MODIFY_ME##. Update these platform-specific properties.
- Pay particular attention to the 'tenants' section of the **pm** and **gm** branches and the datasources section of the **system** branch: define the initial set of tenants for your platform (one 'tenant' node per tenant, can be 0 tenants to many).
- Launch RSDGlassRegUtil with the following parameters to populate ZooKeeper:

  ```
  --configuration-directory zk/config-pm-3.4.3.0-SNAPSHOT --import --
  input-xml-file "copy_of_pm_current_version_template_full.xml" -z
  zookeekerhost:zookeeperport
  ```

- Follow the same instructions for the **system** branch which is required by all other branches.

Example

```
#unzip the latest RSDGlassRegUtil to ./tool
unzip -q RSDGlassRegUtil-{1.0.0.1 or later}-util.zip -d tool
#unzip the required configDirectories
unzip -q RSDGlass-{version}-zkConfigDirectories.zip
unzip -q RSDGlassCommon-{version}-zkConfigDirectories.zip
unzip -q RSDGLASSGovernanceApps-{version}-zkConfigDirectories.zip
unzip -q RSDGlassPolicyManager-{version}-zkConfigDirectories.zip
```

```
ls ./zk
## import system configuration
#create a copy of the template
cp zk/config-system-3.4.3.0-SNAPSHOT/current_version_template_full.xml
system-3.4.3-initial.xml
#edit it replacing all the ##MODIFY_ME## values, and reviewing the other
values
vi system-3.4.3-initial.xml
#validate the modifications
tool/regutil.sh --configuration-directory zk/config-system-3.4.3.0-
SNAPSHOT --validate --input-xml-file system-3.4.3-initial.xml
#import it
tool/regutil.sh --configuration-directory zk/config-system-3.4.3.0-
SNAPSHOT --import --zookeeper localhost:62181 --input-xml-file system-
3.4.3-initial.xml
#repeat the copy, edit, import steps for each required application's
configuration directory
# zk/config-system-3.4.3.0-SNAPSHOT
# zk/config-pm-3.4.3.0-SNAPSHOT
# zk/config-gm-3.4.3.0-SNAPSHOT
# zk/config-govapps-3.4.3.0-SNAPSHOT
```

(optional) adding tenants later:

- Copy the *current_version_template_tenants.xml* file that matches the version of the RSD GLASS applications that are deployed on your platform (for instance, *zk/config-pm-3.4.3.0-SNAPSHOT/tenants/current_version_template_tenants.xml* for RSD GLASS Policy Manager version 3.4.3).
- The process is similar to the full template load except that the template file only contains tenant elements.
- Define as many 'tenant' nodes as you need.
- For each tenant, review and modify all the properties that contain ##MODIFY_ME##, as the values for these are tenant-specific.
- Launch RSDGlassRegUtil with the following parameters to populate ZooKeeper:

  ```
  --configuration-directory zk/config-pm-3.4.3.0-SNAPSHOT/tenants --
  import --input-xml-file
  copy_of_pm_current_version_template_tenants.xm -z
  zookeekerhost:zookeeperport
  ```
- Follow the same instructions for RSD GLASS Governance Manager and RSD GLASS Goverance Apps.

Example

```
#unzip the latest RSDGlassRegUtil to ./tool
unzip -q RSDGlassRegUtil-{1.0.0.1 or later}-util.zip -d tool
#unzip the required configDirectories
unzip -q RSDGlass-{version}-zkConfigDirectories.zip
unzip -q RSDGlassCommon-{version}-zkConfigDirectories.zip
unzip -q RSDGLASSGovernanceApps-{version}-zkConfigDirectories.zip
unzip -q RSDGlassPolicyManager-{version}-zkConfigDirectories.zip
ls ./zk
## add RSD Glass Policy Manager tenants
#create a copy of the tenants template
cp zk/config-pm-{version}/tenants/current_version_template_tenants.xml
pm-3.4.3-new-tenants.xml
#edit it replacing all the ##MODIFY_ME## values, and reviewing the other
values
vi pm-3.4.3-new-tenants.xml
#validate the new tenants
tool/regutil.sh --configuration-directory zk/config-pm-{version}/tenants
--validate --input-xml-file pm-3.4.3-new-tenants.xml
#import the new tenants
```

```
tool/regutil.sh --configuration-directory zk/config-pm-{version}/tenants
--import --zookeeper localhost:62181 --input-xml-file pm-3.4.3-new-
tenants.xml
## repeat for the following configuration branches as required...
# zk/config-pm-{version}/tenants
# zk/config-gm-{version}/tenants
# zk/config-govapps-{version}/tenants
# Note: existing tenants may also be exported for modification,
validation and import as above
tool/regutil.sh --configuration-directory zk/config-pm-{version}/tenants
--export --zookeeper localhost:62181 --path /somewhere/pm/tenants --
output-xmlfile pm-
```

How to migrate an RSD GLASS Cloud configuration

In this example, a configuration for version 3.4.3 will be migrated to the version 3.4.4. For demonstration, the ZooKeeper host is running on localhost:2181, the ZooKeeper configuration root path is */rsd-cloud/dca* and the long version of arguments is used..

1. Unzip the last version of *RSDGlassRegUtil*-{version}*-util.zip*:

```
#wget http://rsd-
sources.rsd.com:8081/nexus/content/repositories/snapshots/com/rsd/glass/RSDGlassR
egUtil/1.0.0.1-SNAPSHOT/RSDGlassRegUtil-1.0.0.1-20150305.080736-2-util.zip
unzip RSDGlassRegUtil-1.0.0.0-20150305.080736-2-util.zip -d RSDGlassRegUtil
cd RSDGlassRegUtil
```

2. Export the current configuration with the configuration directory corresponding to the current version:

```
./regutil.sh --configuration-directory ./glassApps3.4.3 --export --zookeeper
localhost:2181 --output-xmlfile config3.4.3.xml --path /rsd-3.4.3
```

3. Validate the current configuration. Now try to validate the file *config3.4.3.xml* against the 3.4.4 schema. As explained above, the tool will use *current_version_schema.xsd* in the directory migration.

```
./regutil.sh --configuration-directory ./migration --validate --input-xml-
file config3.4.3.xml
```

4. Proceed with migration:
   - If there is a *migrate.sh* in the config directory:

```
./migration/migrate.sh ./regutil.sh config3.4.3.xml config3.4.4.xml
```

   - If there is only one *migration_changeset.xml* in the config directory:

```
./regutil.sh --configuration-directory ./migration --migrate --old-
configuration-xml config3.4.3.xml --output-xmlfile config3.4.4.xml
```

   - If none of the above exist then no migration is required for this branch of the configuration.

5. Verify the migration by comparing the 2 files, to examine the changes made:

```
diff config3.4.3.xml config3.4.4.xml
```

6. Verify that there are no ##MODIFY_ME## values in the migrated xml file.
7. Validate the migrated xml file:

```
./regutil.sh --configuration-directory ./migration --validate --input-xml-
file config3.4.4.xml
[should pass]
```

8. Now import the new configuration into ZooKeeper.

   Note: the tool will upload the new configuration in the path defined in the header of the xml.

   Because the root element's path is the same, it will overwrite the old configuration. You can change the root path to create a new tree before proceeding with the import.

9. Verify that this command only modified the path and revalidate it.

10. If you don't change the root path then you should add the "--import-mode replace" command line option to the following command.

```
./regutil.sh --configuration-directory ./migration --import --input-xmlfile
config3.4.3.xml --zookeeper localhost:2181
```

Configuration Directory Structure

A configuration directory must be given as input to the RSDGlassRegUtil command line tool.

With the release of RSD Cloud 3.4.4 (Governance Manager, Policy Manager, Governace Apps) 4 zip files will be released containing the configuration directories for the branches of the configuration.

There are 4 branches (**system, pm, gm, govapps**). This allows the schema and the migration to evolve separately for each branch of the configuration.

Each configuration directory has the same structure with fixed file names and contains the structure of the configuration for the current release version and also the information on changes from last release.

current_version_schema.xsd

- This is the model of the current release configuration. It is used if a given xml contains a valid configuration for this version of the application (Governance Manager, Policy Manager or Governance Apps).

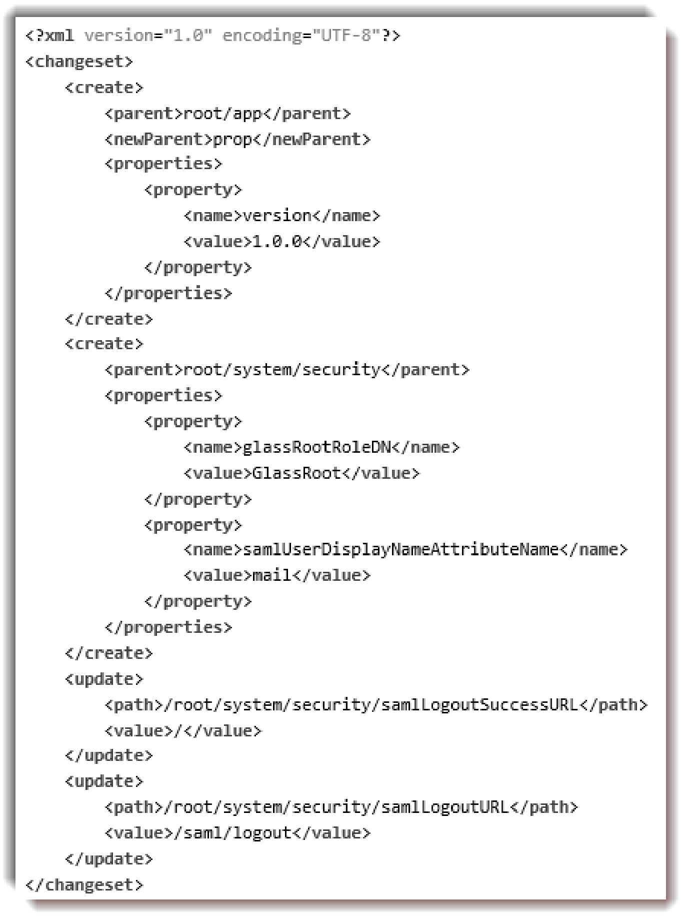
current_version_template_*{full|tenants}*.xml

- This is a template that will be used for fresh installations. Browse the file, review and modify at least all the properties that contain ##MODIFY_ME##
- The values for these properties are platform-specific and should therefore be updated.
- The other properties can also be adjusted as needed.

.xsl files required for import and export

migration_changeset.xml

- Only exists if migration is required.
- Used by **migrate.sh.**
- In some cases there may be more than one change set required. Prefer using **migrate.sh.**



```
<?xml version="1.0" encoding="UTF-8"?>
<changeset>
    <create>
        <parent>root/app</parent>
        <newParent>prop</newParent>
        <properties>
            <property>
                <name>version</name>
                <value>1.0.0</value>
            </property>
        </properties>
    </create>
    <create>
        <parent>root/system/security</parent>
        <properties>
            <property>
                <name>glassRootRoleDN</name>
                <value>GlassRoot</value>
            </property>
            <property>
                <name>samlUserDisplayNameAttributeName</name>
                <value>mail</value>
            </property>
        </properties>
    </create>
    <update>
        <path>/root/system/security/samlLogoutSuccessURL</path>
        <value>/</value>
    </update>
    <update>
        <path>/root/system/security/samlLogoutURL</path>
        <value>/saml/logout</value>
    </update>
</changeset>
```

- **migration.sh** runs the change sets required to migrate from a previous release *(exists only if migration is required).*

Additional installation steps for the SaaS solution

RSD GLASS 3.4 installation in a nutshell

These are the installation steps to perform in addition to installing RSD GLASS® as described in the RSD GLASS® Installation Guide:

- Prerequisite
- Install ZooKeeper
- Install SolrCloud
- Clustering and load balancing
- Create default schemas
- Verify Glass properties
- Install and configure OpenAM
- Set RSD GLASS® configuration in ZooKeeper
- Set GLASS® Governance Apps configuration in ZooKeeper
- Start RSD GLASS® instances

Prerequisite

As an initial prerequisite step, the folder containing temporary files on the application server needs to be secured with access rights. In other words, only the application server should be allowed to access and write to this folder.

Install ZooKeeper

ZooKeeper must be installed. See chapter “The Registry as a configuration tool” above.

Install SolrCloud

ZooKeeper is a prerequisite for SolrCloud. SolrCloud configuration files (solrconfig.xml and schema.xml) and cloud-related parameters must be kept in ZooKeeper.

Except for testing purposes, it is recommended to install SolrCloud on several machines to support high availability and enable efficient data sharding.

Each tenant is provided with one Solr collection (acting as a logical index) that can span multiple SolrCores on different machines. See next chapter “Provisioning tenants”.

Clustering and load balancing

Hazelcast

Hazelcast must be parameterized so that the EC2 Auto Discovery mechanism can discover each member of the cluster.

EC2 is the recommended auto-discovery method. For other methods please refer to the official Hazelcast documentation.

Hazelcast can be configured in ZooKeeper as follows:

Node	Comment
/system/hazelcast/group/group name	Name of the cluster
/system/hazelcast/group/group password	Password for this group
/system/hazelcast/managementCenter enabled	true if you want to use the tool to manage Hazelcast (Management Center enables you to monitor and manage your nodes running Hazelcast)
/system/hazelcast/managementCenter url	URL of the management center (ex : http://localhost:8080/mancenter)
/system/hazelcast/network/join/aws enabled	true if you want to use Amazon EC2 diffusion
/system/hazelcast/network/join/aws accessKey	Access key provided by Amazon
/system/hazelcast/network/join/aws secretKey	Secret key provided by Amazon
/system/hazelcast/network/join/aws region	Amazon region location (see the region mapping below)
/system/hazelcast/network/join/aws hostHeader	IP address of the EC2 API can be found (ex:ec2.amazonaws.com)
/system/hazelcast/network/join/aws securityGroupName	Security group name of the Tomcat's instances you want to filter on
/system/hazelcast/network/join/aws tagKey	To add a level filter in the list of Tomcat's instances what have the key tag and corresponding value
/system/hazelcast/network/join/aws tagValue	the tag value to filter on
/system/hazelcast/network/interfaces enabled	Optional true if you need to use interfaces for Hazelcast (servers mostly have more than one network interface, so you may want to list the valid IPs)
/system/hazelcast/network/interfaces interface	Optional IP addresses of your interfaces. You can specify multiple IP addresses (for example: MLT,[192.168.12.2,10.3.10.*])

Region mapping:

Value to use	Region
ap-northeast-1	Asia Pacific (Tokyo) Region
ap-southeast-1	Asia Pacific (Singapore) Region
ap-southeast-2	Asia Pacific (Sydney) Region
eu-west-1	EU (Ireland) Region
sa-east-1	South America (Sao Paulo) Region
us-east-1	US East (Northern Virginia) Region
us-west-1	US West (Northern California) Region
us-west-2	US West (Oregon) Region

HAProxy

Except for testing purposes, it is recommended to install several instances to provide high availability.

The RSD GLASS® instances must be defined to HAProxy in the haproxy.cfg file. In the example below, two RSD GLASS® instances (glass1 and glass2) are defined:

```
global
     log /dev/log syslog info
     maxconn 4096
     user haproxy
     group haproxy
     pidfile /var/run/harpoxy.pid
defaults
     timeout client 300000
     timeout server 300000
     timeout connect 10000
     timeout http-request 15000
     log global
     mode http
     option httplog
     option dontlognull
     option redispatch
     retries 3
     balance roundrobin
frontend frt_glass
     maxconn 2000
     bind 0.0.0.0:80
     default_backend glass_srvs
backend glass_srvs
     mode http
     cookie SERVERID insert indirect nocache
     server glass1 127.0.0.1:8081 check cookie glass1
     server glass2 127.0.0.1:8082 check cookie glass2
```

Create default schemas

You should create the default schemas on one of the RDS:

- one schema for scheduled action (Quartz) using Glass DDL. This schema will be shared by all the tenants.
- 2 schemas (Governance Manager and Policy Manager default schemas).

Verify Glass properties

The glass.properties file must be located in the 'glassconfig' directory of the server directory.

For the SaaS solution, the following property must be set:

```
# ZOOKEEPER

###########

glass.deploy.mode=cloud

# ZK Server conf

config.store.zk.host=demo-glass4.rsd.com:2181

config.store.zk.solrHost=demo-glass4.rsd.com:2181

config.store.zk.retryPolicy.wait.ms=1000

config.store.zk.retryPolicy.count=3

config.store.zk.connectionTimeout.ms=10000

config.store.zk.sessionTimeout.ms=60000

# ZK Tree conf

config.store.zk.base=/rsd-cloud/rsd
```

All the ZooKeeper instances are defined in the **config.stor.zk.host** property key, in a comma-separated list of host:port pairs.

A sample of the glass.properties file can be found in Appendix1.

All the zookeeper instances for SolrCloud are defined in *config.store.zk.solrHost*, in a comma-separated list of host:port pairs. If you are using SolrCloud CHROOT mode (having Solr configurations in a dedicated node in ZooKeeper), you would have to define your ZooKeeper instances in *config.store.zk.solrHost* like this:

zkhost1:2181,zkhost2:2181,zkhost3:2181/solr

Please note the CHROOT path (/solr) must be set for each host:port pair, if needed.

Install and configure OpenAM

OpenAM must be installed and configured. SAML and OAuth have to be enabled according to the specification.

Note:

When running in cloud mode (not on the premises), the configuration is managed only in ZooKeeper. Both GM, PM and govapps configurations must be completed there. This is a one-time operation for the whole platform.

Sample data is available in the zk_template_3_4_1_full.xml bundled in the Tools directory.

OAuth2 setup

The following security properties are required for the identity provider (OpenAM) to support OAuth2.

Node	Comment
gm/security/wsAuthenticationMode	Authentication mode for WebServices To set for GM and PM Value: OAUTH
system/security/oauth2/checkTokenEndpointUrl	uri to the OpenAM validation token rest service
system/security/oauth2/userIdResponseName	field used for the "UserId" in the oauth response
system/security/oauth2/tenantIdResponseName	field used for the "TenantId" in the oauth response
system/security/oauth2/rolesResponseName	field used for the "Roles" in the oauth response
system/security/oauth2/ClientIdResponseName	field used for the "ClientId" in the oauth response
system/security/oauth2/tenantIdHttpHeaderName	field used for the "TenantId" header in the request header for AppToApp flow

SAML setup

The following security properties are required for the identity provider (OpenAM) to support SAML.

Node	Comment
gm/security/authenticationMode	Authentication mode To set for GM and PM Value: SAML
gm/security/saml/idpMetadataXmlFile	A SAML 2.0 Identity Provider IDPSSODescriptor xml file. Required to register the identity provider with GLASS
gm/security/saml/spAlias	Alias for the Service Provider
gm/security/saml/spBaseURL	base url for Service Provider urls. Will be published in samlSpMetadataFileDestination
gm/security/saml/spEnityId	Entity id for the Service Provider
gm/security/saml/spMetadataFileDestination	A SAML 2.0 Service Provider EntityDescriptor xml file. GLASS will create a file here which is to be registered with the identity provider

Node	Comment
gm/security/saml/keystoreFile	location of key store for certificates published in samlSpMetadataFileDestination
gm/security/saml/keystoreDefaultKey	default key of the key store
gm/security/saml/keystorePassword	password used to access the key store and the default key
system/security/saml/bindingsSSO	SAML SSO bindings must be one or more of post,artifact,paos. Values are separated by commas ','. The first one is used as the default
system/security/saml/logoutSuccessURL	url after successfull logout
system/security/saml/logoutURL	relative url path to execute the logout
system/security/saml/nameIDs	used to set the list of NameID format in the metadata of the service provider file could be null, default value is hardcoded.
system/security/saml/rolesAttributeName	SAML assertion which contains the roles
system/security/saml/tenantIdattributeName	SAML assertion which contains the tenant id
system/security/saml/userIdattributeName	SAML assertion which contains the user id
system/security/saml/userDisplayNameAttributeName	

OpenAM must register the metadata of the applications that carry out the authentication requests.

Conversely, each application must register in ZooKeeper the metadata of OpenAM.

Technical authentication

Technical authentication is a terminology that describes an App to App direct connection. Rather than connecting on behalf of a logged in user, the calling App authenticates to OpenAM with a client Id and a secret Id to get an OAuth2 access token and then connects to RSD GLASS specifying the tenant Id and the OAuth token.

RSD GLASS then validates this token with OpenAM and assigns a role to the authentication equal to the client Id of the App (i.e. if client Id is 'RequestWeb', then a role named 'RequestWeb' must be defined in each tenant with the proper set of authorizations).

Additionally this role must be added in the ACL of each File Plan with the proper set of authorizations. In order to define an ACL on the File plan or any node of a File Plan, you may refer to the “Governance Manager User Guide”.

Set RSD GLASS® configuration in ZooKeeper

Keys may be set through Admin Console commands (zkCli.sh), using Exhibitor for ZooKeeper, the Curator Java interface or via a script.

It is suggested that you use the “**RSDGlassRegUtil**” tool to create all the properties in ZooKeeper based on a template with properties to be customized according to your environment. See 'ZooKeeper provisioning using RSDGlassRegUtil' on page page 10.

Set GLASS® Governance Apps configuration in ZooKeeper

RSD GLASS® Governance Apps requires specific keys in the **'govapps'** directory. In the same way, a tenant may have parameters in its **'govapps/tenants'** subdirectory.

More information can be found in the “RSD GLASS® Governance Apps – Installation Guide”.

Start RSD GLASS® instances

You may now start RSD GLASS® instances, using the Amazon Machine Images (AMI).

Each AMI must contain a web server (Tomcat), RSD GLASS® Policy Manager, RSD GLASS® Governance Manager and RSD GLASS® Governance Apps.

Instances have to be configured to point to RSD GLASS® configuration in ZooKeeper.

Provisioning tenants

Tenants must be provisioned in the Identity provider and in the Registry. RSD GLASS® templates must also be available to define a configuration specific to the tenant.

Steps to execute to provision a new tenant:

- Create an 'initial data' set for the tenant
- Add a new RDS (optional)
- Create SQL schemas in the RDS
- Create a dedicated Solr configuration and collection
- Provision the tenant into ZooKeeper
- Provision the tenant into OpenAM
- Add a repository

Create an 'initial data' set for the tenant

It is assumed here that RSD GLASS® has been installed in mono-tenant mode, and that RSD GLASS® Policy Manager, Governance Manager and RSD Admin Center are available.

The 'initial data' stands for the RSD GLASS® information and security data for all tenants.

1. Prepare one or more role permission schema

Provision the tenant in the Registry, using the required keys as follows.

2. Prepare RSD GLASS® templates

You have to prepare "Master Classification" and "Record Class" templates. With RSD GLASS® Policy Manager, you can prepare the required Master Classification and Record Classes that will be used as a model for all tenants. More information can be found in the "RSD GLASS® Policy Manager User Guide".

When creating a root Record class, users should not be used in security ACL rules. Only roles should be used in ACL rules.

Be sure the SEC_TARGET_USER, SEC_USER, SEC_USER_ROLE tables are empty. Then export your Policy Manager schema.

3. Import the templates

Switch to multitenant configuration.

The templates that have been prepared are used to set a new "initial data" for the tenant:

- Create your new schema.
- Set the data of your new schema with the export you just made.

Add a new RDS (optional)

Since you may use an already existing RDS for the new tenant, this step is optional and may be required only from a space management viewpoint.

All RDS are under the '/system/datasources' ZooKeeper's node.

The required keys are as follows:

Node	Comment
.../driver	The driver the application will use to connect your data source (ex: com.mysql.jdbc.Driver).
.../flexypool/maxOverFlowPoolSize	Number of connections that flexy pool can create over the maxPoolSize initially set.
.../flexypool/retryAttempt	Number of retry attempts before giving up.
.../flexypool/timeOutInMillis	Time in milliseconds to wait before a timeout. If the connection acquiring time takes more than this value, a pool size increment is attempted.
.../login	The user name to connect.
.../password	The password to connect.
.../properties/minIdle	The minimum number of connections that can remain idle in the pool, without extra ones being created, or zero to create none.
.../properties/maxIdle	The maximum number of connections that can remain idle in the pool, without extra ones being released, or negative for no limit.
.../properties/poolInitSize	Number of connections when the connection pool is started.
.../properties/poolMaxSize	Maximum of connections for the connection pool.
../properties/validationQuery	The query you want to use to check if the datasource is up (ex : SELECT 1;).
.../url	The URL where the app can find the datasource (jdbc:mysql://localhost:3306).

Create SQL schemas in the RDS

Dedicated RSD GLASS® schemas must be created into the selected RDS:

- A new schema for RSD GLASS® Policy Manager, by importing the SQL template schema.
- A new schema for RSD GLASS® Governance Manager.
- A new schema could also be needed for Roles and Permissions depending on the topology used for Role and Permission settings.

Create a dedicated Solr configuration and collection

Prerequisites

The installation of SolrCloud (version 4.10.2) relies upon a Solr-Jetty package made by RSD.

When your data is too large for one node, you can break it up and store it in sections by creating one or more shards. Each shard is a portion of the logical index, or core; it is the set of all nodes containing that section of the index.

A shard is a way of splitting a core over a number of "servers", or nodes. For example, you might have a shard for data that represents each state, or different categories that are likely to be searched independently, but are often combined. Additionally, each shard can also have multiple replicas for additional robustness.

The Solrjetty package has a default Solr configuration: */jetty/solr/default/conf.*

In this folder you will find all the files needed to configure the behavior of Solr.

Each file contained in this folder can be configured to your needs except for *schema.xml*, which must be kept as it is, because it defines the base indexation fields needed for RSD GLASS®.

Another important file, the *solrconfig.xml* file found in this config, can be modified according to your needs except for one specific part, the schemaFactory XML element:

```
<schemaFactory class="ManagedIndexSchemaFactory">

  <bool name="mutable">true</bool>

  <str name="managedSchemaResourceName">managed-schema</str>

</schemaFactory>
```

This provides a managed-schema file so that RSD GLASS® can fully operate with its dynamic metadata.

Therefore the default schema.xml, will be automatically renamed to schema.xml.bak, and a copy of this file will be created and named managed-schema.

Each new field will be added in the managed-schema file by RSD GLASS®.

Upload Solr configuration and create collection

To upload a new Solr configuration and to create a collection linked to it, use the tools provided by Solr and by SolrCloud distribution.

The example below relies upon the fact that you are using ZooKeeper chroot mode (https://wiki.apache.org/solr/SolrCloud#Zookeeper_chroot). ZooKeeper's Chroot mode allows you to define a dedicated node to upload an application configuration.

In the example, the whole configuration has been set in ZooKeeper as /solr node.

This means that when you start your first Solr instance, you must specify /solr to the ZooKeeper hosts lists (-DzkHosts parameter):

<u>First shard startup example</u>

```
-server -Djetty.port=8085 -Dcollection.configName=c1 -DzkHost=zk-master:2181/solr
-DnumShards=2 -DreplicationFactor=1 -Dbootstrap_confdir=./solr/default/conf -
DSTOP.PORT=9090 -DSTOP.KEY=stopkey -Xmx1024m -Xms512m -jar start.jar
```

<u>Second and additional shard startup example</u>

```
-server -Djetty.port=8086 -DzkHost=zk-master:2181/solr -DSTOP.PORT=9091 -
DSTOP.KEY=stopkey -Xmx1024m -Xms512m -jar start.jar
```

Note that if you do not use the ZooKeeper chroot mode, all the Solr files that describe the Solrcloud cluster will be put at the ZooKeeper root level (/).

In order to upload a new configuration for a tenant, proceed as follows:

1. Go into the cloud-scripts directory that is located in the Solrjetty distribution (ie: */opt/rsd/solr1/scripts/cloud-scripts*)
2. Execute the following command:

   ```
   ./zkcli.sh -zkhost master-zk:2181/solr -cmd upconfig -confdir
   /opt/solr/default/conf -confname c1
   ```

   After having created this new configuration, you can create a new collection that will be linked to this newly created configuration. You can use SolrCloud's REST API by invoking the CREATE action.

   The Collections API can be used to create a collection with a specific number of shards and replicas. You just have to invoke this URL with your parameters:

   http://<**SOLR_URL**>:<**SOLR_PORT**>/solr/admin/collections?action=CREATE&name=<**COLLECTION_NAME**>&collection.configName=<**SOLR_CONFIG_NAME**>&numShards=<**NB_SHARDS**>&replicationFactor=<**NB_REPLICAS**>

Parameter	Comment
SOLR_URL	The URL (hostname) for Solr.
SOLR_PORT	The port for Solr.
COLLECTION_NAME	The name of the collection to be created.

Parameter	Comment
SOLR_CONFIG_NAME	Defines the name of the configuration (which must already be stored in ZooKeeper, in the configs node) to use for this collection. When not set, will take same name as the given <COLLECTION_NAME>.
NB_SHARDS	The number of shards to be created as part of the collection.
NB_REPLICAS	The number of replicas to be created for each shard.

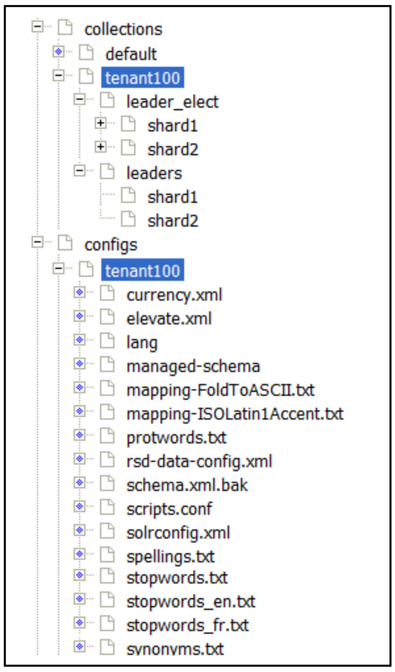
The output content will include the status of the request and the new core names. If the status is anything other than "success", an error message will explain why the request failed.

<u>Collection creation curl example</u>

```
curl
"http://localhost:8085/solr/admin/collections?
action=CREATE&name=c1&numShards=2&replicationF
actor=1&maxShardsPerNode=1&collection.configNa
me=c1"
```

In the example opposite, a collection is defined in ZooKeeper ('configs' node) for a 'tenant100' tenant.

The Solr configuration for this tenant is defined in this new node.



```
collections
  default
  tenant100
    leader_elect
      shard1
      shard2
    leaders
      shard1
      shard2
configs
  tenant100
    currency.xml
    elevate.xml
    lang
    managed-schema
    mapping-FoldToASCII.txt
    mapping-ISOLatin1Accent.txt
    protwords.txt
    rsd-data-config.xml
    schema.xml.bak
    scripts.conf
    solrconfig.xml
    spellings.txt
    stopwords.txt
    stopwords_en.txt
    stopwords_fr.txt
    synonyms.txt
```

Provision the tenant into ZooKeeper

A new node whose name is your tenant’s name must be created in each application that the tenant will use: ‘gm/tenants’, ‘pm/tenants’ and ‘govapps/tenants’. You can use the RSDGlassRegUtil dedicated tool to execute this step, see ZooKeeper provisioning using RSDGlassRegUtil page 10.

The value set in the 'datasource' node must match a name already defined in the '/system/datasources' node (see the “Add a new RDS” paragraph).

The value set in the 'schema' node must match an already defined RSD GLASS® schema (see the “Create RSD GLASS® schemas” paragraph).

The required keys are as follows:

Node	Comment
.../gm/datasource	Value for the datasource node for the RDS.
.../gm/schema	Name of the RSD GLASS® schema.
.../pm/datasource	Value for the datasource node for the Policy Manager RDS.
.../pm/schema	Name of the RSD GLASS® Policy Manager schema.
.../pmReadOnly/datasource	Value for the datasource node for the Policy Manager RDS.
.../pmReadOnly/schema	Name of the RSD GLASS® Policy Manager schema.
.../repositories	Used to set the repositories used by this tenant (see paragraph “Add a repository”).

Node	Comment
.../solr/collection	Name of the Solr's collection you created (see the “Create a dedicated SolrCloud collection” paragraph).
.../govapps	Sets the govapps that is needed for the tenant (see the related documentation).
…/roleManagement/datasource	Value for the datasource node for the Role Management RDS.

Provision the tenant into OpenAM

A tenant-ID and associated roles must be defined in the identity provider for the users belonging to the tenants. The calling application will get those ID and roles (along with the SAML assertions or the OAuth bearer token) for authentication. RSD GLASS will validate the authorization and use the application client ID as the logged-in user and as the associated roles.

Add a repository

A repository must be assigned to the tenant in ZooKeeper (not using the RSD GLASS Governance Manager interface). It has to be defined in the node named ‘gm/tenants/<tenant_name>/**repositories’**.

The repository may be physical or virtual (associated to a physical repository).

To add a new physical repository, you create a node in the 'repositories/**config**' subnode.

The required keys are as follows:

Node	Comment
.../comments	Set a comment to explain the use of the repository.
.../configXml	Set the repository configuration in XML format (see also the repository's documentation).
.../connectionKeepAliveSec	Set the 'time to live' for your connection (see also the repository's documentation).
.../driverClassName	Set the Java class name you need to use.
.../isDeactivated	Set 'true' to deactivate the repository, else set 'false'.
.../maxNbConnections	Maximum simultaneous connections to the repository.
.../minNbConnections	Minimum number of connections to the repository.
.../repoId	Repository ID, same name as the repository.

To add a new virtual repository, you create a node in the 'repositories/**virtual**' subnode.

The required keys are as follows:

Node	Comment
.../comments	Set a comment to explain the use of the repository.
.../configXml	Set the repository configuration in XML format (see the repository's documentation).
.../refRepoId	Repository ID, same name as the physical repository.
.../storageLevel	Default storage level for imported records (see the repository's documentation).
.../virtualId	Repository ID, same name as the repository.

Updating tenants

A tenant can be updated after provisioning. Here is how to proceed.

Requisites

It is assumed that:

- The RSD GLASS® Policy Manager (PM) "initial data" templates have been used to create the PM database schema for a given tenant.
- RSD GLASS® has been installed in mono-tenant mode, and RSD GLASS® Policy Manager and RSD Admin Center are available.
- This mono-tenant mode RSD GLASS® Policy Manager is configured to use the tenant's PM database and schema.

Modify role permissions

One or more role permissions schemas need to be referenced in the RSD GLASS Policy Manager and Governance Manager tenant configuration. A tenant's role permission schema may be modified after the tenant is configured.

To create a role permission schema copy an exisiting one and then modify as desired using RSD Admin Center. To do this you need to login to RSD Admin Center using a role that is a 'Super Administrator' that has the right to administrate Policy Manager role permissions.

The list of ‘Super Administrator’ roles are defined in ZooKeeper at */rsd/pm/security/superAdminRoles.*