Functional Specification
Sun GlassFish Communication Server

Author(s): binod.pg@sun.com, sreeram.duvur@sun.com

Version: 1.3

Revision History

Version	Description	Date	Author
1.0	First version	10th Oct 2007	Sreeram, Binod
1.1	Incorporated comments from Kristoffer	25th Oct 2007	Binod
1.2	Preparation for LSARC Inception Review	12th July 2008	Sreeram
1.3	Added feedback at Fast Track	9th Sep 2008	Sreeram

1 Introduction

Sun GlassFish Communication Server (SGCS) adds SIP and Telco related capabilities on top of Sun GlassFish Enterprise Server (SGES). SGES is developed as project GlassFish (http://glassfish.dev.java.net), and SGCS is developed as project SailFin (http://sailfin.dev.java.net). SailFin is an affiliate of GlassFish community in java.net.

SGCS is a full-featured Telco Application Server with load balancing, clustering and failover and administration features.

1.1 Terminology

DAS
Domain Administration Server of the Application Server

IMS
The IP Multimedia Subsystem (IMS) is an architectural framework for delivering internet protocol (IP) multimedia to mobile users.
http://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

CSCF
The CSCF provides session control for subscribers accessing services within the IP Multimedia Subsystem
http://en.wikipedia.org/wiki/IP_Multimedia_Subsystem

SNMP
SNMP is used by network management systems to monitor network-attached devices for conditions that warrant administrative attention.
http://tools.ietf.org/html/rfc1157

GMS (Project Shoal)
Project Shoal is a Group Management System heavily used in SJSAS but it can also be used in other contexts.

1.2 Features supported by 1.0 release of SGCS

High level features supported by SGCS are enumerated below. For details of each functionality, please refer to individual functional specifications.

1.2.1 Java EE 5

SGCS is developed as layer on top of SJSAS. It will support all the features available in SJSAS including Java EE 5 compatibility.

1.2.2 JSR 289 and JSR 116

SGCS will contain a sip stack that is compliant with RFCs 3261, 3262, 3265, 3311, 3515, 3903 etc. It will also expose the SIP Servlets Java APIs defined by JSR 116 and JSR 289. The JSR 289 container will be integrated to SGCS as a listener to the tomcat container.

1.2.2.1 Converged Sip and Http Sessions

SGCS will support converged HTTP and SIP session functionality as defined by JSR 289 and JSR 116.

1.2.2.2 Application Router

SGCS will support deployment of Application router archives (jar files) using the extension module mechanism. An application router that follow a simple alphabetical rule for routing the requests will be provided by default. JSR 289 specification describes semantics of default application router. That will be provided as well.

1.2.2.3 Deployment of Sip and Converged applications

SGCS will support deployment of pure Sip applications and converged applications using the extension-module mechanism in GlassFish. It will also support sun-sip.xml as the sun specific deployment descriptor for Sip applications. Please see Appendix-1 for the details.

1.2.2.4 Security Enhancements

Digest Authentication will be supported for both HTTP and SIP protocols. P-Asserted-Identity will be supported for SIP protocol. The Sip container will be enhanced to support SIPS over TLS.

1.2.3 Converged Http and Sip Load Balancer

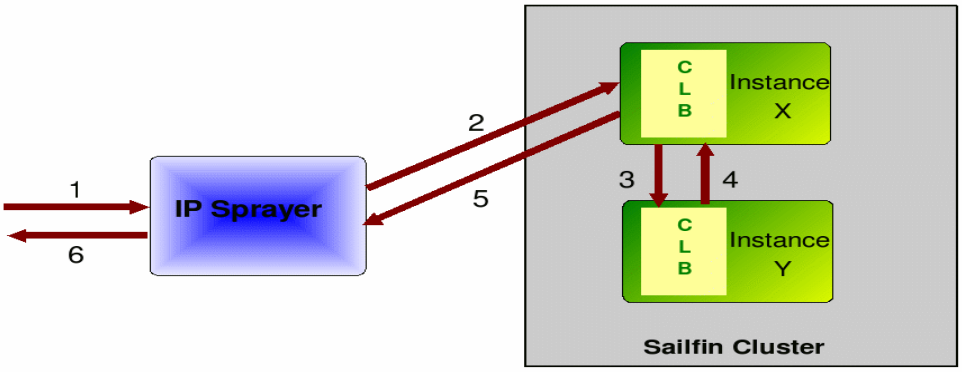
SGCS will provide a built-in load balancer that supports stand-alone Http and SIP Applications as well as converged Sip and Http applications. The load balancer will not be a separate java process nor a plug-in to a Web Server. It will be implemented as part of SGCS instance itself. This is a novel approach and a significant new direction compared to GlassFish. Over the long term this architecture may become useful in GlassFish as well, but that is not an immediate goal for this project. The new architecture permits features such as failover and failback which permit the quality of service that is demanded in telecom service deployments.

Converged Load-Balancer(CLB), is designed to provide high-availability to applications deployed on Sun Communication Application Server, aka sailfin. It will distribute requests across the instances in cluster to increase throughput of the system. It will also fail-over request from unhealthy and inaccessible instances to healthy and available instances.

Features of Converged Load-Balancer are:

1. It can handle both SIP(S) and HTTP(S) requests.
2. It can load-balance converged (application having web and sip components), pure-web and pure-sip applications.
3. It will distribute the load across healthy and available instances in cluster, thus enabling a higher throughput from the system
4. It maintains stickiness of requests in a session to single instance till serving instance is healthy, available and enabled
5. It will fail-over request from unhealthy/unavailable/disabled instances to instances which are healthy, available and enabled
6. It supports round-robin and consistent-hash algorithm for load-balancing
7. It supports Data-Centric-Rule(DCR) to extract hash-key used in consistent-hash algorithm

Illustration below depicts the functioning of converged load-balancer:



- Step 1 : Client request comes to IP Sprayer
- Step 2 : IP sprayer selects any of the sailfin instances in the cluster and forwards request to that instance, which in above illustration is instance X.
- Step 3 : CLB on Instance X takes following action

- If it is http request, then it does a match of context-root
  - if match is found, then router for that context-root it used to select an instance, to service request, based on configured algorithm. In above illustration it will be instance Y.
  - If match is not found, request is processed by local instance itself. In above illustration, it will be instance X
- If it is a sip request
  - If dcr file is specified then hash key is extracted based on dcr rules. Otherwise hash key is extracted using default headers
  - Hash key is used to select an instance to service request. In above illustration it will be instance Y.
- The request is proxied to instance Y if it selected as instance to service request

- Step 4 : CLB on Instance Y receives the request. It finds out that request is already proxied from another instance. Without any further processing, it passes the request to the container so that it can be serviced. CLB then sends the response back to CLB on Instance X.
- Step 5 : CLB on Instance X in turn sends the response back to IP sprayer
- Step 6 : IP sprayer sends out the response back to the client

Note: In above illustration instance X and instance Y can be same instance as well. In such a case, request in not proxied. It is just passed up the stack.

Supported Algorithms

Converged load-balancer currently supports two load-balancing algorithm

1. Round-robin : Instance to service new request are selected in a round robin fashion.
2. Consistent-hash : Instance to service new request is selected based on hash-key extracted from request.

In both cases, once a session is established, sticky information is set on response. Subsequent requests will have that sticky information. Thus subsequent requests part of same session will stick to same instance.

There are two suggested configuration of Converged load-balancer :

- Configuration 1: This should be used when only pure web applications are deployed. The user does not provide a dcr-file.
  - Round-robin algorithm for all http requests
  - Consistent-hash algorithm for all sip requests, in case converged/pure-sip applications are deployed. The hash-key

used to select instance is extracted using from-tag,to-tag,call-id parameters of the request.

- Configuration 2: This should be used when converged/pure-sip applications are deployed on the application server. The user must provide a dcr-file in this case, to extract hash key from sip and http requests. If dcr-file is not provided, sip and http requests part of same session may be serviced by different instances.
    - Round-robin algorithm for http requests belonging to pure web applications
    - Consistent-hash algorithm for http requests belonging to converged applications
        - If any dcr rule matches http request, hash-key is extracted using that rule
        - If none of the rules matches http request, hash key is extracted using remote host and port of the http request
    - Consistent-hash algorithm for sip request belonging to converged/pure-sip applications
        - If any dcr rule matches sip request, hash-key is extracted using that rule
        - If none of the rules matches sip request, hash key is extracted using from-tag,to-tag,call-id parameters of the sip request

1.2.4 Session Replication for Sip and Http Sessions

Session replication capability in SGCS will support both SIP and HTTP sessions. It will also support replication of Sip Timers, SIP Dialog state and SipApplicationSession objects as defined by JSR 289.

The current session replication framework in SJS AS will be extended to support usecases demanded by the Sip application state. There are no new external interfaces other than those already defined by GlassFish.

HADB will not be supported in this release.

1.2.5 Administration Support for SGCS

The configuration of SGCS is based on additional configurational elements added to domain.xm related to sip protocol listeners. The additions are all optional which keeps it compatible with earlier product releases.

asadmin CLI commands and GUI will be developed to support the functionality introduced in SGCS. This will cover both Sip container and Converged load balancer.

The current monitoring capabilities of the SJSAS will be extended to support SGCS specific monitoring and AMX. The administration support will be an extension on top of the current glassfish administration framework.

1.2.6 Netbeans Tooling

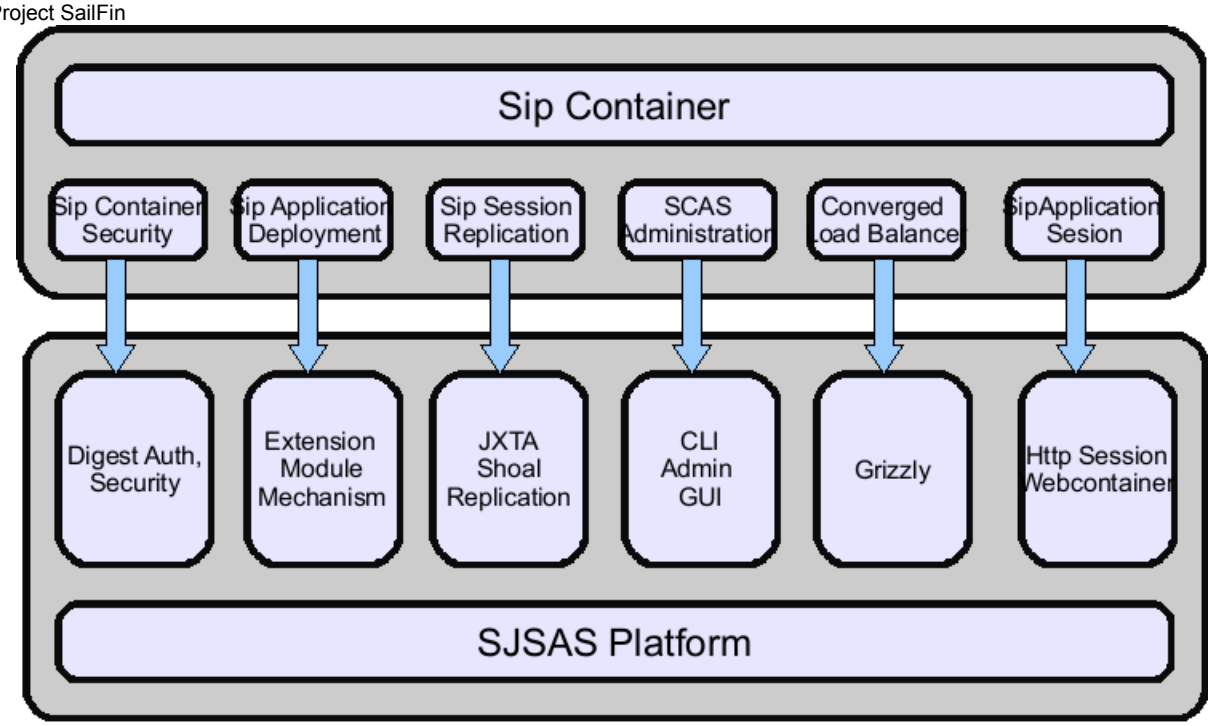
Support for Sip Servlet applications based on JSR 289 will be provided on top of Netbeans 6.1. A simple Sip client will be added to enable development time testing. This plug-in will be provided via the netbeans update center.

2 Design Overview

2.1 Relationship with SJSAS

Since SGCS is meant to add functionality on top of SJSAS, the product is implemented as a layer on top of SJSAS. Necessary extensions are implemented in base SJSAS so that the new functionality can be plugged in. All of the SJSAS functionality will be available in SGCS as well. It also mean that where not specified the architecture assume the SJSAS architecture and thus SGCS will inherit all the advantages and limitations of the SJSAS.

The following picture capture the current list of interfaces.



All these project private interfaces are implemented in the SJSAS 9.1.1 release and are not described further.

2.1.1 Profiles

SJSAS support developer and cluster profiles. SGCS will continue to support them. Basic Sip Servlet Container will be available by default in developer profiles. User will be able to deploy and run a pure Sip or Converged application in a single instance. Converged Load Balancer and Sip Session replication will be available only in cluster profile. SGCS will not support enterprise profile of the SJSAS as this does not meet the requirements of telecom service providers.

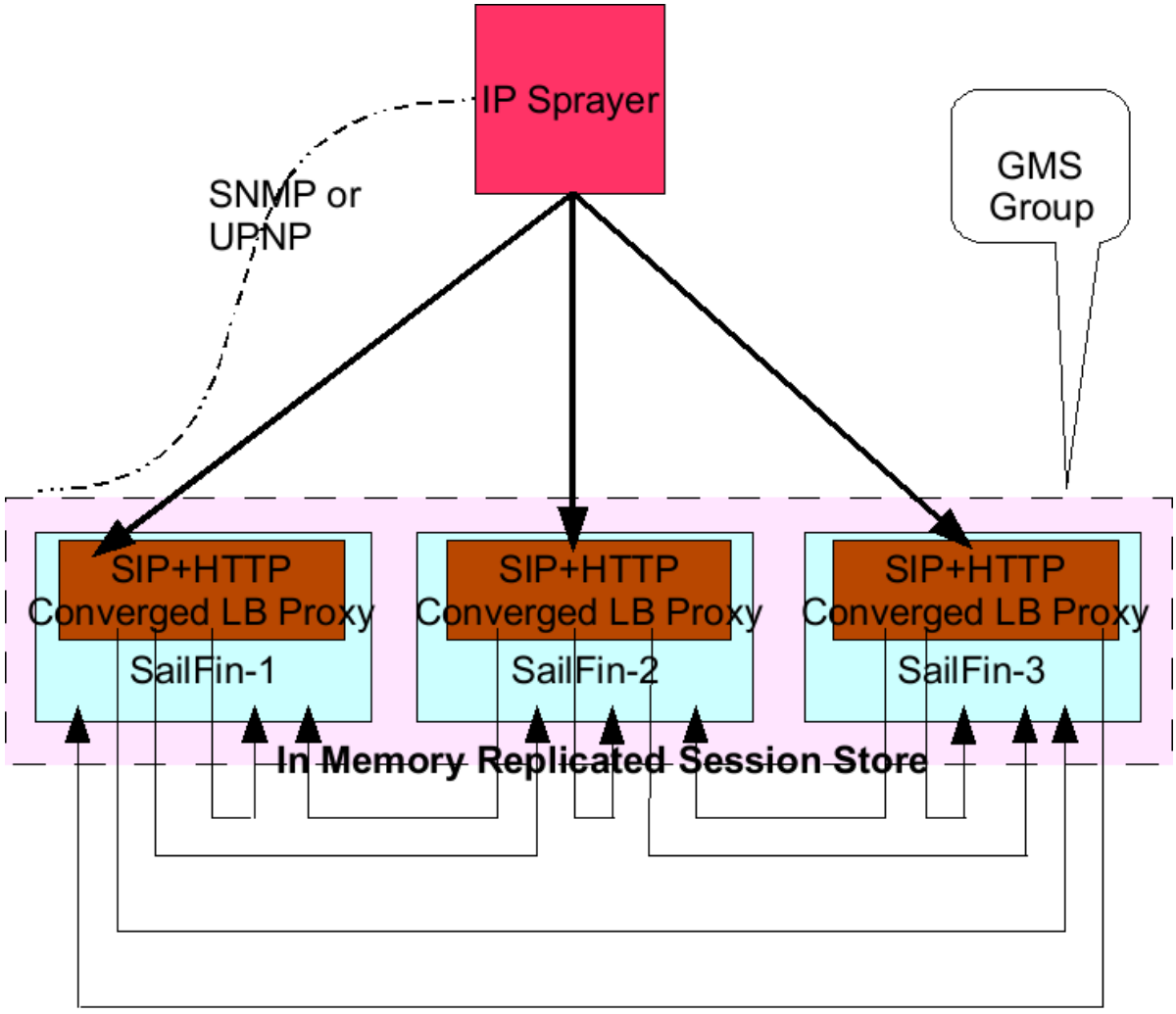
2.2 Product Deployment Topology.

SGCS can be deployed to run Java EE Applications, Sip applications or Converged Sip and Java EE applications. These applications can use any of the Java EE and other functionalities supported by JSR-289 and by developer or cluster profiles of the SJSAS.

It is expected that for most of the production deployments of the SGCS, the Converged Load Balancer will be used. However, it would be possible to use the

native load balancer plug-in also with SGCS using a manual setup process. Native load balancer can be used only for HTTP load balancing.

It will be possible to use the converged load balancer as part of the cluster instances itself. Each instance of the cluster will act as both load balancing proxy and the back-end server instance.



It is very likely that the load balancer tier is fronted by a hardware IP level load balancer (referred to as a IP sprayer). The IP sprayer abstraction enables a single IP view of SGCS cluster to external clients. Examples of the IP Sprayer could be an IP Load Balancer appliance, IP virtualization capabilities such as those found in Sun Cluster or TIPC/Linux.

The IP sprayer will be able to view the shape of the load balancer cluster using protocols like SNMP or its own custom policies. This is beyond the scope of this project.

It is expected that all the instances of the SGCS cluster are in the same subnet. A cluster spanning multiple subnets will not be supported in this release.

2.3 Security

Digest authentication is implemented in both Http and Sip containers. The current JDBC realm will be enhanced to support Digest authentication. In most of the IMS deployment, if the application server is fronted by a CSCF, the initial authentication will be done by CSCF. In that case, a special p-asserted-identity header will be added to the Sip Message to indicate that it is from a trusted source. SGCS will support p-asserted-identity.

There will be a way to configure a trusted intermediate information to administratively determine an incoming or outgoing sip traffic is from a trusted entity or not. See section 5.2.3 for details.

We also added TLS support in SIP socket listeners.

Sip Servlet authentication will be integrated to SJSAS security framework so that the security context from the Sip Servlet will be propagated to EJBs.

3 Availability

The current session persistence framework that is based on a ring topology will be used for replication of Sip timers, dialogs, dialog fragments, sessions , Converged Http Sessions and Sip Application Sessions. The state of the application at any point is represented by this inter-related complex of objects. Since these objects are replicated to another instance, the memory requirement for session storage will increase by 100%.

The availability story will be based on the converged load balancer, session replication and shoal cluster membership service. Shoal framework will be enhanced to support a NIO based discovery mechanism as against the current UDP based communication.

SGCS will support only asynchronous mode of the replication. It means that any replication activity will be asynchronously replicated to the its replica buddy and no acknowledgment mechanism for the replication will be implemented. There is due to a known performance limitation of synchronous bidirectional pipes, as currently implemented in JXTA. We will transparently switch to a reliable synchronized mode in a future release.

3.1 Rolling Upgrade

Upgrading an application in SGCS will follow similar procedure as SJSAS. Only self-compatible versions of the applications will be supported, which means the new application should use the same same database schemas and session information and should have a compatible business logic. The old sessions are serialized out, the new application version is loaded and the saved sessions are de-serialized. The converged load balancer will block traffic to the application server instance being upgraded and resume it afterwards. Sessions may migrate to other instances and get modified during the upgrade which may require that some restored sessions require an update. The process iterates through all the instances in the cluster. Following are the steps for upgrading the application.

1. Backup the domain.xml using the command line interfaces.
2. Switch off dynamic reconfiguration for the cluster.
3. Deploy the new version of the application. Since the dynamic reconfiguration is switched off, the new bits wont be synchronized to the cluster instances and they will just remain in the domain administration server.
4. Disable one server instance from the load balancer using the asadmin command so that load balancer quiesce the requests to the instance. During this period, load balancer will make sure that all the new requests are routed to a different instance.
5. After the quiescing period is over, restart the instance. During restart, the instance will get synchronized with DAS and thus the application bits will be upgraded. When the instance is restarted, it's sessions will be migrated back from the other instance. Also, the load balancer will start routing all the requests to the failed over instance again as per the consistent hashing algorithm.
6. Repeat the above steps for all the instances.

It will also be possible to do a minor compatible upgrade of the application server using similar steps.

4 Performance

SGCS will be tested for a minimum of 10 instances in a cluster. Exact performance goals of SGCS is documented at the “Performance Requirement Documentation”

5 Management and Monitoring

SJSAS administration/management framework will be extended to support SGCS. The extensions will be for the following.

- Asadmin CLI framework to honor an additional CLI descriptor.
- GUI to insert new screens SGCS.

- Mbean framework to pickup additional descriptor files.
- Dynamic reconfiguration framework to honour an extended domain.xml file.
- AMX.

There will be enhancements to domain.xml for the following.

- New sip-service element, similar to http-service.
- New sip-container element to configure the container
- New converged-load-balancer related elements.
- Sip-container availability related elements.
- Security Related Elements for Identity Assertion.

There will be new asadmin commands and GUI screens to manage these. More details are available in the Administration Functional Specification.

The current monitoring framework of SJSAS will be extended to expose Sip Servlet monitoring capabilities. The exact list of monitoring attributes are listed in the Administration functional specification. The sip container will also be enhanced to support callflow.

5.1 DTD changes for SIP container

The following new elements have been introduced:

λ **sip-service**: 0 or 1 occurrence of this element to represent a SIP service configuration containing various settings for access-log, SIP listeners, request processing, keep alive, connection pool, SIP protocol and additional properties.

```
<!ELEMENT sip-service
  (access-log?, sip-listener+, request-processing?, keep-alive?,
  connection-pool?, sip-protocol?, property*)>
```

λ **sip-listener**: 1 or more occurrences of this element to represent SIP listener(s). Default SIP listener listens on port 5060. For secure mode (transport=tls), port is 5061.

```
<!ELEMENT sip-listener (ssl?, property*)>
```

```
<!ATTLIST sip-listener
  id CDATA #REQUIRED
  address CDATA #REQUIRED
  port CDATA #REQUIRED
  transport (udp_tcp | tls) "udp_tcp"
  enabled %boolean; "true">
```

Attributes for sip-listener:

address

IP address of the listen socket. Can be in dotted-pair or IPv6 notation. Can also be any for INADDR-ANY. Configuring a listen socket to listen on any is required if more than one sip-listener is configured to it.

id
Unique identifier for sip listener.

port
Port number to create the listen socket on. Legal values are 1 - 65535. On Unix, creating sockets that listen on ports 1 - 1024 requires superuser privileges. Default SIP listener port is 5060. When transport=tls, it is 5061.

transport
Specifies the type of transport layer protocol. Default is “tcp_udp”

- **sip-container-availability**: 0 or 1 occurrence representing SIP session persistence settings and additional properties.

```
<!ENTITY %sip-session-save-frequency "(sip-transaction)">

<!ELEMENT sip-container-availability (property*)>

<!ATTLIST sip-container-availability
  availability-enabled %boolean; #IMPLIED
  persistence-type CDATA "memory"
  persistence-frequency %sip-session-save-frequency; #IMPLIED
  persistence-scope %session-save-scope; #IMPLIED
  repair-during-failure %boolean; "true">
```

Attributes for sip-container-availability:

availability-enabled
This boolean flag controls whether availability is enabled for SIP session persistence.

persistence-type
Specifies the sip session persistence mechanism for sip applications that have availability enabled. Default is "memory".

repair-during-failure
Specifies whether a forward and reverse repair should be performed on an instance that has (re)joined the cluster.

persistence-frequency
The persistence frequency used by the session persistence

framework.

persistence-scope
The persistence scope used by the session persistence framework.

- **sip-protocol**: 0 or 1 occurrence containing settings for SIP link, SIP timers and additional properties.

```
<!ELEMENT sip-protocol (sip-link?, sip-timers?, property*)>

<!ATTLIST sip-protocol
  error-response-enabled %boolean; "false"
  default-tcp-transport %boolean; "false">
```

Attributes for sip-protocol:

default-tcp-transport
Boolean flag to denote if transport=tcp should be inserted in URI of contact and record-route header. Default is "false".

error-response-enabled
Boolean flag to denote if error response should be sent. If "true", respond 400 on bad request or drop. Default is "false".

- **sip-link**: 0 or 1 occurrence representing SIP connection settings.

```
<!ELEMENT sip-link EMPTY>

<!ATTLIST sip-link
  connection-alive-timeout-in-seconds CDATA "120"
  max-queue-length CDATA "50"
  write-timeout-in-millis CDATA "10"
  write-timeout-retries CDATA "25">
```

Attributes for sip-link:

connection-alive-timeout-in-seconds
Defines the time, in seconds, a SIP link is allowed to be inactive before the connection is closed. Default is 120 seconds.

max-queue-length
Defines the maximum number of simultaneous write requests or

connect requests, or both, that can be waiting to write on a link. Default is 50 requests.

write-timeout-in-millis
Defines timeout value in milliseconds for a single write operation of a SIP link. Range 1-50ms. Default is 10ms.

write-timeout-retries
Defines the number of retries to perform a single write operation of a SIP link. Range 1-25. Default is 25.

- **sip-timers**: 0 or 1 occurrence representing SIP timers settings. Detailed descriptions for the timers can be found in RFC 3261 Chapter 17

```
<!ELEMENT sip_timers EMPTY>

<!ATTLIST sip_timers
  t1_in_millis CDATA "500"

  t2_in_millis CDATA "4000"
  t4_in_millis CDATA "5000">
```

Attributes for sip-timers:

t1-in-millis
SIP timer T1 (round trip time estimate) in milliseconds. For unreliable transports (such as UDP), the client transaction retransmits requests at an interval that starts at T1 seconds and doubles after every retransmission. T1 is an estimate of the round-trip time (RTT), and it defaults to 500 ms. Nearly all of the transaction timers described here scale with T1, and changing T1 adjusts their values.

t2-in-millis
SIP timer T2 (maximum retransmit interval for non-INVITE requests and INVITE responses) in milliseconds. For unreliable transports, requests are retransmitted at an interval which starts at T1 and doubles until it hits T2. If a provisional response is received, retransmissions continue for unreliable transports, but at an interval of T2. The default value of T2 is 4000ms, and it represents the amount of time a non-INVITE server transaction will take to respond to a request, if it does not respond immediately.

t4-in-millis
SIP timer T4 represents the amount of time the network will take to clear messages between client and server

transactions. The default value of T4 is 5000ms.

1.1.1 DTD changes for converged load balancer administration

The following new elements related to converged load balancer configuration are described here:

- **converged-lb-configs**: 0 or 1 occurrence of this element which describes the configured converged load balancer configurations in a domain.

```
<!ELEMENT converged-lb-configs (converged-lb-config*)>
```

- **converged-lb-config**: 0 or more occurrences of this element which represents a converged load balancer view of the load balanced deployment. This is used by the converged load balancer to configure itself.

```
<!ELEMENT converged-lb-config

  (converged-lb-policy, (converged-lb-cluster-ref* | server-ref*),  property*)>


<!ATTLIST converged-lb-config
  name CDATA #REQUIRED>
```

Attributes for convered-lb-config:

name

The name of the converged load balancer configuration

- **converged-lb-cluster-ref**: 0 or more occurrences of this element relating to a converged cluster that needs to be load balanced.

```
<!ELEMENT converged-lb-cluster-ref EMPTY>


<!ATTLIST converged-lb-cluster-ref
  ref CDATA #REQUIRED
  self-loadbalance %boolean; "true">
```

Attributes for converged-lb-cluster-ref:

ref

A name to converged cluster defined.

self-loadbalance

Boolean attribute which specifies whether configured cluster self load balances incoming requests to itself. If it's configured to do so, load balancer is an intrinsic component of the participating server instances in the cluster. Default value is “true”.

- **converged-lb-policy**: Specifies the load balancing policy used by the converged load balancer.

```
<!ELEMENT converged-lb-policy (hash-rules, property*)>


<!ATTLIST converged-lb-policy
  http CDATA "round-robin"
  sip CDATA "from-tag,to-tag,call-id"
  dcr-file CDATA #IMPLIED>
```

Attributes for converged-lb-policy:

dcr-file
: It denotes the data centric rules file name, which provides the rules for applying consistent hashing on both HTTP and SIP requests.

http
: Specifies the load balancing policy used for the http requests. The default implied value is round-robin.

sip
: Specifies the parameters on which consistent hashing policy is applied to obtain the hashkey. This can be specified as single value or comma seperated values of parameter names to hash on. In case more than one parameter is specified, then the concatenated values of the parameters would be used for applying the consistent hashing. The default value implied will be "from-tag,to-tag,call-id".

- **converged-load-balancer**: 0 or 1 occurrence of this element describing a converged load balancer. JavaEE server upon startup can come up with a load balancer that facilitates high availability of converged application deployments - SIP and HTTP requests.

```
<!ELEMENT converged-load-balancer (proxy)>


<!ATTLIST converged-load-balancer
  name CDATA #REQUIRED
  config-file CDATA #REQUIRED
  auto-commit CDATA "false"

  converged-lb-config-name CDATA #REQUIRED>
```

Attributes for converged-load-balancer:

auto-commit
: when true, config file changes should be propagated to clb instances immediately.

name
: Converged load balancer's name.

config-file
: Converged load balancer's configuration file. This can be an

absolute or relative path. Relative path is resolved with the config directory of the server.

- **proxy**: Specifies the Proxy component of the JavaEE server. This would provide for pass through of HTTP/SIP requests to remote HTTP/SIP endpoint; that is configured via http-service/sip-service of the remote instance.

```
<!ELEMENT proxy (property*)>

<!ATTLIST proxy
  request_pool_size CDATA "50"
  send_retry_count CDATA "3"
  read_timeout_in_millis CDATA "1500">
```

Attributes:

read-timeout-in-millis
: the duration for which we would wait for data from the client in the socket channel

request-pool-size
: denotes the number of request objects that will be created and pooled by the proxy

send-retry-count
: the number of retries the proxy would attempt with the remote instance when sending of data fails.

Known Properties : Eventually some of them may become attributes.

max-parallel-connections
: maximum number of outbound connections to a backend instance.

high-water-mark
: maximum number of active outbound connections Controller will handle.

connections-to-reclaim
: number of LRU connections, which will be reclaimed in case highWaterMark limit will be reached.

proxy-server-read-timeout
: Timeout for proxy to read from the server channel once response is available.

socket-receive-buffer-size
: proxy-backend socket receive buffer size.

socket-send-buffer-size

proxy-backend socket send buffer size.

client-socket-read-timeout
client socket send read timeout.

5.1.1 DTD changes for Identity Assertion Trust Configuration

Identity assertion trust domain configuration information as per RFC 3325. P-Asserted -Identity header received from hosts and domains can be trusted. P-Asserted-Identity header has identity of a user who was authenticated at another node in the network.

Please refer to the SailFin security specification for more details.

```
<!ELEMENT identity-assertion-trust ((trusted-entity* | trust-handler))>

<!ATTLIST identity-assertion-trust
  id CDATA #REQUIRED

  is-default %boolean; "false">

<!--trusted-entity
  Trusted intermediate trusted hosts/domains as per RFC 3325.


 attributes
  id
    Unique identifier for the trusted entity.

  trusted-as
                trusted-as with value 'intermediate' represents configuration information for
                incoming messages,if it has value 'destination' then configuration under trusted-
                entity is applied to outgoing messages.


 Used in:
  identity-assertion-trust
-->

<!ELEMENT trusted-entity (ip-address, host-name?, principal?)>


<!ATTLIST trusted-entity
  id CDATA #REQUIRED
  trusted-as (intermediate | destination) #IMPLIED>


<!--trust-handler

 Used in:
  identity-assertion-trust

-->
```

```
<!ELEMENT trust-handler EMPTY>

<!ATTLIST trust-handler
  class-name CDATA #REQUIRED>

<!--ip-address
  Identifies the trusted host on the network.eg : 129.169.223.2

 Used in:
  trusted-intermediate

-->

<!ELEMENT ip-address ( #PCDATA )>

<!--domain-name
  Identifies the trusted host on the network using domain names.
  eg: sun.com, cisco.com. All hosts from sun.com domain are trusted.

 Used in:
  trusted-intermediate

-->

<!ELEMENT host-name ( #PCDATA )>
```

5.1.2 Stack Layer configuration:

Defines the configuration of a stack of layers, typically related to protocols, such as SIP. Used in sip-container

```
<!ELEMENT stack-config (stack-layer*, property*)>

<!ATTLIST stack-config

  layer-order CDATA #REQUIRED>

Attributes:
  layer-order
    A comma seperated list indicating the order of the stack
    layers. Use the the stack-layer.id for the list elements.

<!--stack-layer
  Defines a layer of a stack.


 children
  property
    Any property is a javabean property injected in the layer
    class, if a corresponding javabean setter exist in the layer
    class.


 attributes
```

```
  class-name
    Fully qualified name of the layer class.
  id
    Unique identifier for stack-layer.


 Used in:
  stack-config

→

<!ELEMENT stack-layer (property*)>

<!ATTLIST stack-layer
  id CDATA #REQUIRED
  class-name CDATA #REQUIRED>
```

5.2 New CLI Commands

5.2.1 SIP listener

The commands to create, delete and list SIP listener. The target option (create and delete commands) and operand (list command) is the name of the target being operated on.

Syntax

- create-sip-listener [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4949] [--secure | -s] [--user admin_user] [--passwordfile file_name] [--transport udp_tcp|tls] [--enabled=true] [--target target(Default server)] --siplisteneraddress address --siplistenerport listener_port sip_listener_id
- delete-sip-listener [--target target] sip_listener_id
- list-sip-listeners target

Argument	Definition	Default	Range
--siplisteneraddress	IP address or hostname (resolvable by DNS) to be used for listening to requests.		
--siplistenerport	Port number to create the listen socket on. On Unix, creating sockets that listen on ports 1 - 1024 requires super user privileges. Default SIP listener port is 5060. When transport=tls, it is 5061.	5060|5061	1-65535
--transport	Specifies the type of transport layer protocol.	udp_tcp	udp_tcp|tls
--enabled	Boolean attribute. If set to "false" this module will not be loaded at server start up.	true	
--target	Name of target being operated on. The valid targets for this command are config, instance, cluster, or server. By default the target is the 'server'.	server	

5.2.2 Converged Load Balancer

Commands to create delete and list converged load balancers and the converged load balancer configs.

- **create-converged-lb-config**
  The create-converged-lb-config command is used to create a converged load balancer configuration. The converged load balancer configuration name must be unique in the domain, and must not conflict with any Node Agent, configuration, cluster, or server instance names in the domain. This gives a flat name space so that the dotted name notation can be used to access any of these entities without ambiguity. If config_name operand is not specified and --target option is specified, then a load balancer configuration is implicitly created with an unique name. If --target option is not specified and config_name operand is specified, then a load balancer configuration is created but with no references to any target. If both --target option and config_name operand are specified, then a lb configuration is created referencing the specified target. An error is displayed if neither --target option or config_name operand is not specified on the command line.

Syntax

create-converged-lb-config [--reloadinterval 60] [-httplbpolicy policy_name] [--siplbpolicy policy_name] [--dcrfilename file_name] [--target target] [--property (name=value)[:name=value]*] [clb_config_name]

Argument	Definition
--reloadinterval	reload pool interval in seconds at which load balancer has changed. If it has change, load balancer would reload it. Value of 0 would imply that polling is disabled.
--target	The target can either be a cluster or an unclustered instance.
--httplbpolicy	policy name to be used for routing the http requests
--siplbpolicy	policy name to be used for routing the http requests
--dcrfilename	filename of an xml file where complex rules are specified for both http and sip requests.
--property	name/value pairs
config_name	The name of the new converged load balancer configuration. This name must not conflict with any other load balancer groups, agents, configurations, clusters, or server instances in the domain.

- **delete-converged-lb-config**

The delete-converged-lb-config command deletes a load balancer configuration. The load balancer must not reference any clusters or instances before it can be deleted.

Syntax

delete-converged-lb-config clb_config_name

Argument	Definition
clb_config_name	The name of the converged load balancer configuration to delete.

- **list-converged-lb-configs**
  The list-converged-lb-configs command lists the load balancer config and its clusters/instances, or all the load balancer config in the domain.

  Syntax
  list-converged-lb-configs [target]

Argument	Definition
target	target can either be a instance name or cluster. If the target is either a cluster or an instance, then the lb config that references to the the cluster or instance will be displayed. If target is not specified, then the list of all lb configs will be displayed.

- **create-converged-lb-ref**
  The create-converged-lb-ref command is used to add an existing cluster to an existing load balancer configuration.

  Syntax

  create-converged-lb-ref –clbconfig config_name [--selfloadbalance] [--lbEnableAllInstances] [--lbEnableAllApplications] target

Argument	Definition
--clbconfig	The name of the converged load balancer configuration
--selfloadbalance	If it is true, cluster load balances the incoming requests to itself.
--lbenableallinstances	Enable all the associated instances for a target cluster
--lbenableallapplications	enable all the associated applications for a target
target	The target to be added to the load balancer configuration. Target can either be a cluster or an instance name.

- **delete-converged-lb-ref**
  The delete-converged-lb-ref command is used to delete a cluster/instance reference from a load balancer config. It is important to note that if you wish not to interrupt users accessing applications in the server you should ensure that all of its instances have been disabled before removing the cluster.

  Syntax

  delete-converged-lb-ref --clbconfig config_name target

Argument	Definition
--clbconfig	The name of the converged load balancer configuration
target	The target to be deleted from the load balancer configuration. target can either be a cluster or an instance name

- **create-converged-lb**
  The create-converged-lb command is used to create a converged loadbalancer.

  Syntax

  create-converged-lb --clbconfig config_name [--configfile file_name] [--autocommit] [--property (name=value)[:name=value]*] [--target target] load_balancer_name

Argument	Definition
--clbconfig	Name of the converged-lb-config used by this converged load balancer
--configfile	File name of the converged load balancer
--autocommit	Immediately propagate the changes to clb instances
--property	Name/value pairs
--target	target config name
load_balancer_name	Name of the converged load balancer

- **delete-converged-lb**
  Command to delete the loadbalancer element.

  Syntax

  delete-converged-lb [target (default server)]

Argument	Definition
load_balancer_name	lb name

- **list-converged-lbs**
  Command to list all converged LB configs.

  Syntax

  list-converged-lbs

- **set-dcr-file**
  Command to upload the DCR file to DAS. The uploaded file will then be synced up by the instances.

  Syntax

  set-dcr-file --dcrfile dcr_file_name [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure|-s=true] [--user admin_user] [--passwordfile file_name] clb_config_name

Argument	Definition
dcrfile	Path to the local dcr file which needs to be uploaded
clb_config_name	CLB config pertaining to the DCR file

5.2.3 Identity Assertion Trust Management Commands

Command to create/delete/list trust configurations.

- create-trust-config [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure|-s=true] [--user admin_user] [--passwordfile file_name] [--isdefault=false] [--trusthandler=class_name] [--property (name=value)[:name=value]*] [--trustedas=intermediate|destination] [--ipaddress=ip_address] [--hostname=host_name] [--principal=principal_name] [--entityid=trusted_entity_id] [--target target(Default server)] [trust-id]

Note that the trust config and entity ids are automatically generated. They will be of form trustid-0, trustid-1.. entityid-0, entityid-1..

Invoking this command without any options will create a trust configuration with a default trust handler.

Argument	Definition
isdefault	Boolean flag to mark/unmark a trust config as default. Only one of the configured trust configs can be made as default
trusthandler	Specify the implementation class name that implements com.sun.enterprise.security.trust.TrustHandler
trustedas	trusted-as with value 'intermediate' represents configuration information for incoming messages,if it has value 'destination' then configuration under trusted-entity is applied to outgoing messages.
ipaddress	Identifies the trusted host on the network.eg : 129.169.223.2
hostname	Identifies the trusted host on the network using domain names. eg: sun.com, cisco.com. All hosts from sun.com domain are trusted.
principal	Principal of the client
entityid	Identifier for trusted entity

- delete-trust-config [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure|-s=true] [--user admin_user] [--passwordfile file_name] [--target target(Default server)] trust-id

  This command will delete the specified trust config from the target.
- list-trust-configs [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure|-s=true] [--user admin_user] [--passwordfile file_name] [--trustid=trust-id] [target(Default server)]

  This command can be used to list the trust configs as well as trusted entities.
- create-trusted-entity [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure|-s=true] [--user admin_user] [--passwordfile file_name] [--trustedas=intermediate|destination] [--hostname=host_name] [--principal=principal_name] [--target target(Default server)] --trustid=trust_id --ipaddress=ip_address [entity_id]

  This command will create a trusted entity for given trust config. See create-trust-config for option descriptions.
- delete-trusted-entity [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure|-s=true] [--user admin_user] [--passwordfile file_name] [--target target(Default server)] --trustid=trust_id entity_id

This command will delete the trusted entity from the trust config.

5.3 Changes to existing CLI commands

λ create-ssl

Existing usage text:

Usage: create-ssl --type [http-listener|iiop-listener|iiop-service] --certname cert_name [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure | -s] [--user admin_user] [--passwordfile file_name] [--ssl2enabled=false] [--ssl2ciphers ssl2ciphers] [--ssl3enabled=true] [--ssl3tlsciphers ssl3tlsciphers] [--tlsenabled=true] [--tlsrollbackenabled=true] [--clientauthenabled=false] [--target target(Default server)] [listener_id]

create-ssl should accept an additional type 'sip-listener' to configure the SSL element in the selected sip-listener.

λ delete-ssl

Existing usage text:

Usage: delete-ssl --type [http-listener|iiop-listener|iiop-service] [--terse=false] [--echo=false] [--interactive=true] [--host localhost] [--port 4848|4849] [--secure | -s] [--user admin_user] [--passwordfile file_name] [--target target(Default server)] [listener_id]

delete-ssl should accept an additional type 'sip-listener' to delete the SSL element in the selected sip-listener.

λ create-domain

For any profile (developer, cluster) create-domain should create a domain with SIP configuration. This can be handled by using GlassFish profile management feature. See section 2.2

5.4 Formal Interfaces

See One Pager.

5.4.1 Reference Documents

- Administration Functional Specification
  - http://wiki.glassfish.java.net/attach/FunctionalSpecsOnePagers/sailfin_admin.doc
- Load balancer Functional Specification
  - http://wiki.glassfish.java.net/attach/FunctionalSpecsOnePagers/ConvergedLB-FSD.pdf
- Converged LB Proxy Functional Specification.
  - http://wiki.glassfish.java.net/attach/FunctionalSpecsOnePagers/lb_proxy_fsd.doc
- Session Replication Functional Specification.
  - http://wiki.glassfish.java.net/attach/FunctionalSpecsOnePagers/FSD-SSR-2.3.odt
- Deployment Functional Specification.

- http://wiki.glassfish.java.net/attach/FunctionalSpecsOnePagers/sailfin_fsd_deployment.doc
- Security Functional Specification.
  - http://wiki.glassfish.java.net/attach/FunctionalSpecsOnePagers/sailfin_fsd_security.odt
- Converged Http Session Functional Specification.
  - http://wiki.glassfish.java.net/attach/FunctionalSpecsOnePagers/ConvergedHttpSession_FunctionalSpec.doc

6 Packaging, Files, and Location

SGCS will be delivered as a file based installer that will install SJSAS and then the additional SGCS functionality on top of it. The file based installer will contain the J2SE 6.0 version that will be used for testing SGCS.

SGCS installer will add SGCS jar files to the lib directory of the SJSAS so that they will be picked up by SJSAS classloader. The asadmin script will be modified to pickup a SGCS client jar file as well.

7 Documentation Requirements

Following are the set of documentation that will need to be either enhanced or newly written

- Developers Guide.
- Administration Guide.
- HA administration Guide.
- Performance Tuning Guide.
- Deployment Planning Guide.

8 Open Issues

None

Appendix 2

```
<!ENTITY % boolean "(yes | no | on | off | 1 | 0 | true | false)">


<!-- load balancer Configuration
    Configure load balancer to load balance the request. It contains
    configuration details related to cluster, and
    properties related to load balancer.
```

```
-->

<!ELEMENT loadbalancer (cluster*, property*)>
<!ATTLIST loadbalancer http-policy CDATA "round-robin"
                       sip-policy CDATA "From-tag,To-tag,Call-id"
                       dcr-file CDATA "">


<!--  Cluster Configuration
      Provides configuration information related to all clusters to
      which loadbalancer would route the requests.

      name      Required attribute that defines the name of the cluster.

      self-loadbalance
            Specifies whether configured cluster self load balances incoming
            requests to itself. If its configured to do so, load balancer is an
            intrinsic component of  the participating server instances in the cluster.
-->

<!ELEMENT cluster (instance*, web-module*, property*)>
<!ATTLIST cluster name CDATA #REQUIRED
                  self-loadbalance %boolean; "true">


<!--   Server instance configuration
      name        identifies the instance
      enabled     specifies whether instance is active (enabled)
                  for requests to be load balanced to it.

      disable-timeout-in-minutes  specifies the quiescing timeout interval in
                                  minutes, upon elapse of which load balancer
                                  would close all the open connections related
                                  to the instance being disabled and no further
                                  requests would be routed to the instance.
                                  Default value would be 31 minutes (i.e. more
                                  than the default session idle timeout which is
                                  30 minutes).

      listeners                   Required attribute that specifies the SIP and HTTP
                                  listeners for the instance. This attribute can be used to
                                  specify multiple listeners for a instance delimited with a
                                  space. For example, "sip://server1:5060 http://server1:8080
-->
```

```
<!ELEMENT instance EMPTY>
<!ATTLIST instance
                name              CDATA #REQUIRED
                enabled           %boolean; "true"
                disable-timeout-in-minutes  CDATA     "31"
                listeners         CDATA     #REQUIRED>


<!-- Deployed Web Modules (Applications).

   context-root   context root of the application deployed

-->

<!ELEMENT web-module EMPTY>
<!ATTLIST web-module
                  context-root    CDATA     #REQUIRED>


<!ELEMENT property EMPTY>
<!ATTLIST property       name      CDATA      #REQUIRED
                    value     CDATA      #REQUIRED>
```