Roles, Services and Processes

Roles

A '"Role"' is a logical collection of sipXecs services. Since some services work closely together, the role abstraction simplifies administration by collecting these related services into groups - rather than having to assign individual services to servers and understand the (constantly changing) relationships between them, the administrator just assigns roles to servers and lets sipXconfig worry about the details.

A sipXecs server can be assigned to provide various PBX related capacities: ACD (Automated Call Distribution), SIP Trunking, Conferencing, Instant Messaging, Management, SIP Router, and Voicemail. The administrator configures these capabilities by assigning roles to defined servers. These capacities are assigned in the System>Server>Server Roles screen. When there are multiple servers in a system, a role might be assigned to a dedicated server to enhance performance. By default, all roles are enabled in a server.

Role Definitions

During initialization, roles are configured by the file *sipXconfig/neoconf/src/org/sipfoundry/sipxconfig/service/bundle.beans.xml*. Each role is a bundle bean, and all beans are children of an abstract bundle. The "autoEnable" properties controls whether a bundle (role) is enabled or not. Since the abstract bundle has the property "autoEnable" set to true, all roles are enabled by default.

```
<bean id="abstractBundle" abstract="true">
<property name="autoEnable" value="true" />
<property name="resetAffectLocation" value="false" />
</bean>
```

The following table relates the Role as seen in the GUI to the bundles names and bean id in the bundle.beans.xml file.

Role Label	Bundle Name	Bundle Bean Id
ACD	acd	acdBundle
SIP Trunking	borderController	borderControllerBundle
Conferencing	conference	conferenceBundle
Instant Messaging	im	imBundle
Management	management	managementBundle
Primary SIP Router	primarySipRouter	primarySipRouterBundle
Redundant SIP Router	redundantSipRouter	redundantSipRouterBundle
Voicemail	voicemail	voicemailBundle

Bundle Properties

All bundle beans shares the same SipxServiceBundle class implementation. Besides the unique bean id, all bundle beans' constructor-arg value is passed into the SipxServiceBundle class to identify the bundle.

The following table describes some of the properties in a bundle bean:

Property Name	Description	Default Value
autoEnable	This property specifies if the role can be enabled automatically during initial installation. A role can be set to autoEnable but not automatically enabled on a location instance, depending on the bundle's onlyPrimary and onlyRemote properties.	true
max	This property limits the maximum number of locations that can be assigned the role in a system. For example, there can be only one (and only one) primarySipRouter location in a system.	none
min	This property sets the lower limit on the number of location that is assigned the role in a system. For example, there must be one (and only one) primarySipRouter location in a system.	none
onlyPrimary	Indicates whether the role can only be assigned to the primary server.	none
onlyRemote	Indicates whether the role can only be assigned to a remote server.	none
resetAffectLocation	If true, the location whose role is being changed is stored in database via hibernate and listeners can take specific actions. Currently only the ACD bundle sets this property to true. The AcdContextImpl will create or delete the ACDServer as required.	false

By reading a role's property values, one can note the supported instances of a role in a sipXecs domain. For example, since the min and max properties for the Management role is set to 1, there must exist one and only one Management role in a domain. Likewise, one can infer that the Redundant SIP Router is optional, and no more than two can exist in a domain.

Services

A Role requires one or more services to be configured and running. For example, the Voicemail role requires the "Voicemail and Auto Attendant", "Media Services", and "Voicemail MWI" services to be running. One service might be required by multiple roles. For example, "Media Services" is required by both Conferencing and Voicemail roles.
Once a new role has been assigned, the associated service will be started automatically if not already running. Likewise, if a role has been unassigned, services no longer required will be stopped.
Services can be started, stopped or restarted manually in System>Server>Services screen. In the same screen, each service's configuration screens can be accessed.

Service Definition

Services are defined in the file *sipXconfig/neoconf/src/org/sipfoundry/sipxconfig/service/service.beans.xml*. Each service is described as a bean and each service bean is a child of the abstract sipxService bean. The abstract service bean sipxService defines properties relevant to all services and its corresponding java class is SipxService. Likewise, each concrete service bean has a corresponding java classe which reflects the bean hierarchy. For example, the sipxSupervisorService bean is derived from the abstract sipxService bean, and the corresponding SipxSupervisorService bean class is derived from the abstract SipxService class.
The SipxService class is pivotal because it ties the service to its setting model and and the Hibernate ORM by extending BeanWitFurthermore, it implements the ServiceLifeCycle interface to allow all SipxService subclasses to customize and control their own life cycle behavior.
The service bean xml also defines the service manager bean (sipxServiceManagerImpl) and the class (SipxServiceManagerImpl) which manages the services and the association to the bundles.
Each service bean has a "bundles" property which is a list of bundles (roles) which depend on the service. For example, the sipxProxyService is required by both primary and secondary SIP Router bundles:

```
<property name="bundles">
<set>
<ref bean="primarySipRouterBundle" />
<ref bean="redundantSipRouterBundle" />
</set>
</property>
```

The service beans xml contains information on the services' associated process name, model files, configurations and dependent bundles. The modeling files contain setting descriptions and these drive the configuration parameters as well as GUI screens for each service. On the System>Server>Services screen, each service have a link to its configuration screen. The configuration screen will allow the users to update any configurable parameters described in the model file.

```
<bean id="sipxProxyService" class="org.sipfoundry.sipxconfig.service.SipxProxyService" parent="sipxService"
scope="prototype">
<property name="processName" value="SIPXProxy" />
<property name="modelName" value="sipxproxy.xml" />
<property name="modelDir" value="sipxproxy" />
<property name="editable" value="true" />
<property name="configurations">
<list>
<ref bean="sipxProxyConfiguration" />
<ref bean="peerIdentitiesConfiguration" />
</list>
</property>
<property name="bundles">
<set>
<ref bean="primarySipRouterBundle" />
<ref bean="redundantSipRouterBundle" />
</set>
</property>
</bean>
```

Service beans can be described in its own bean xml file. For example, the sipxOpenfireService bean is described in sipxOpenfire-service-models.beans.xml. However, the bean is derived from sipxService, and defines the same properties as the other service beans. More specifically, the property "bundles" is also configured to identify the set of bundles (imBundle) which requires the service.

Service Bean Properties

The table below shows the major set of properties in the *service.beans.xml* and how each property is used by sipXecs.

Property Name	Description	Value Example
configDir	The directory where the service configurations are found. This is defaulted to /etc/sipxpbx under the installation directory for all services. For the source code directory, it is at sipXconfig/neoconf/etc	{sysdir.etc}
logDir	The directory where the log files are found. This is defaulted to /log/sipxpbx	{sysdir.log}
bundles	The set of bundles or roles, which depends on this service. The set is a set of bundle bean is (see Role Definitions section above.	
configurations	The list of configuration files required by this service. The value is a list of references to configuration file beans	sipxSupervisorConfiguration
coreContext	References the core context bean	coreContext
modelDir	The subdirectory directory where the model file reside.	acd freeswitch sipxconfig
modelName	Model file name	sipxacd.xml freeswitch.xml sipxconfig.xml
processName	The name of the process associated with the service	ACDServer FreeSWITCH ConfigServer
restartable	Specifies if the service can be restarted by sipxconfig GUI. Default is true.	true/false
editable	Specifies if the service is editable. Defaults to true but currently not used?	true/false

Service Configuration Properties

Each service has related configuration files. The configuration files are described within the _service.beans.xml_ file as configuration file beans and referenced in the service bean definition.
For the configuration files defined within the services bean xml, the files are either XML files or velocity template generated files. Note that velocity templates can generate files of any format, including XML files. The majority of the configuration file beans are derived from the ServiceConfigurationFile bean, which has SipxServiceConfigurationFile as the bean class. SipxServiceConfigurationFile extends TemplateConfigurationFile, thus provides the velocity template capabilities. The remaining configuration file beans' bean classes are derived from the java class XmlFile, which provides the XML document capabilities. Configurations, which are velocity based, will setup its velocity context by overriding setupContext while configurations which are XML based will set up its XML document by overriding getDocument.
All configuration file bean classes are children of AbstractConfigurationFile.
The table below shows the major set of properties in the *service.beans.xml* which are related to service configuration files.

Property Name	Description	Value Example
restartRequired	Specifies if configuration file update requires a restart of the service. Value defaulted to true in the AbstractConfigurationFile. This property is used by ServiceConfiguratorImpl when replicating configuration files for determining whether the service will be marked for restart.	true/false
template	The velocity template file of the configuration file.	freeswitch/freeswitch.xml.vm
name	The name of the configuration file	freeswitch.xml
directory	The directory where the configuration file resides. The default is {sysdir.etc} which translate to /etc/sipxpbx	{sysdir.etc}/freeswitch/conf

Roles and Services Associations

The table below shows the associated services for each role. For the "Required Service" column, services which are required by only one role is shown in **bold**. Services required by more than one role (shared) are in *italic*, except for services shared between the Primary and Secondary SIP Routers, which are in normal font.

The labels in the "Role" and "Required Service" columns are as seen on the System>Servers>Services GUI screen and configured in the sipXconfig-web. properties file. The Bean Id column is the SipxService bean which is defined in the service.bean.xml file. The "Process Name" column is from the "processName" property of each SipxService definition. The last column "Process/Class" shows either the Linux executable (in lower case), the java class to be loaded (fully qualified java class name), or a ruby mainline entry point.

Role	Required Service	Bean Id / class	Process Name	Process / Class
Primary SIP Router	Shared Appearance Agent	SipxSaaService	SharedAppearanceAgent	sipxsaa
	Park	SipxParkService	ParkServer	sipxpark

	Paging	SipxPageService	PageServer	ort.sipfoundry.sipxpage.SipXpage
	Media Relay	SipxRelayService	SipXrelay	org.sipfoundry.sipxrelay.SymmitronServer
	SIP Registrar	SipxRegistrarService	SIPRegistrar	sipregistrar
	SIP Proxy	SipxProxyService	SIPXProxy	sipXproxy
	Presence	SipxRlsService	ResourceListServer	sipxrls
	Call Control	SipxRestService	SipXrest	org.sipfoundry.sipxrest.RestServer
Redundant SIP Router	Media Relay	SipxRelayService	SipXrelay	org.sipfoundry.sipxrelay.SymmitronServer
	SIP Registrar	SipxRegistrarService	SIPRegistrar	sipregistrar
	SIP Proxy	SipxProxyService	SIPXProxy	sipXproxy
	CDR HA Tunnel	SipxCallResolverAgentService	CallResolver-Agent	stunnel
Management	**CDR**	SipxCallResolverService	CallResolver	sipxcallresolver-2.0.0/lib/main.rb
	Statistics	SipxMrtgService	sipXmrtg	/usr/bin/mrtg
	Configuration	SipxConfigService	ConfigServer	org.mortbay.jetty.Server /home/carson/scs3/INSTALL/etc/sipxpbx/sipxconfig-jetty.xml
	Phone Provisioning	SipxProvisionService	sipXprovision	org.sipfoundry.sipxprovision.SipXprovision (starts org.sipfoundry.sipxprovision.auto.Servlet)
	Call Control	SipxRestService	SipXrest	org.sipfoundry.sipxrest.RestServer
SIP Trunking	**SIP Trunking**	SipxBridgeService	SipXbridge	org.sipfoundry.sipxbridge.Gateway
ACD	**ACD**	SipxAcdService	ACDServer	sipxacd
	ACD Agent Status	SipxPresenceService	PresenceServer	sipxpresence
	ACD Statistics	SipxConfigAgentService	ConfigAgent	/usr/bin/ruby sipxconfig-agent
Call Center	**Call Center**	SipxOpenAcdService	OpenACDServer	/opt/OpenACD/bin/openacd
Conferencing	**Conference Recording**	SipxRecordingService	sipXrecording	org.sipfoundry.sipxrecording.SipXrecording
	Media Services	SipxFreeswitchService	FreeSWITCH	/usr/local/freeswitch/bin/freeswitch
Voicemail	**Voicemail MWI**	SipxStatusService	SIPStatus	sipstatus
	Voicemail and AutoAttendant	SipxIvrService	sipXivr	org.sipfoundry.sipxivr.SipXivr
	Media Services	SipxFreeswitchService	FreeSWITCH	/usr/local/freeswitch/bin/freeswitch
Instant Message	**Instant Messaging**	SipxOpenfireService	SipXopenfire	/opt/openfire/bin/openfire

Process

Each service bean definition contains a processName property which defines the name of the process associated with the service. When a service is to be started, stopped or restarted, SipxProcessContextImpl class retrieves the process name from the SipxService instance so that the correct process can be managed on the host. At the same time, SipxProcessContextImpl will ask AuditLogContextImpl to write an audit log into sipxconfig.log using the processName. Similarly, ConfigVersionManagerImpl uses the processName to uniquely identifies the process when setting the config version of the process.
All process description xml files are installed in sipxecs/process.d/. There are total of 23, one for each service, except for the sipxAlarmService which shares the sipxsupervisor process. The name element of each file corresponds to the processName in the service.bean.xml. This is how a SipxService is correlated with its process.
For example, the xml snippet below is from the file share/sipxecs/process.d/freeswitch-process.xml. The name element "FreeSWITCH" correlates with the property value "FreeSWITCH" for the sipxFreeswitchService bean in service.beans.xml file.

```
<?xml version='1.0' encoding='iso-8859-1' standalone='yes'?>
<sipXecs-process xmlns='http://www.sipfoundry.org/sipX/schema/xml/sipXecs-process-01-00'>
<name>FreeSWITCH</name>
<version>4.3.0</version>
<commands>
<configtest>
<execute>/home/carson/scs3/INSTALL/bin/freeswitch.sh</execute>
<parameter>--configtest</parameter>
</configtest>
<start>
<execute>/home/carson/scs3/INSTALL/bin/freeswitch.sh</execute>
</start>
<stop>
<execute>/home/carson/scs3/INSTALL/bin/freeswitch.sh</execute>
<parameter>--stop</parameter>
</stop>
</commands>
```

ConfigureBundlesPanel

This class handles the role assignment validation when updating the role assignment. This will start a chain of invocations which will:

- validate the new bundle assignment against bundle properties - see SipxServiceManagerImpl.setBundlesForLocation() where filter() validates the onlyPrimary and onlyRemote properties and verifyBundleCardinality() verifies the max/min properties against the cluster - see SipxServiceManagerImpl.setBundlesForLocation()'s filter() and verifyBundleCardinality().
- prepare the services for restart/stoppage by calling the LifeCycle methods. For service to be started, the onInit() is called, for services to be stopped, the onDestroy() will be called. - see Location.resetBundles()
- stopping services no longer required -see SipxProcessContextImpl.manageServices()
- replicate service configuration files - see serviceConfigurator.enforceRole()

Log Files

As mentioned previously, the service definition in the service.bean.xml file contains the processName property. This property is outputted to the sipxconfig.log file to trace service process activations. For example, if you assign the Voicemail role on a server, you will see an auditlog similar to below:

```
auditlog:"Replicated freeswitch.xml to bcmsl2072.ca.nortel.com"
auditlog:"Started FreeSWITCH  on bcmsl2072.ca.nortel.com"
auditlog:"Started SIPStatus  on bcmsl2072.ca.nortel.com"
auditlog:"Started sipXivr  on bcmsl2072.ca.nortel.com"
```

where sipXivr, SIPStatus, and FreeSWITCH are all process names of SipxService defined the service.beans.xml file to be a required service for the Voicemail role.

Display String Properties

Bundles and services are show in the GUI with names that do not match the source code (e.g. java classes and xml files). These GUI names are show in the System>Server>Server Roles and System>Server>Services sipXconfig screens. The GUI names are set in the sipXconfig web applicaton properties file sipXconfig-web.properties and can help to correlate with the bundle and services defined in bundle.beans.xml and service.beans.xml. The relevant bundle and service GUI label are shown below:

```
bundle.management=Management
bundle.primarySipRouter=Primary SIP Router
bundle.redundantSipRouter=Redundant SIP Router
bundle.voicemail=Voicemail
bundle.conference=Conferencing
bundle.borderController=SIP Trunking
bundle.acd=ACD
bundle.im=Instant Messaging

label.sipxAcdService=ACD
label.sipxBridgeService=SIP Trunking
label.sipxCallResolverService=CDR
label.sipxCallResolverAgentService=CDR HA Tunnel
label.sipxConfigService=Configuration
label.sipxConfigAgentService=ACD Statistics
label.sipxFreeswitchService=Media Services
label.sipxImbotService=MyBuddy
label.sipxIvrService=Voicemail and Auto Attendant
label.sipxMrtgService=Statistics
label.sipxParkService=Park
label.sipxPageService=Paging
label.sipxPresenceService=ACD Agent Status
label.sipxProxyService=SIP Proxy
label.sipxRecordingService=Conference Recording
label.sipxRelayService=Media Relay
label.sipxRegistrarService=SIP Registrar
label.sipxRlsService=Presence
label.sipxSaaService=Shared Appearance Agent
label.sipxStatusService=Voicemail MWI
label.sipxSupervisorService=Supervisor
label.sipxOpenfireService=Instant Messaging
label.sipxRestService=Call Control
label.sipxProvisionService=Phone Provisioning
```