HP Mobile Subscriber Activation Solution Pack v7.0.0

Delivery Guide

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c) (1,2).

Copyright Notices

Trademark Notices

Java™ is a registered trademark of oracle and/or its affiliates.

Linux is a U.S. registered trademark of Linus Torvalds.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of the Open Group.

Windows® and MS Windows® are U.S. registered trademarks of Microsoft Corporation.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Table of Contents

1 Architectural Introduction 6
1.1 Northbound Interface 6
1.2 Service Catalog 8
1.3 Activation Workflows 9
1.4 Services Specifications 10

2 Installation 11
2.1 Prerequisites 11
2.2 Install MSA 11
2.3 Configuring MSA installation 12
2.3.1 HPSA configuration 12
2.3.1.1 SOSA Module 12
2.3.1.2 WSC Module 12
2.3.1.3 TMPC Module 14
2.3.1.4 SmartBean Search Module 14
2.3.1.5 CORBA Module 15
2.3.2 SOSA Configuration 15
2.3.2.1 NGWS protocol adapter 15
2.3.2.2 Bulk protocol adapter 16
2.3.3 Report Module Datasource Configuration 17
2.4 Uninstall MSA 17
2.5 Upgrade MSA 18

3 Customizing the solution 19
3.1 Use Case 1: Add a new service 19
3.1.1 Use case analysis 19
3.1.2 Step 1: Define the Northbound interface 19
3.1.3 Step 2: Create the target adaptor 21
3.1.4 Step 3: Create a new endpoint 21
3.1.5 Step 4: Develop the activation workflow 22
3.1.6 Step 5: Define the error management 24
3.1.7 Step 6: Test the service 25
3.2 Use Case 2: Customize the northbound interface 26
3.2.1 Use case analysis 26
3.2.2 Step 1: Edit the service parameters 26
3.2.3 Step 2: Test the service 28
3.3 Use Case 3: Customize the command template 28
3.3.1 Use case analysis 28
3.3.2 Step 1: Edit the command template 29
3.3.3 Step 2: Test the service 30
3.4 Use Case 4: Customize the error management 30
3.4.1 Use case analysis 31
3.4.2 Step 1: Edit the error 31
3.5 Use Case 5: Create a new report 32
3.5.1 Use case analysis 32
3.5.2 Step 1: Create the SQL query 32
3.5.3 Step 2: Edit the report module configuration file 33
3.5.4 Step 3: Test the new report 34
3.6 Use Case 6: Combine two services 34
3.6.1 Use case analysis 35
3.6.2 Step 1: Create the new service request in the service catalog 35
3.6.3 Step 2: Test the service 37

4 High performance architecture 39

5 Template management 41

5.1 Managing a template from the user interface 41
5.1.1 CommandRoutine Operations 43
5.1.1.1 Search/List 43
5.1.1.2 Edit 43
5.1.2 Template Operations 45
5.1.2.1 Search/List 45
5.1.2.2 Create 45
5.1.2.3 Import 47
5.1.2.4 Edit 49
5.1.2.5 Export 50
5.1.2.6 Delete 52
5.2 Querying Command Templates from Workflows 53

6 Web Service Connectivity 54
6.1 Customizing Service Descriptors 54
6.2 Paths 56
6.3 Making Web Service Requests from Workflows 56

7 Report Module 59
7.1.1 Accessing the Report Module Web UI 59
7.1.2 Available Operations 59
7.2 Report Module configuration 60
7.2.1 Query Section 60
7.2.1.1 Name Section 61
7.2.1.2 Description Section 61
7.2.1.3 Order Section 61
7.2.1.4 Max Length Result section 61
7.2.1.5 SQL Section 61
7.2.1.6 Input Section 62
7.2.1.7 Redirections Section 62
7.3 Redirections (Types of Representations) 63

8 SmartBean Search Module 66
8.1 Indexed Bean Configuration 66
8.2 Workflow Nodes 66
8.2.1 SBSGetBeanNode 66
8.2.1.1 Wildcard Feature 68
8.2.2 SBSGetBeanWithPatternNode 68
8.2.3 SBSGetBeansNode 70

9 Bulk Activations 72
9.1 Protocol Adapter Configuration 72
9.2 Request File Format 74
9.2.1 Request Parameters 74
9.2.2 Activation Requests 74
9.3 Response File Format 75
9.4 Web Interface 75
9.4.1 Importing a Bulk Activation Request File 75
9.4.2 Listing Pending Bulk Activations 76

Appendix 77

In This Guide

This guide contains the instructions to deploy Mobile Subscriber Activation (MSA) HP Business Solution Pack v7.0.0 and to adapt off-the-shelf solution to specific environments.

This manual falls in the following parts:

- Architectural Introduction.
- Installation processes.
- Configuration of the features available off-the-shelf in MSA.
- How to extend the set of MSA functionalities.

Audience

The audience for this guide is the Solutions Integrator (SI). The SI should have a combination of some or all of the following capabilities:

Understands and has a solid working knowledge of:

- UNIX® commands.
- Windows® system administration.

Understands networking concepts and language.

Is able to program in Java™ and XML.

Understands security issues.

Understands the customer's problem domain.

References

Other manuals for HP Mobile Subscriber Activation Solution Pack v7.0.0:

- HP Mobile Subscriber Activation SP v7.0.0 – Services Specifications.
- HP Mobile Subscriber Activation SP v7.0.0 – User's Guide.

Manuals for HP Service Activator V7.0 or upper:

- HPSA Service Activator Installation Guide.
- HPSA Extension Pack Installation Guide.

1 Architectural Introduction

MSA is a preconfigured solution based on HP Service Activator to activate mobile services.

MSA offers a set of configured mobile services that can be used off of the shelf as well as a set of modules that allow the inclusion of new services in an easy way and on the fly. In the way MSA could be intended as a framework where new services could be included without the need of compile new source code.

Additionally due MSA is intended for the mobile environment, is a solution that supports a high performance as well as is a scalable solution

The next figure shows the MSA's components in a logical view. This view shows the process that a service request follow since it is received in the northbound interface until the services are activated into the target.

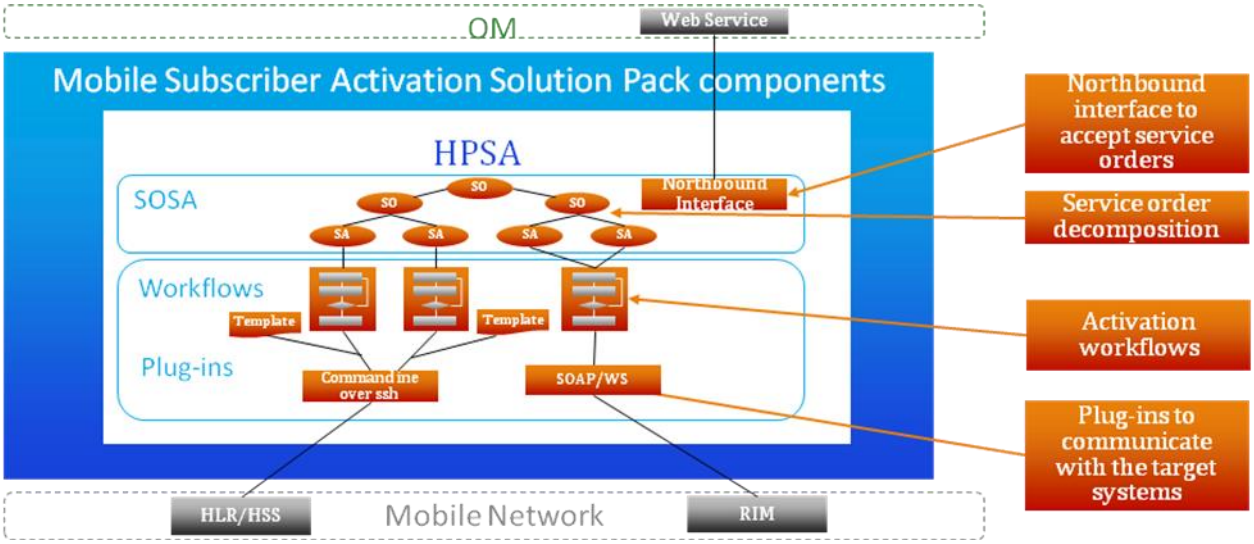


Figure 1 – MSA architecture

In the next bullets are explained each of these components.

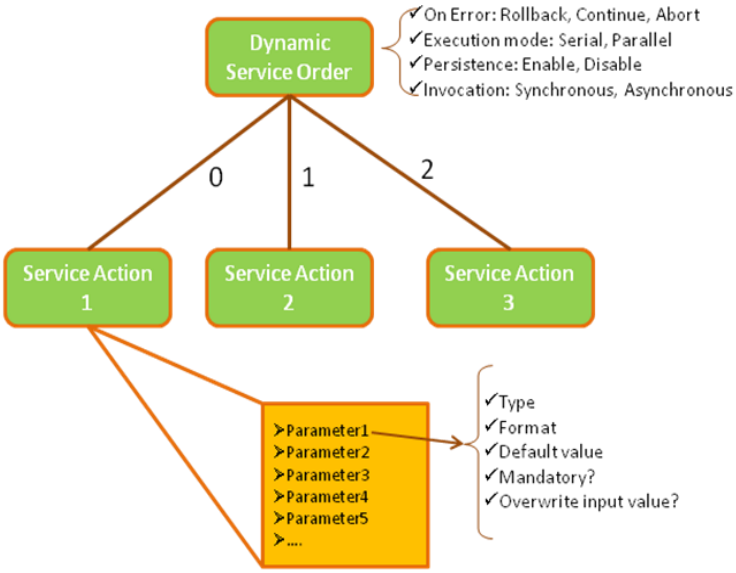
1.1 Northbound Interface

Out-of-the-box MSA offers a generic northbound interface to integrate it with an order management system. This generic northbound interface is the ones used to integrate MSA with the HP Order Management Mobile Solution Pack.

This generic northbound interface defines a service request as a service identifier and a set of parameters define as pairs name/value.

```
<service>
  SERVICE ACTION 1
    Parameter1:value1
    Parameter2:value2
    Parameter3:value3
    Parameter4:value4
</service>
```

This generic definition allows using the same interface for all the services defined in the service catalog. Additionally this interface allows combining multiple services in a single request and allows the user defined the execution mode. The next picture shows an example of complex service request:

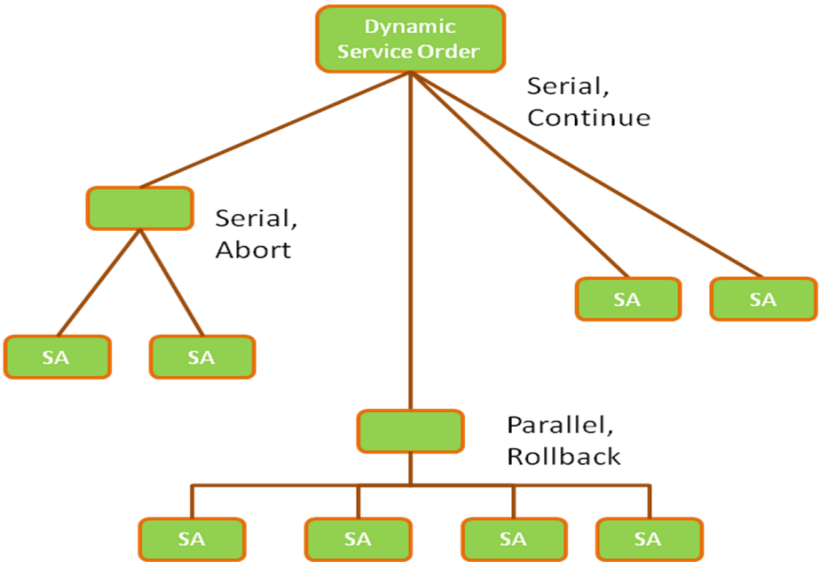



```
<services
 DYNAMIC SERVICE ORDER
mode="parallel"
onerror="rollback“
 persistence="enable">
  <service>
    SERVICE ACTION 1
      Parameter1:value1
      Parameter2:value2
      Parameter3:value3
      Parameter4:value4
  </service>
  <service>
    SERVICE ACTION 2
      Parameter1:value1
  </service>
  <service>
    SERVICE ACTION 3
      Parameter1:value1
      Parameter2:value2
      Parameter3:value3
  </service>
  <services/>
```

On the left if represented the logical structure of the service, in this case the complex service is composed by 3 single services represented by Service Action 1, 2 and 3. This single service must be defined in the service catalog.

On the right side of this figure is represented the pseudo structure of the request that must be used to invoke this service.

Due the composition of a complex service request is generic the complexity of it could be as higher and the customer need. The next figure shows an example of complex service request, where the single service actions are combined in some different ways.

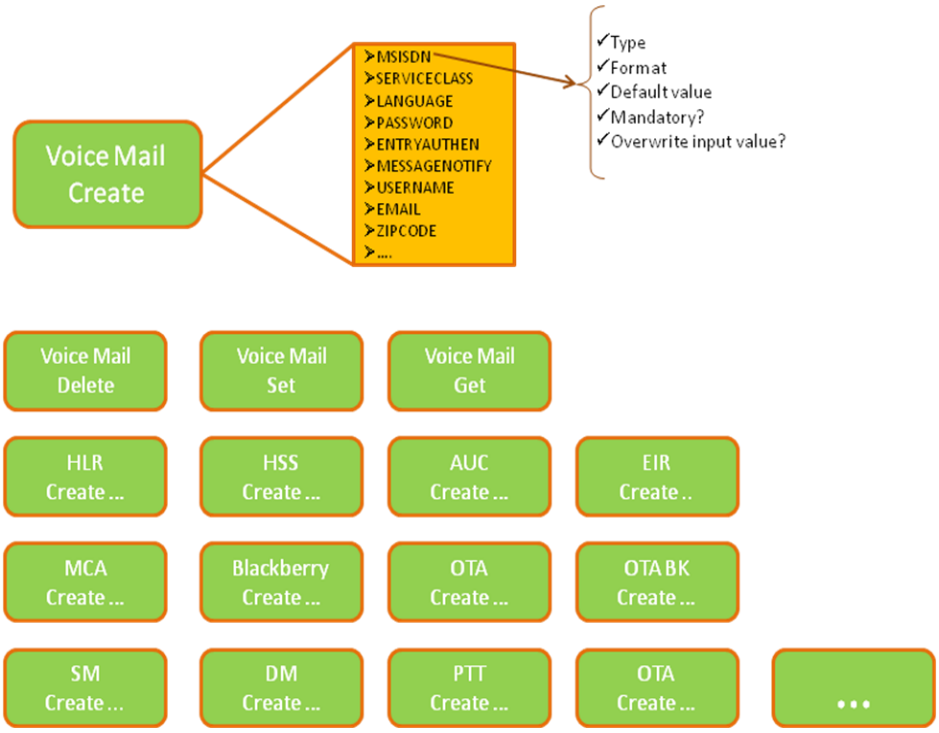


1.2 Service Catalog

MSA offers off-the-shelf a set of mobile services. These services cover most off the services used by the Mobile Service Provides.

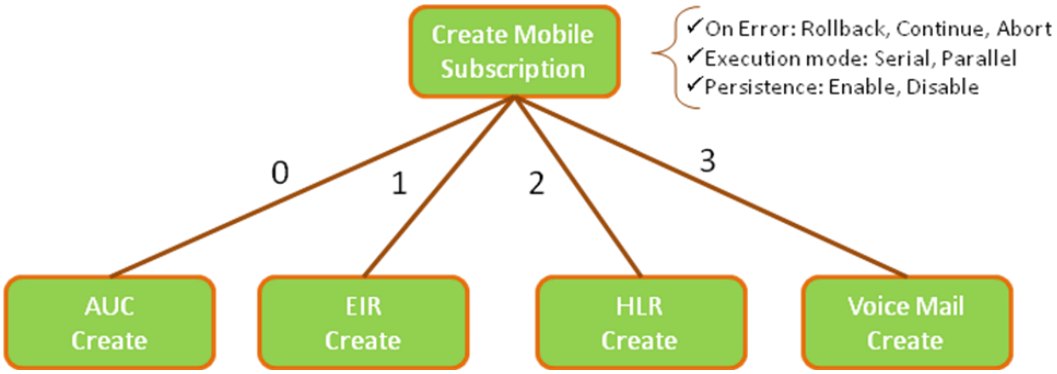
As MSA is between the commercial representation of a service and the physical implementation the service on the network, MSA offers an abstraction layer of the physical definition of a service in the targets.

The next picture shows most of the services included in the service catalog:



For each service is defined the list of parameters needed to activate the service in target. For more details about the parameters defined for each service see the Appendix on this document.

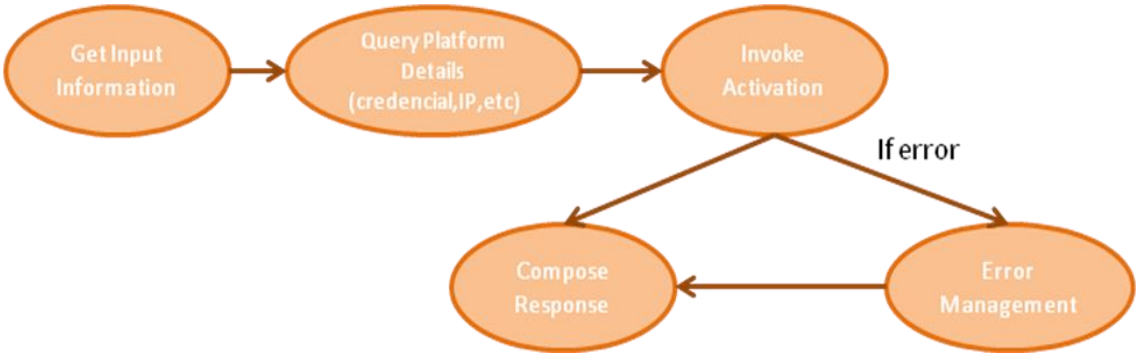
Each service include in MSA is defined in the service catalog as a single service action therefore can be combined to create a complex service request composed by multiples single services. The next picture shows an example of a complex service request to create a typical mobile subscription on the network:



1.3 Activation Workflows

The business logic used to activate the services included in MSA is implemented in the activation workflow; these are HP Service Activator workflows.

Each workflow executes a set of step to complete the activation of a service. The next figure shows the mains steps of all the MSA's workflows:



This is the description of each step:

First step is the reception of the input parameters received in the northbound interface, next based on the parameters the workflow select the correct platforms where the service must to be activated. The selection of the platform is based on a predefined mappings, that allow the customer distribute the subscriptions into the available platforms.

Once the right platform has been selected, the workflow queries some characteristic of this platform as the access information and the target version. Based on this information the workflow selects the correct plugin to execute the activation into the target.

The activation is execute with a different node depends on the technology supported for the target.

After the activation took place, the workflow receive the response from the target, this response is evaluated in the workflow and returned to the northbound systems. In case of error the workflow translate the target error code into a meaningful code and description and return it to the northbound system.

In the workflow if selected all the information that will be stored in the history database.

All the queries executed in the workflows are done using a cache avoiding access directly to the database.

1.4 Services Specifications

These are the services include in the MSA service catalog:

- Activation of Voice and Data Mobile services on HLR over EMA of Ericsson.
- Activation of subscriptions on AUC and EIR over EMA of Ericsson.
- Activation of LTE (4G) services over HP HSS.
- Activation of Number Portability service (MNP) over Huawei targets.
- Activation of Value-added services (VAS)
    - Voice Mail (VM) and Missed Call Alerts (MCA) over ZTE targets
    - Push to talk (PTT) service over Kodiak targets.
    - OTA and OTA Backup service over Gemalto targets
    - Blackberry service over RIM targets.
- Activation of operator services, these services are used by the Mobile Service Provider to manage some aspect of the subscriptions.
    - Device Manager (DM) over Gemalto targets.
    - Service Manager (SM) over Gemalto targets.
    - Service Delivery Platform (SDP) over Huawei targets.

For all of these services MSA offers the translation from the logical service definition offer in the northbound interface, and the physical activation of the service into the network targets.

Additionally MSA offer some other features related with the services:

- Service Definition, for each service are defined all the operation supported by itself and also are defined all the parameters supported for each operation. The parameters definition includes type and format validations as well as the capability to define a default value.
- Error Management, for each service MSA manages all the potential errors returned by the target and a translation into a meaningful description of it.

For more information about the service definition, please review the MSA delivery guide (**HP Mobile Subscriber Activation SP v7.0.0 – Services Specifications**), there you can found a complete definition of all the services include in MSA.

2 Installation

2.1 Prerequisites

Prerequisites for MSA Solution are:

- HP Service Activator 7.0 or upper and the latest available patch successfully installed.
- The CRModel solution must be deployed on HPSA.
- The Extension Pack 7.0 or upper for HPSA must be installed.

2.2 Install MSA

In order to deploy the MSA solution, ensure that the prerequisites described in the previous section are met and then follow the steps below:

1. Copy the provided MSA.zip file into the $ACTIVATOR_OPT/SolutionPacks directory.
2. Ensure that HPSA is not running, otherwise stop it now. For more information on stopping and starting the HPSA service, refer to section "**Starting and Stopping Service Activator**" in the *HPSA Installation Guide*.
3. Run the deployment manager from the $ACTIVATOR_OPT/bin directory.
4. Click on System Database Connection and enter database credentials, then click OK.
5. Click on Local Deployment.
6. Click on Import Solution.
7. Select from zip/tar file and browse for SolutionPacks/MSA.zip.
8. Click on Import.
9. Click on Deploy Local Solution.
10. Select the MSA Solution.
11. Select the appropriate deployment file depending on the version, the database and the operating system (see the table below).

Deployment file	HPSA version	Database	Operating System	Test
deploy_7_0_0_oracle.xml	7.0.0	Oracle	Windows	No
deploy_7_0_0_ppas.xml	7.0.0	PPAS (Postgres Plus Advanced Server)	Windows	No
deployTest_7_0_0_oracle.xml	7.0.0	Oracle	Windows	Yes
deployTest_7_0_0_ppas.xml	7.0.0	PPAS	Windows	Yes
deploy_upgrade_6_x_to_7_0_0.xml*	7.0.0	Any	Windows	Any

deployUnix_7_0_0_oracle.xml	7.0.0	Oracle	Unix	No
deployUnix_7_0_0_ppas.xml	7.0.0	PPAS	Unix	No
deployUnixTest_7_0_0_oracle.xml	7.0.0	Oracle	Unix	Yes
deployUnixTest_7_0_0_ppas.xml	7.0.0	PPAS	Unix	Yes
deployUnix_upgrade_6_x_to_7_0_0.xml*	7.0.0	Any	Unix	Any

12. Check Create inventory tables.
13. Click on Deploy solution.

Deployment of the solution is now complete but additional configuration steps are required. Please review the Configuration section for more information.

2.3 Configuring MSA installation

This section describes the configuration steps that the SI must follow to make the MSA solution fully functional.

2.3.1 HPSA configuration

MSA include some modules than need to be included into the HPSA configuration. A HPSA module is (from the HPSA Overview document):

"Modules are used for all functions which interface the workflow manager to its environment, like synchronizing with external sources of input, sending messages and emails, accessing the database, interacting with the transaction manager to execute activation tasks, etc."

2.3.1.1 SOSA Module

In order to get a correct communication between HPSA and SOSA is necessary to configure the response module in $ACTIVATOR_ETC/config/mwfm.xml file.

```
<Module>
<Name>sosa_async_responser</Name>
<Class-Name>com.hp.spain.engine.module.sosa.SosaAsyncResponserImpl</Class-Name>
<Param name="errors_async_persistence_file" value="$ACTIVATOR_VAR/tmp/errors_async_responser.dat"/>
        <Param name="write_in_queue" value="false"/>
        <Param name="sosa_async_queue" value="sosa_async_queue"/>
</Module>
```

Don't forget to replace the $ACTIVATOR_VAR with the correct installation path.

To apply this change is necessary to reload the HPSA configuration for the HPSA user interface.

2.3.1.2 WSC Module

To enable the WSC module is necessary declare it in the mwfm.xml file. This is the minimal configuration that must be set for this module.

```
<Module>
        <Name>wsc</Name>
        <Class-Name>com.hp.ov.activator.mwfm.engine.module.wsc.WSCModule</Class-Name>
        <Param name="database_module" value="db"/>
</Module>
```

Note: the value for the database_module must exist in the mwfm.xml configuration. By default this module exists on HPSA.

Some additional parameter could be configured for this module. Next table show the allowed configuration the WSC module.

Parameter	Required	Description	Reconfigurable	Default
database_module	Yes	The name of the database module to use in order to retrieve WSC service and endpoint definitions.	Yes	none
queue_class	No	This can be set to the com.hp.ov.activator.mwfm.module.WeightedEngineQueue, com.hp.ov.activator.mwfm.module.SimpleEngineQueue, or com.hp.ov.activator.mwfm.module.PriorityEngineQueue. The WeightedEngineQueue use the PRIORITY case-packet variable in a weighted way to prioritize the items on which the activation threads operate. Items that have the same priority will be processed in FIFO order. The SimpleEngineQueue will not do any prioritization of activation requests. They will be processed in FIFO order. The PriorityEngineQueue uses the PRIORITY case-packet variable to prioritize the items on which the activation threads operate. Items that have the same priority value will be processed in FIFO order.	No	com.hp.ov.activator.mwfm.engine.module.WeightedEngineQueue
queue_name	No	The name of the queue the job will be waiting in while a	No	external_request_queue

		request is being processed		
retry_count	No	Number of times to retry processing of the external request.	No	3
retry_interval	No	Interval between each retry. Defined in milliseconds.	No	10000
min_threads	No	The minimum number of threads to maintain for processing request.	No	1
max_threads	No	The maximum number of threads to maintain for processing request.	No	3

2.3.1.3 TMPC Module

This module enables workflows to manage a template cache with the information of template data. HPSA will load this cache when starting the server.

In order to arrange properly, the module must be configured in the Micro-Workflow Manager ("mwfm.xml"). We must set the parameter specified in the following table.

Parameter	Required	Description	Reconfigurable	Default
database_module	Yes	The name of the database module to use in order to retrieve template information.	No	None

The way we configure this module in the mwfm.xml is adding the following code snippet into mwfm.xml in order to enable the new feature.

```
<Module>
   <Name>TMPCModule</Name>
   <Class-Name>com.hp.ov.activator.mwfm.engine.module.tmpc.TMPCModule</Class-Name>
   <Param name="database_module"   value="db"/>
</Module>
```

2.3.1.4 SmartBean Search Module

This module must be configured in order to access cached bean information from workflows, including the ones shipped with the MSA solution. This configuration is done in the *mwfm.xml* file.

```
<Module>
            <Name>SmartBeanSearch</Name>
            <Class-Name>com.hp.ov.activator.mwfm.engine.module.sbsm.SmartBeanSearchModule</Class-Name>
            <Param name="database_module" value="db"/>
</Module>
```

This module recognizes the following configuration parameters:

Parameter	Required	Description	Reconfigurable	Default
database_module	Yes	Database module used in order to load the configuration from database.	Yes	None

2.3.1.5 CORBA Module

To enable the CORBA module is necessary declare it in the mwfm.xml file.

```
<Module>
<Name>corba_module</Name>
<Class-Name>com.hp.ov.activator.mwfm.engine.module.corba.CorbaModule</Class-Name>
<Param name="dynamic_loading_folder" value="$ACTIVATOR_ETC/config /corba/descriptors" />
<Param name="pool_systems" value="$ACTIVATOR_ETC/config /corba/ems-pool.xml" />
<Param name="max_threads" value="10"/>
<Param name="min_threads" value="2"/>
<Param name="retry_count" value="1"/>
<Param name="database_module" value="db"/>
<Param name="technology_adapter" value="IOR"/>
<Param name="retry_interval" value="20"/>
<Param name="connection_timeout" value="3600000"/>
<Param name="timeout" value="30000"/>
</Module>
```

It is also necessary to configure $JBOSS_HOME/standalone/deployments/hpsa.ear/META-INF/jboss-deployment-structure.xml.

```
<dependencies>
        .....
        <module name="org.omg.api" services="export" />
        <module name="org.jacorb" services="export" />
</dependencies>
```

2.3.2 SOSA Configuration

2.3.2.1 NGWS protocol adapter

In order to receive requests via web service interface it is necessary to configure the NGWS protocol adapter in SOSA. Add the following block in configuration file $SOSA_CONF/sosa_conf.xml, inside the <ProtocolAdapters> element.

```
<ProtocolAdapter className="com.hp.sosa.modules.sosamodule.protocoladapters.ngws.NGWSProtocolAdapter"
name="NGWS_PA">
        <Parameter name="ngws.host" value="127.0.0.1"/>
        <Parameter name="ngws.port" value="8071"/>
        <Parameter name="ngws.min.threads" value="2"/>
        <Parameter name="ngws.max.threads" value="10"/>
        <Parameter name="ngws.path" value="ngws"/>
</ProtocolAdapter>
```

This protocol adapter must be started (users can start/pause protocol adapters using web interface of Solution Container. Besides, the SOSA queues can be blocked after installation, and it´s necessary unlock them to process requests).

Note: to make this webservice accessible from any other server, you should configure the parameter ngws.host with the value 0.0.0.0. Otherwise is going to be accessible only from the local server.

If you have just performed a new installation, you may have to configure the username and password to connect to MWFM. This configuration must be done in the same file, in MWFM Service Action Executor section.

```
<ServiceActionExecutor name="MWFM_SA_EXECUTOR"
className="com.hp.sosa.modules.sosamodule.executors.mwfm.MwfmServiceActionExecutor" max_parallelism="0">
        <Parameter name="host" value="127.0.0.1"/>
        <Parameter name="port" value="2000"/>
        <Parameter name="user" value="username"/>
        <Parameter name="password" value="password"/>
        <Parameter name="async_interval" value="60" />
        <Parameter name="launch_retries" value="1" />
        <Parameter name="copy_cp_to_output" value="false" />
        <Parameter name="timeout" value="90000" />
        <Parameter name="timeout_interval" value="30000" />
</ServiceActionExecutor>
```

Please note that the password must be encrypted, this can be achieved with the crypt utility shipped with HPSA.

Finally the jetty.start parameter must be set. This parameter is located in $SOSA_CONF/sosa.xml file.

```
<Parameter name="jetty.start" value="true" />
```

2.3.2.2 Bulk protocol adapter

In order to use the bulk activations feature it is necessary to configure the bulk protocol adapter in SOSA. Add the following block in configuration file $SOSA_CONF/sosa_conf.xml, inside the <ProtocolAdapters> element.

```
<ProtocolAdapter className="com.hp.ov.activator.mobile.sosa.protocoladapters.bulk.BulkProtocolAdapter" name="BULK_PA">

   <Parameter name="db.pool.name" value="db_sosa_catalog"/>

   <Parameter name="bulk.archive.directory" value="/path/to/bulk/archive"/>
   <Parameter name="bulk.request.directory" value="/path/to/bulk/requests"/>
   <Parameter name="bulk.result.directory" value="/path/to/bulk/results"/>

   <Parameter name="bulk.default.username" value="ftpuser"/>
   <Parameter name="bulk.mail.from" value="sosa@yourcompany.com"/>
   <Parameter name="bulk.mail.server.authenticate" value="true"/>
   <Parameter name="bulk.mail.server.hostname" value="mail.yourcompany.com"/>
   <Parameter name="bulk.mail.server.password" value="secret"/>
   <Parameter name="bulk.mail.server.port" value="25"/>
   <Parameter name="bulk.mail.server.secure" value="false"/>
   <Parameter name="bulk.mail.server.username" value="smtpuser"/>
   <Parameter name="bulk.mail.subject" value="Bulk Activations Report"/>

   <!- Fine tuning

   <Parameter name="bulk.disable.persistence.for.services" value="MCA,SDP,PTT"/>
   <Parameter name="bulk.enable.persistence.for.services" value="MNP,HLR"/>
   <Parameter name="bulk.max.attachment.size" value="20971520"/>
   <Parameter name="bulk.monitor.interval" value="5000"/>
   <Parameter name="bulk.override.persistence" value="false"/>
   <Parameter name="bulk.so.timeout" value="60000"/>
         ->

</ProtocolAdapter>
```

This protocol adapter must be started (users can start/pause protocol adapters using web interface of Solution Container).

2.3.3 Report Module Datasource Configuration

In order to use the Report Module it is necessary to take into account some configuration issues that are explained in this section.

Report Module can be configured to work with:

- The EP database. In the first case, it is possible:
  - to use the uiDB datasource that is already configured for SC applications, or
  - to create a new datasource for any of the following reasons:
    - To use different connection parameters
    - To ensure a certain number of connections
- A different database, meaning that it will be necessary to configure a new datasource.

For more information about JBoss datasource configuration:
https://community.jboss.org/wiki/DataSourceConfigurationInAS7#Defining_the_DataSource_itself

In any case, a datasource alias must be added in order to specify what datasource RM will use. Datasource alias configuration is defined in the

$JBOSS_HOME/standalone/deployments/hpsa.ear/ep.war/WEB-INF/alias.xml file.

The following examples show what to add to the *alias.xml* file in the cases previously described:

1. If uiDB is used:

```
<alias>
  <datasource-name>hpsa/jdbc/uiDB</datasource-name>
  <datasource-alias>reportmodule</datasource-alias>
</alias>
```

2. Otherwise, if an external datasource (e.g. rmds) has been created:

```
<alias>
  <datasource-name>hpsa/jdbc/rmds</datasource-name>
  <datasource-alias>reportmodule</datasource-alias>
</alias>
```

2.4 Uninstall MSA

In order to undeploy the MSA solution, follow the steps below:

1. Ensure that HPSA is not running, otherwise stop it now. For more information on stopping and starting the HPSA service, refer to section "**Starting and Stopping Service Activator**" in the *HPSA Installation Guide*.
2. Run the deployment manager from the $ACTIVATOR_OPT/bin directory.
3. Click on System Database Connection and enter database credentials, then click OK.
4. Click on Local Deployment.
5. Click on Undeploy Local Solution.
6. Select the MSA Solution.
7. Click on Undeploy solution.

2.5 Upgrade MSA

In order to upgrade the MSA solution after upgrade HPSA 6.x to HPSA 7.0.0 follow the steps below:

1. Ensure that HPSA is not running, otherwise stop it now. For more information on stopping and starting the HPSA service, refer to section "**Starting and Stopping Service Activator**" in the *HPSA Installation Guide*.
2. Run the deployment manager from the $ACTIVATOR_OPT/bin directory.
3. Click on System Database Connection and enter database credentials, then click OK.
4. Click on Local Deployment.
5. Click on Undeploy Local Solution.
6. Select the MSA Solution.
7. Check Do not undeploy SQL.
8. Click on Undeploy solution.
9. Click on Delete Local Solution.
10. Copy the new (release for 7.0.0) provided MSA.zip file into the $ACTIVATOR_OPT/SolutionPacks directory.
11. Click on Import Solution.
12. Select from zip/tar file and browse for SolutionPacks/MSA.zip.
13. Click on Import.
14. Click on Deploy Local Solution.
15. Select the MSA Solution.
16. Select the appropriate deployment file depending on the version, the database and the operating system (see the table below).

Deployment file	HPSA version	Database	Operating System	Test
deploy_upgrade_6_x_to_7_0_0.xml	7.0.0	Any	Windows	Any
deployUnix_upgrade_6_x_to_7_0_0.xml	7.0.0	Any	Unix	Any

17. Check Do not deploy SQL.
18. Click on Deploy solution.

Upgrade of the solution is now complete but additional configuration steps are required. Please review the Configuration section for more information in order to ensure that your configuration is aligned with the current environment.

3 Customizing the solution

In this section are described a set of use cases that should be used by a System integrator to get familiarize with the solution and understand the principles need to use and customize this solution.

3.1 Use Case 1: Add a new service

Use case description: *"The customer needs to activate a new service (Voice Mail) in his solution. The target manufacturer where the service needs to be activated is ZTE."*

3.1.1 Use case analysis

Note: For this use case it supposed that the customer only want to have the operation Create Subscriber on the voice mail platform.

First step to do the analysis, is get the parameters that the customer want use to invoke the create subscriber operation. For this case it supposed that only is going to define the phone number and a mail address of user.

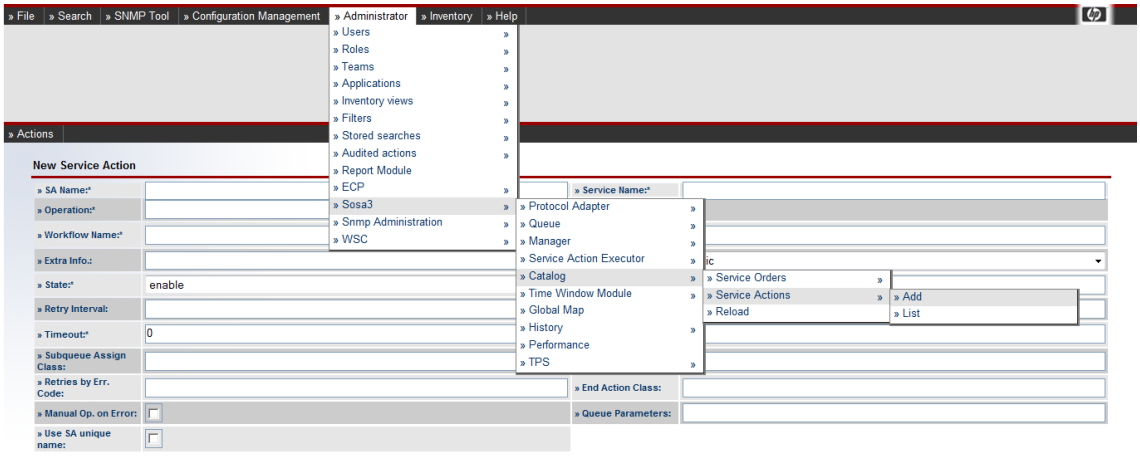
Next step of the analysis is study the target interface, to check the operation supported by the target and the parameters needed for each operation.

The required steps to develop this service in the application are:

1. Define the northbound interface
2. Create the target adaptor to establish the connection with target
3. Develop the activation workflow
4. Define the error management

3.1.2 Step 1: Define the Northbound interface

The northbound interface definition must be done on the SOSA web interface. The first step is defining the name of the service action (SA) in the catalog. For this case the name is VOICEMAIL-MOBILE-CREATE. The next picture shows how to create a new service action in the SOSA catalog.



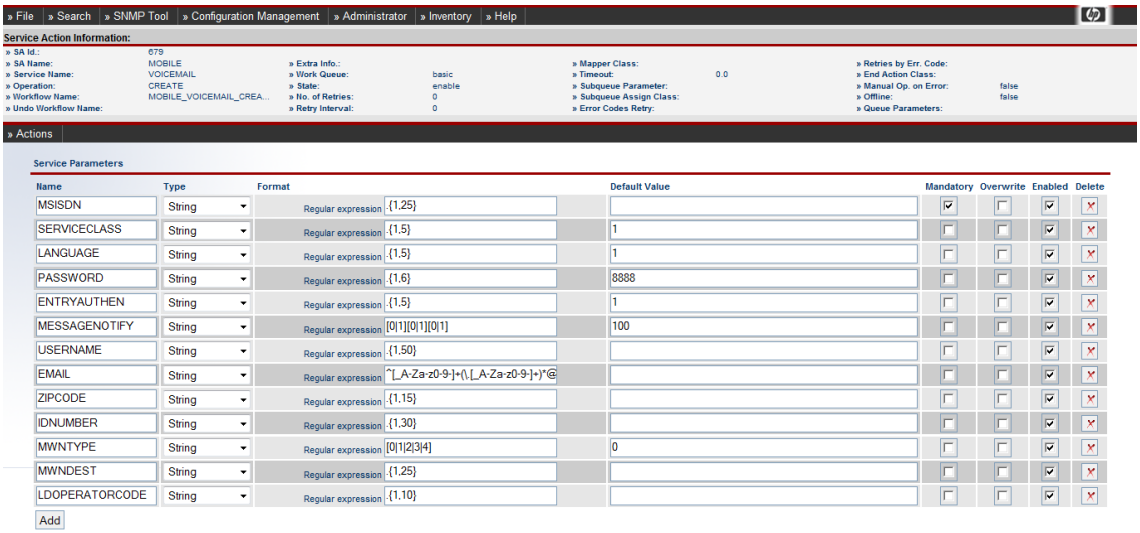
Once the SA has been create is necessary define the parameters associated with the create subscriber operation.

Is highly recommendable defined in the northbound interface all the possible parameters that target support, is this way the service definition can be manage easily and on the runtime.

From the target specification it can be obtain the list of parameters that are configurable for the create subscriber operation. This is the list of parameters for the create subscriber operation as well as it format.

Name	Type	Format	Mandatory	Default Value
MSISDN	String	.{1,25}	Yes	
SERVICECLASS	String	.{1, 5}	No	1
LANGUAGE	String	.{1, 5}	No	1
PASSWORD	String	.{1,6}	No	8888
ENTRYAUTHEN	String	.{1, 5}	No	1
MESSAGENOTIFY	String	[0|1][0|1][0|1]	No	100
USERNAME	String	.{1, 50}	No	
EMAIL	String	^[_A-Za-z0-9-]+(\.[_A-Za-z0-9-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$	Yes	
ZIPCODE	String	.{1, 15}	No	
IDNUMBER	String	.{1, 30}	No	
MWNTYPE	String	[0|1|2|3|4]	No	0
MWNDEST	String	.{1, 25}	No	
LDOPERATORCODE	String	.{1, 10}	No	

This parameter must to be defined as service parameters in SOSA catalog. See an example in the next picture.



For each parameter it can be define a regular expression to validate the parameter value. If the parameter does not match with this regular expression an error will be returned in the northbound invocation.

Additionally it can be define a default value for each parameters, in this use case it have been defined some default values for some parameters.

3.1.3 Step 2: Create the target adaptor

The selected service need to be activated via webservice. For this reason the target adaptor must be done using the Webservice Connectivity Module (WSC). In the section 6 of this guide it can be found more information about this module.

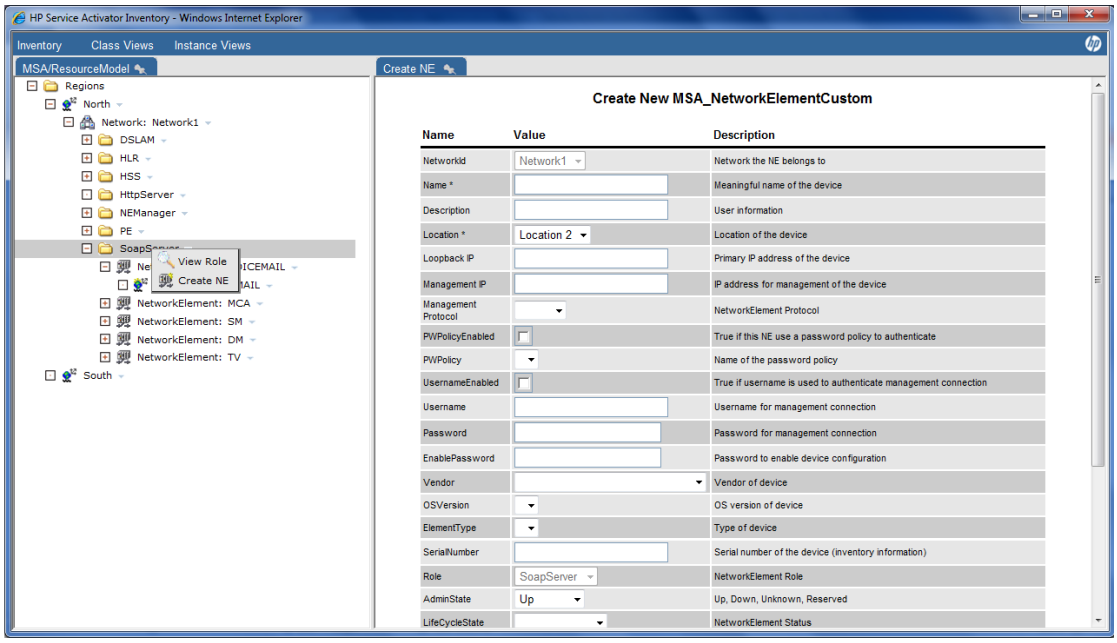
To create the target adaptor is necessary to follow the next steps:

1. Create the new webservice (VoiceMail) using the user interface.
2. Import the webservice wsdl (See the wsdl in the appendix)
3. Generate the descriptor using the user interface
4. Edit the generated descriptor and set an alias for the fields MSISDN and EMAIL
5. Activate the service
6. Reload the configuration from the HPSA web interface to load the new webservice into the module

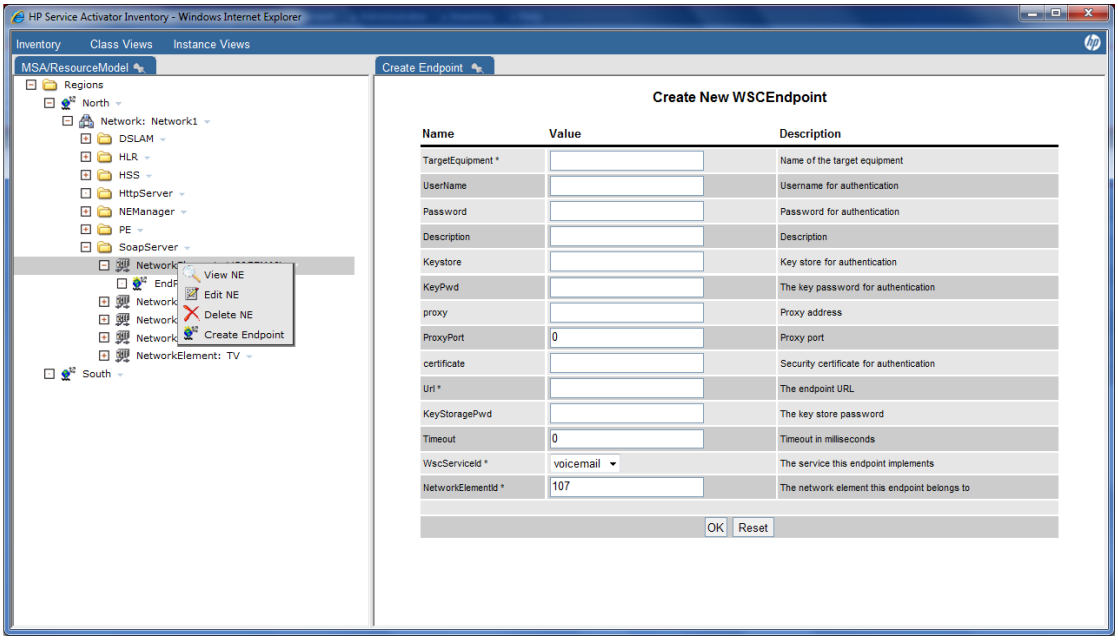
After these step the service is ready to be used from the activation workflow.

3.1.4 Step 3: Create a new endpoint

Go to the inventory tree view to create a new target under the SoapServer branch, and create a new target:



Once the target has been created, right click on the target name and create a new endpoint:



Those are the field that the user must fill to create an endpoint:

- TargetTarget, unique name to identify the endpoint, set VOICEMAIL as target target
- UserName: Username for authentication, this field is not mandatory leave it empty.
- Password: Password for authentication, this field is not mandatory leave it empty.
- Description: Description set by the user, this field is not mandatory leave it empty.
- Keystore: Key store for authentication, this field is not mandatory leave it empty.
- KeyPwd : The key password for authentication , this field is not mandatory leave it empty.
- Proxy: Proxy address, this field is not mandatory leave it empty.
- ProxyPort: Proxy port, this field is not mandatory leave it empty.
- Certificate: SSL Certificate, this field is not mandatory leave it empty.
- Url: Url where the target server is listening, set http://localhost:8088/mockIvmsbossHttpBinding for this field.
- KeyStoragePwd: KeyStore password, this field is not mandatory leave it empty.
- Timeout: Timeout for the request in milliseconds, this field is not mandatory leave it empty.
- WscServiceId : Relation with the service used by this endpoint, select voicelmail.
- NetworkElementId: Relation with the target, this field is read only.

3.1.5 Step 4: Develop the activation workflow

To develop the activation workflow you should use the Workflow Designer Application, this application is included in the HP Service Activator installer.

It is highly recommended see the Workflow.pdf document (included in the HPSA core documentation) to get familiarized with the workflow development.

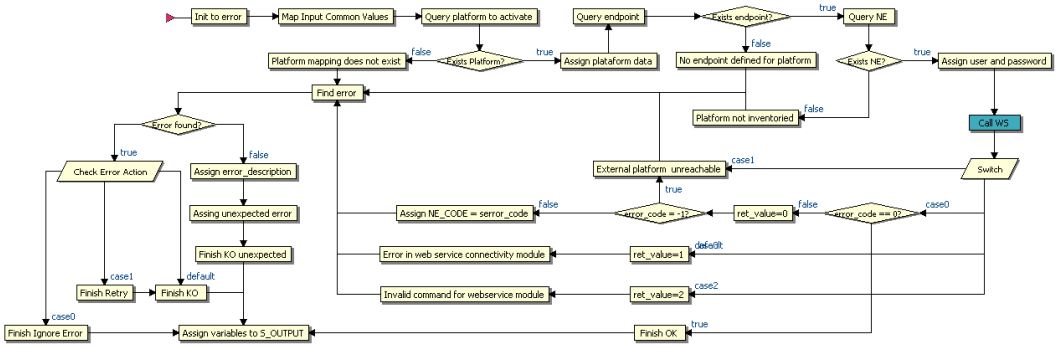
The main steps for this workflow are:

- Get the MSISDN value from the S_INPUT hashmap, this hashmap contains all the parameters defined for this service in the Service Catalog. The keys of this hashmap are the parameter names and the values of the hashmap are the parameter values.
- Query the platform where the service will be activate, this query should be done using the node com.hp.ov.activator.mwfm.component.sbsm.SBSGetBeanWithPatternNode. This node searches the information in the cache to avoid access to the database.
- Invoke the Webservice Connectivity Module using the node com.hp.ov.activator.mwfm.component.wsc.WSCRequestNode, this node receive as an input the S_INPUT hashmap. These are the parameter declared for this invocation:

```
<Name>Call WS</Name>
    <Action>
      <Class-Name>com.hp.ov.activator.mwfm.component.wsc.WSCRequestNode</Class-Name>
      <Param name="exec_time" value="constant:invocation_time"/>
      <Param name="input_attr_map" value="S_INPUT"/>
      <Param name="method_name" value="constant:createUser"/>
      <Param name="module" value="wsc_module_name"/>
      <Param name="request_trace" value="COMMAND_SENT"/>
      <Param name="response_trace" value="NE_OUTPUT"/>
      <Param name="service_name" value="constant:voicemail"/>
      <Param name="target_target" value="beanEndpoint.Targettarget"/>
      <Param name="throw_excep" value="false"/>
      <Param name="input_attr_name0" value="operID"/>
      <Param name="input_attr_value0" value="Targetuser"/>
      <Param name="output_attr_name0" value="response"/>
      <Param name="output_attr_value0" value="error_code"/>
      <Param name="input_attr_name1" value="operPwd"/>
      <Param name="input_attr_value1" value="Targetpassword"/>
    </Action>
    <Next-Node>Switch</Next-Node>
```

- Evaluate the response of this node.
- Fill the response hashmap S_OUTPUT, with the values returned by the target.
- Control the potential errors of the workflow and invoke the error management module in error case, to translate the error code.

This picture represents the workflow that need to be developed:



Once that workflow have been developed should be deployed using the Workflow designer, after the deployment the user need to reload the workflow using the HPSA administrator GUI.

The source code of this workflow can be found in the MSA installer.

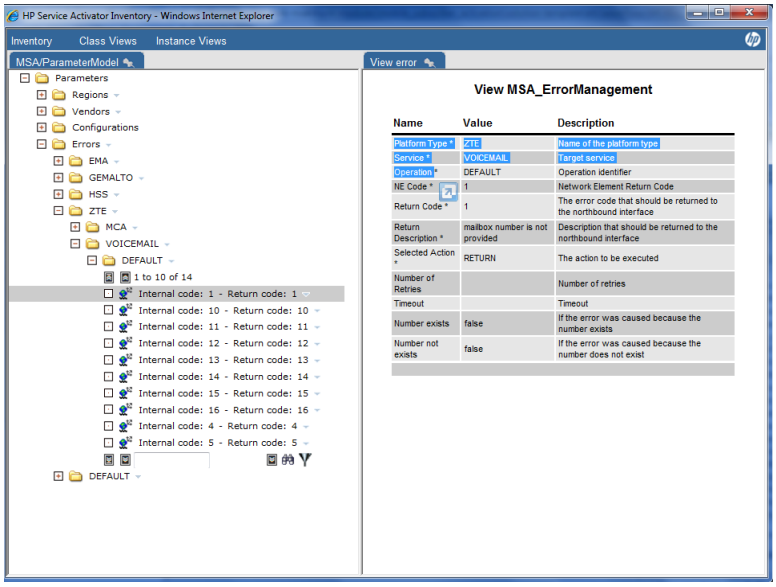
3.1.6 Step 5: Define the error management

There are two kinds of errors that can be managed for a service:

- The workflow errors, these are the error managed in the workflow nodes, the code for these errors are defined by the developer and are hardcoded in the workflow.
- The errors returned the de targets, these are the error that the platform returns, and the codes for these errors are defined for the manufacturer and should be described in the platform documentation.

Both of these kinds of errors can be managed for the Error Management module, basically this module do a translation of an error code in another error code and description defined for the user. The error codes are defined in the database using an inventory view. Each error is related with a Platform Type, a Service and an Operation. In case that is not needed to discriminate for any of these fields the user should set the value to DEFAULT.

Next picture shows the inventory view for the errors defined for the voicemail service:



3.1.7 Step 6: Test the service

After follow of these steps the service is ready to be tested. If there is a ZTE platform available to test the service this could be used, if the platform is not available it can be used any software to simulated the target like SoapUi or similar.

Also could be used a software like SoapUI to send the input request. This is an example of request that could be used on Soap UI to test the voice mail service.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws"
xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
  <soapenv:Header/>
  <soapenv:Body>
    <ngws:startDynamicOrderSync>
      <dyn:serviceRequest>
        <dyn:services>
          <dyn:service>
            <dyn:name>VOICEMAIL</dyn:name>
            <dyn:type>MOBILE</dyn:type>
            <dyn:action>CREATE</dyn:action>
            <dyn:characteristics>
              <dyn:characteristic>
                <dyn:name>MSISDN</dyn:name>
                <dyn:value>1</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>SERVICECLASS</dyn:name>
                <dyn:value>2</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>LANGUAGE</dyn:name>
                <dyn:value>3</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>PASSWORD</dyn:name>
                <dyn:value>4</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>ENTRYAUTHEN</dyn:name>
                <dyn:value>5</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>MESSAGENOTIFY</dyn:name>
                <dyn:value>101</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>USERNAME</dyn:name>
                <dyn:value>7</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>EMAIL</dyn:name>
                <dyn:value>foo@bar.baz</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>ZIPCODE</dyn:name>
                <dyn:value>9</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>IDNUMBER</dyn:name>
                <dyn:value>10</dyn:value>
```

```
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>MWNTYPE</dyn:name>
                  <dyn:value>3</dyn:value>
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>MWNDEST</dyn:name>
                  <dyn:value>12</dyn:value>
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>LDOPERATORCODE</dyn:name>
                  <dyn:value>13</dyn:value>
                </dyn:characteristic>
              </dyn:characteristics>
            </dyn:service>
          </dyn:services>
        </dyn:serviceRequest>
        <ngws:user>foobar</ngws:user>
      </ngws:startDynamicOrderSync>
    </soapenv:Body>
  </soapenv:Envelope>
```

3.2 Use Case 2: Customize the northbound interface

Use case description: *"The customer needs to fix a default value for the parameter* LDOPERATORCODE in the service Voice Mail.*"*

3.2.1 Use case analysis

The first step for the analysis is defined the value that the customer want to set for the parameter LDOPERATORCODE, for this example we supposed that the value is 5.

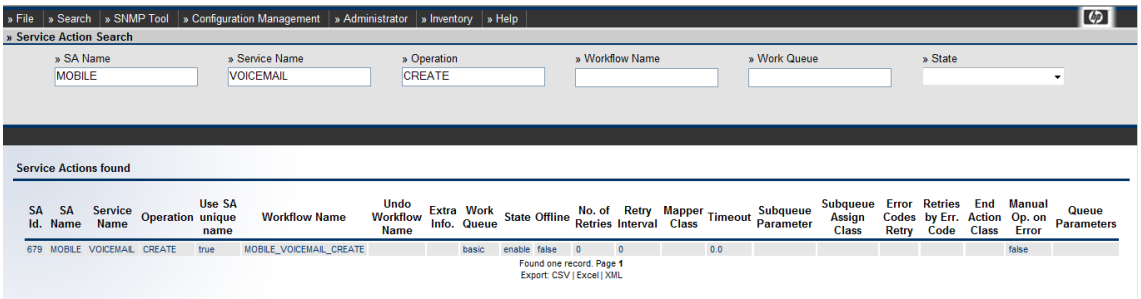
Next step is determine if this parameter will be received in north bound interface and if this parameter should be overwrite with value received from the northbound interface or not. For this example we supposed that the value could not be overwrite, this mean that for this service the value that is going to be send to the target will be 5.

The required steps to implement this customization are:

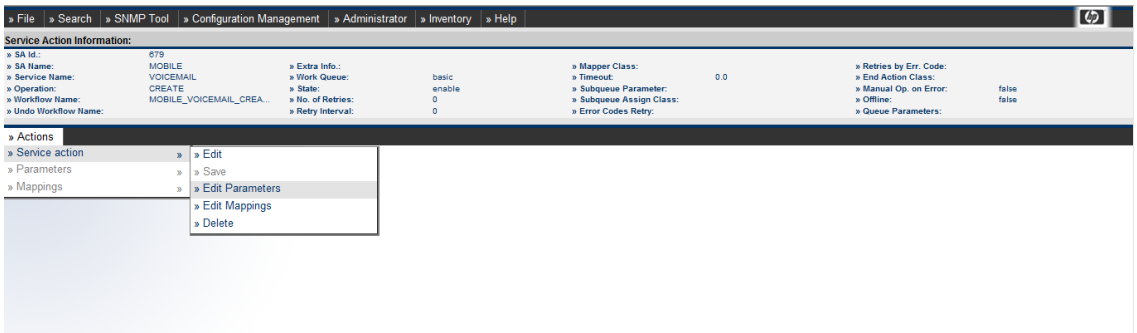
1. Edit the service parameters
2. Test the service

3.2.2 Step 1: Edit the service parameters

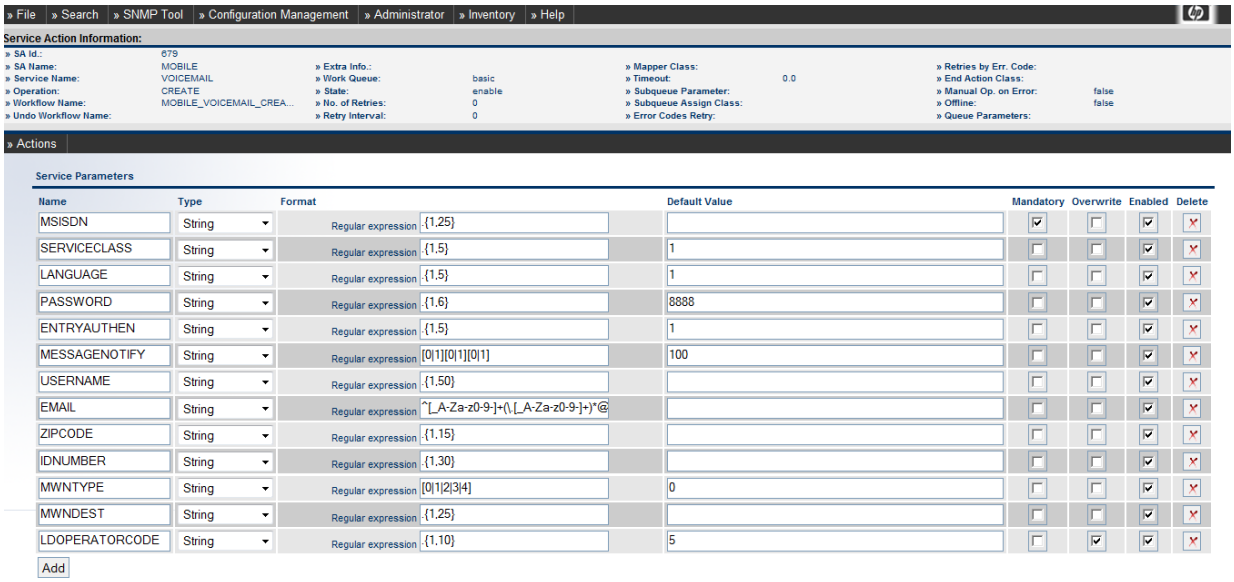
To edit the service parameter for the service VoiceMail Create is necessary go to the Service Catalog and look for the service Mobile VoiceMail Create.



Then in necessary open the parameter definition for this service:

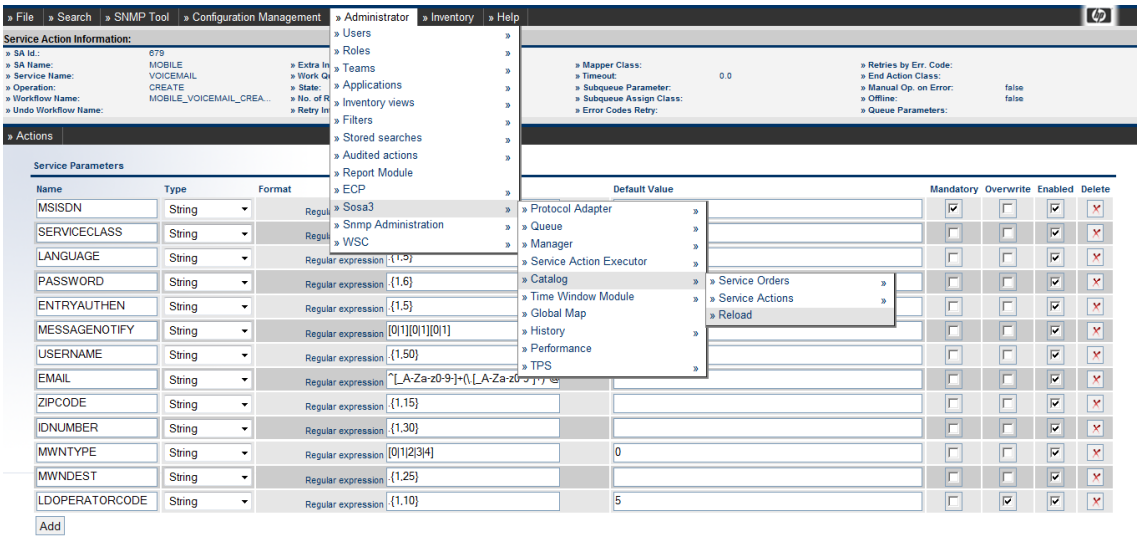


Now we are going to set the value 5 for the field LDOPERATORCODE and check the column overwrite to indicate that the default value takes precedence over the request value. Be sure that the column enabled is check to indicate that the field is enabled and the mandatory column is unchecked to indicate that this field is not mandatory in the northbound interface.



After edit the LDOPERATORCODE parameters in necessary save the changes this operation is available under the Action menu.

The next step is to reload the SOSA catalog to apply these changes in the catalog. The changes will not be applied until the catalog was reloaded. Next picture show how to reload the SOSA catalog.



3.2.3 Step 2: Test the service

To test the service, is necessary send the next set of requests and do the make the appropriate checks:

1. Send a request including the parameter LDOPERATORCODE with the value 8 and check that the value sent to the target is 5.
2. Send a request without include the parameter LDOOPERATORCODE and check that the value sent to the target is also 5.

For this example could be use the same request that we used in the use case 1 of this document.

3.3 Use Case 3: Customize the command template

Note: The previous examples are for service whose activation is based on webservice, this example is based on a service whose activation is base in CLI (Command Line Interface).

Use case description: *"The customer wants to add a new parameter (IDTEST) in the command to activate a subscription in the HLR."*

We supposed that this new parameter IDTEST is supported for the target, and that the activation is going to be performed over an EMA platform.

3.3.1 Use case analysis

First step is study how we want to manage this parameter:

- Is mandatory? It supposed that the parameter is mandatory for this service.
- Which are the possible values supported for this parameters? It supposed that the possible values are 0 or 1.
- Does the customer want to fix a default valued for the parameter? It supposed that the customer want to fix the value 1 for it.
- Will this parameter be received in the northbound interface? It supposed that this parameter is not going to be received in the northbound interface.

As the parameter is not going to be received in the northbound interface, it must be declared in the command template.

Next step is to establish the command syntax, to do that is necessary to get the information from the target manufacturer. For this example it supposed that the syntax of the command including the new parameter is the next one:

CREATE:HLRSUB:MSISDN,<value>: IMSI,<value>:**IDTEST,<value>:** PROFILE,<value>: TRANSID,<value>: CAW,<value>: CAT,<value>: STYPE,<value>: CSP,<value>: OBI,<value>: OBO,<value>: OSB1,<value>: OSB2,<value>: OSB3,<value>: OSB4,<value>: OBR,<value>: NAM,<value>: PDPCP,<value>: RSA,<value>: SCHAR,<value>: HOLD,<value>: MPTY,<value>: AMSISDN,<value>: CLIP,<value>: CLIR,<value>: COLR,<value>: COLP,<value>: SOCLIR,<value>: SOCLIP,<value>: SOCOLP,<value>: CFU,<value>: DCF,<value>: CFNRY,<value>: CFNRC,<value>: CFB,<value>: DBSG,<value>: OFA,<value>: BAIC,<value>: BAOC,<value>: PRBT,<value>: BEARER_SERVICES,<value>: TELESERVICES;

The last step is to identify the manufacturer, element type and OS version of the target where the customer wants to do the modification. It supposed that the manufacturer is Ericsson, the element type is APG40C and the OS version is R13

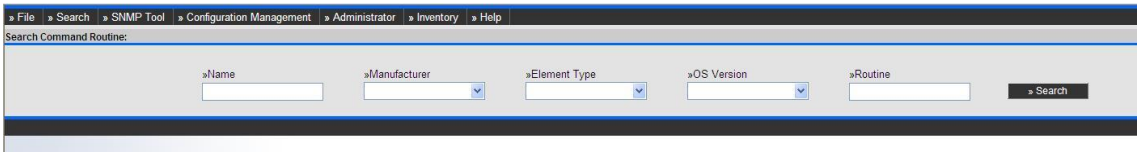
The required steps to implement this customization are:

1. Edit the command template for the service MOBILE HLR CREATE on EMA.
2. Test the service

3.3.2 Step 1: Edit the command template

Go to the command template application in the solution container web interface and search the template that you want modify. To access the Search (List) page, go to "Administrator -> ECP -> Activation Commands Template -> Command Routine -> **List**"

The search page will be displayed, empty for now. At the top-most part you will find the fields you can use to filter your search:

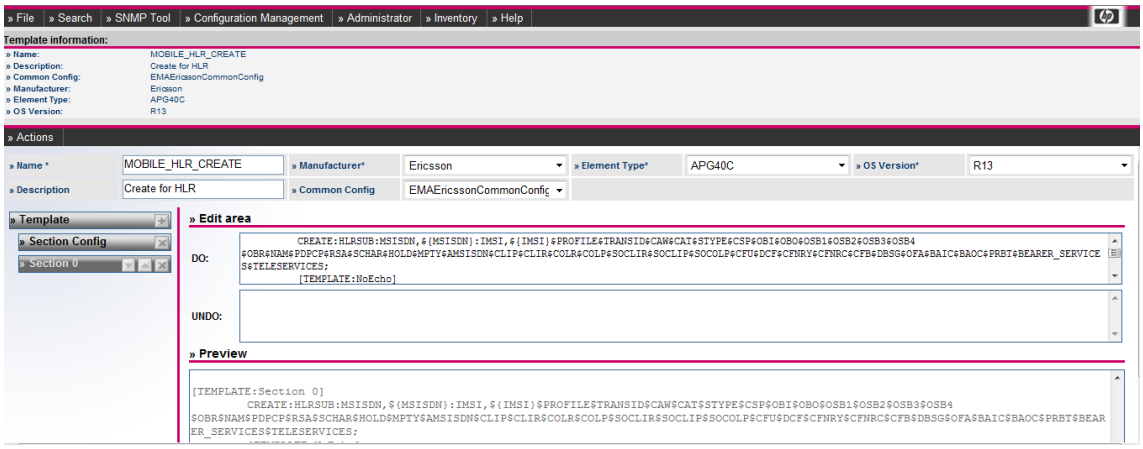


Select the manufacturer Ericsson, the element type APG40C and OS version R13 and click the button search.

Once you hit the "Search" button, the results will be listed. Now click on the MOBILE_HLR_CREATE to go to the Edition page.

After click on this element you can see a preview of the command template. To ed

it the template click on Edit option under Actions menu. The next picture shows the edition screen.



Select the Section 0 in the left tree and edit the command, this is the new command that need to be appears in the section 0:

CREATE:HLRSUB:MSISDN,${MSISDN}:IMSI,${IMSI}:**IDTEST,1** $PROFILE$TRANSIDCAWCAT$STYPE$CSPOBIOBO$OSB1$OSB2$OSB3$OSB4OBRNAM$PDPCP$RSA$SCHAR$HOLD$MPTY$AMSISDN$CLIP$CLIR$COLR$COLP$SOCLIR$SOCLIP$SOCOLP$CFUDCFCFNRY$CFNRC$CFB$DBSG$OFA$BAIC$BAOC$PRBT$BEARER_SERVICES$TELESERVICES;

[TEMPLATE:NoEcho]

[TEMPLATE:Pattern "RESP.*:([0-9]+);"]

[TEMPLATE:Variable "CODE"]

The difference with the previous command is that we have include "**IDTEST,1:** " that was the objective of this customization.

To apply the change in the template click Save option in the Action menu. This action will update the command template into the database.

For the change to take effect in the activation process go to the HPSA Administrator web and click on Reload configuration, this option in under the Self management menu.

For more information about the Command Template Module, see the section 5 in this document.

For more information about the command syntax see ECP.pdf document, this document is provided with the Extension Pack installation kit.

3.3.3 Step 2: Test the service

To test the service launch a request for the service MOBILE HLR CREATE, without the parameter IDTEST and verify that this parameter appears in the command sent to the target.

3.4 Use Case 4: Customize the error management

Use case description: *"The customer needs to change the error code and the description returned in the northbound interface for the error code 10 in the Voice Mail service."*

3.4.1 Use case analysis

The first step is to identify the error code and the description that the user wants to return in the northbound interface when the error returned by the VoiceMail platform is 10.

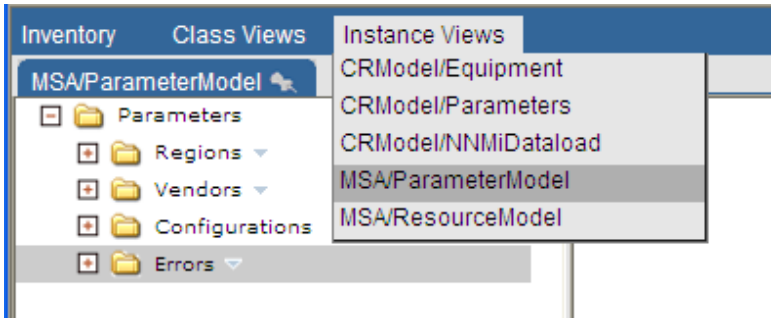
It supposed that the user want to receive and error code 1001 and the description should be "Internal error on the Voice Mail platform"

The required step to implement this customization is:

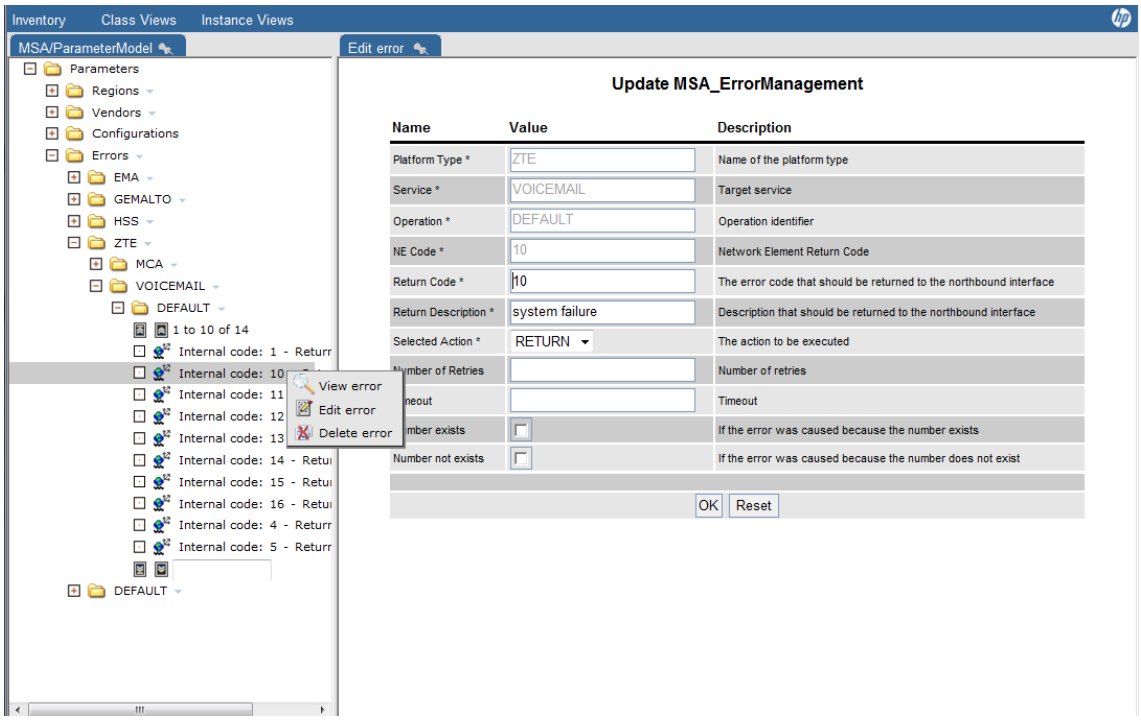
1. Edit the Error

3.4.2 Step 1: Edit the error

To edit the error go to *Inventory* → *Open* and select *Class Views* → *MSA/Instance Views.*



Now go to the Error 10 on the VoiceMail branch, select the error 10 and right click on to get the Edit option. Click on Edit option to see the Edit form.



Now update the field Return Code with the value 1001 and the Return description with the value "Internal error on the Voice Mail platform", and click on OK button to save the information into the database.

Note that after any change in the error management HPSA configuration must be reloaded to get that changes take effect.

After reload the HPSA configuration the error code and the description returned in the northbound interface will be the new ones.

3.5 Use Case 5: Create a new report

Use case description: *"The customer needs to have a new report with the next requirements:*

Input: MSISDN, IMSI, ACTION and Start Date.

Output: MSISDN, IMSI, TYPE, ACTION, S_CODE, S_DESCRIPTION, SOSA_CODE, SOSA_DESCRIPTION, NE_CODE, NE_DESCRIPTION, COMMAND_RECEIVED, COMMANDS_SENT, NETWORKELEMENT, CREATION_TIME, NE_ACT_TOTAL_TIME, PROCESSING_TIME, USERNAME, WF_INSTANCE, NE_OUTPUT."

3.5.1 Use case analysis

Note: for this example is necessary to have some knowledge about SQL and the database model.

The analysis of this case consists only in to identify where is the information in the database, for this example all the information is the in the table history_hbm_sa. This table is the one used in MSA to store the information for all the activations.

The required steps to implement this customization are:

1. Create the SQL Query
2. Edit the report module configuration file
3. Test the new report

3.5.2 Step 1: Create the SQL query

Write the sql in any sql editor in order to test it. For this example the sql that we are going to use is:

```
 SELECT
MSISDN, IMSI, TYPE, ACTION, S_CODE, S_DESCRIPTION, SOSA_CODE, SOSA_DESCRIPTION,
NE_CODE, NE_DESCRIPTION, COMMAND_RECEIVED, COMMANDS_SENT, NETWORKELEMENT,
CREATION_TIME, NE_ACT_TOTAL_TIME, PROCESSING_TIME, USERNAME, WF_INSTANCE,
NE_OUTPUT
 FROM
HISTORY_HBM_SA
 WHERE
   START_TIME > TO_TIMESTAMP ( ?,'DD-MM-YY HH24:MI:SS'))
    AND
   START_TIME < TO_TIMESTAMP (?,'DD-MM-YY HH24:MI:SS'))
    AND
```

upper(ACTION) = upper(?)

Set correct values for the inputs and test the query.

3.5.3 Step 2: Edit the report module configuration file

Edit the Report Module configuration file $JBOSS\standalone\deployments\hpsa.ear\ep.war\WEB-INF\classes\reportmodule\ReportModuleQueryDefinition.xml

Add in this file de new query using the format defined for the Report Module. The text does need to be added in the configuration file is:

```
<query>
        <name>Activations by item</name>
          <description>Search History Service Actions by item</description>
          <order>100</order>
          <maxlengthresult>1000</maxlengthresult>
          <sql>
            <![CDATA[
                                NE_ACT_TOTAL_TIME,
                                MSISDN,
                                IMSI,
                                TYPE,
                                ACTION,
                                S_CODE,
                                S_DESCRIPTION,
                                SOSA_CODE,
                                SOSA_DESCRIPTION,
                                NE_CODE,
                                NE_DESCRIPTION,
                                COMMAND_RECEIVED,
                                COMMANDS_SENT,
                                NETWORKELEMENT,
                                CREATION_TIME,
                                PROCESSING_TIME,
                                USERNAME,
                                WF_INSTANCE,
                                NE_OUTPUT
                    FROM
                      HISTORY_HBM_SA
                    WHERE
                          ('$From$' is null OR START_TIME > TO_TIMESTAMP ('$From$','DD-MM-YY HH24:MI:SS'))
                           AND
                          ('$To$' is null OR START_TIME < TO_TIMESTAMP ('$To$','DD-MM-YY HH24:MI:SS'))
                          AND
                          ('$OPERATION$' is null OR upper(ACTION) = upper('$OPERATION$'))
]]>
                  </sql>
                  <input>
                          <parameter mandatory="false">
                                  <name>OPERATION</name>
                                  <type>Select</type>
                                  <description>Operation</description>
                                  <defaultValue></defaultValue>
                                  <listOfValues>
                                          <value></value>
                                          <value>CREATE</value>
                                          <value>GET</value>
                                          <value>SET</value>
                                          <value>DELETE</value>
```

```
                    </listOfValues>
                </parameter>
                <parameter mandatory="false">
                    <name>From</name>
                    <type>datetime</type>
                    <description>From Date</description>
                    <defaultValue></defaultValue>
                </parameter>
                <parameter mandatory="false">
                    <name>To</name>
                    <type>datetime</type>
                    <description>To Date</description>
                    <defaultValue></defaultValue>
                </parameter>
            </input>
            <redirections>
                <redirect>
                    <jsp>table.jsp</jsp>
                    <type>Table</type>
                </redirect>
            </redirections>
        </query>
```

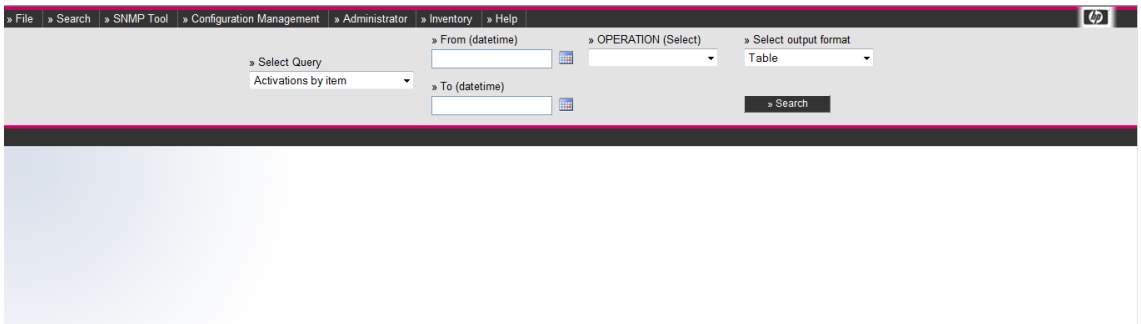
Save the file in the filesystem.

For more information about the Report Module see the section 7 in this document

3.5.4 Step 3: Test the new report

To test the new report, go to the Report Module application in the solution container "Administrator -> Report Module "

Now the new Report "Activation by Item" should be appears in the reports list



Execute the report clicking on the Search button and verify that the information that appears at the bottom table is the expected.

3.6 Use Case 6: Combine two services

Use case description: *"The customer wants to combine the activation of two services for a single service request*. The services to combine are VoiceMail and MCA (Missed Alert Calls) for the operation create*"*

3.6.1 Use case analysis

The analysis of this case consists on determine which are the parameters needed to activate both services, the parameters needed will be the summation of the parameter of both services. The next table represents the parameters needed to activate this new complex service.

Name	Type	Format	Mandatory	Default Value
MSISDN	String	.{1,25}	Yes	
SERVICECLASS	String	.{1, 5}	No	1
LANGUAGE	String	.{1, 5}	No	1
PASSWORD	String	.{1,6}	No	8888
ENTRYAUTHEN	String	.{1, 5}	No	1
MESSAGENOTIFY	String	[0|1][0|1][0|1]	No	100
USERNAME	String	.{1, 50}	No	
EMAIL	String	^[_A-Za-z0-9-]+(\.[_A-Za-z0-9-]+)*@[A-Za-z0-9]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,})$	Yes	
ZIPCODE	String	.{1, 15}	No	
IDNUMBER	String	.{1, 30}	No	
MWNTYPE	String	[0|1|2|3|4]	No	0
MWNDEST	String	.{1, 25}	No	
LDOPERATORCODE	String	.{1, 10}	No	
NOTIFYPHONE	String	.{1,25}	No	

Also is necessary to determine which will be the execution mode for of this new service request. It supposed that both services should be activate in serial, executing the Voice Mail first and the MCA in the second position, and in case of error the action to apply will be continue.

To finish this analysis we will decide the name of this new service, for this example we are going to use: VAS-MOBILE-CREATE.

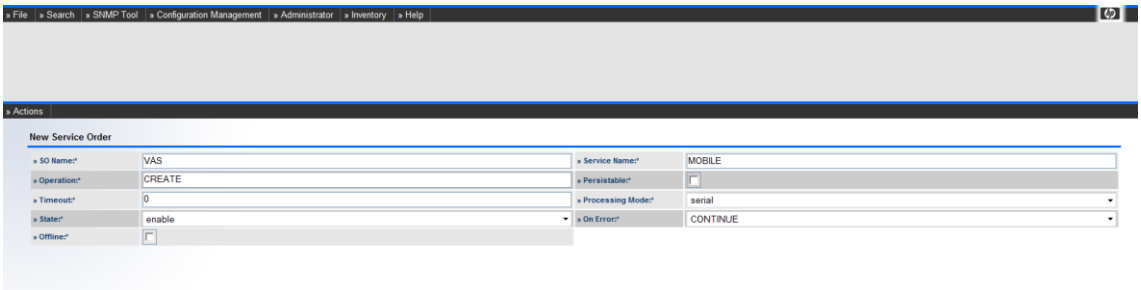
The required steps to implement this customization are:

1. Create the new Service Request in the service catalog
2. Test the service

Note: the addition of this new complex service will be on the fly and without affection to the current services.

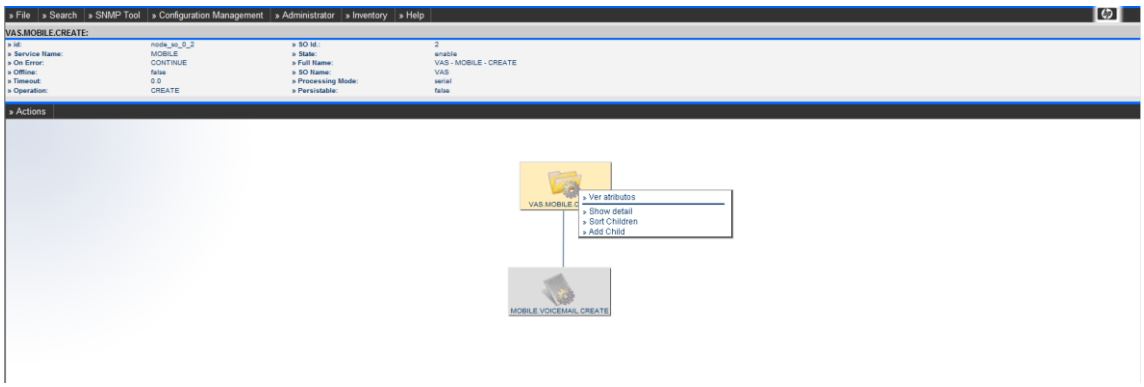
3.6.2 Step 1: Create the new service request in the service catalog

To create the new service request in the catalog is necessary to add a new service order in Sosa Catalog. To execute this action go to Administrator → Sosa3→ Catalog→Service Order→Add, and fill the fields as the next picture shows:



In this form we have configured the new service order with the options that we decided in the analysis of the case.

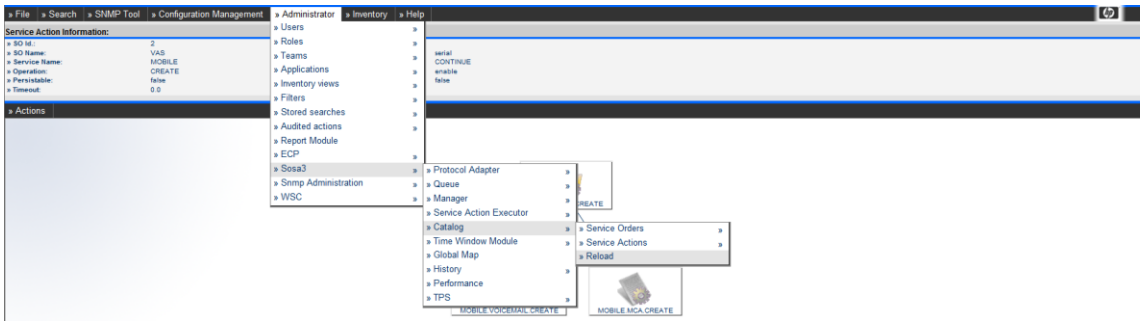
After save the new service order is necessary to add the services that will compose this service order. To do this task, right click on the service order that appears in the screen after save the service order and select the option Add Child.



After select the option Add Child a list will be appears with all the services that could be added to this service order.

Execute this step twice, one to add the service Voice Mail and another one to add the service MCA. Then click on Save operation under Actions menu. Next picture represent the how the new service order should look like after add the two services:

Once that service has been attached to the service order and the service order has been saved, to apply the change and have the new service order ready to be executed, it is necessary to reload the Sosa catalog.



3.6.3 Step 2: Test the service

To test the service order that we have created is necessary to create a new service request, supposing that we are using the default northbound interface; the request that we need to send to execute this new service is the next:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ngws="http://www.hp.com/sosa/protocoladapter/ngws" xmlns:dyn="http://www.hp.com/sosa/dynamicserviceorder">
  <soapenv:Header/>
  <soapenv:Body>
    <ngws:startDynamicOrderSync>
      <dyn:serviceRequest>
        <dyn:services>
          <dyn:service>
            <dyn:name>VOICEMAIL</dyn:name>
            <dyn:type>MOBILE</dyn:type>
            <dyn:action>CREATE</dyn:action>
            <dyn:characteristics>
              <dyn:characteristic>
                <dyn:name>MSISDN</dyn:name>
                <dyn:value>1</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>SERVICECLASS</dyn:name>
                <dyn:value>2</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>LANGUAGE</dyn:name>
                <dyn:value>3</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>PASSWORD</dyn:name>
                <dyn:value>4</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>ENTRYAUTHEN</dyn:name>
                <dyn:value>5</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>MESSAGENOTIFY</dyn:name>
                <dyn:value>101</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>USERNAME</dyn:name>
                <dyn:value>7</dyn:value>
              </dyn:characteristic>
              <dyn:characteristic>
                <dyn:name>EMAIL</dyn:name>
                <dyn:value>foo@bar.baz</dyn:value>
              </dyn:characteristic>
```

```
                <dyn:characteristic>
                  <dyn:name>ZIPCODE</dyn:name>
                  <dyn:value>9</dyn:value>
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>IDNUMBER</dyn:name>
                  <dyn:value>10</dyn:value>
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>MWNTYPE</dyn:name>
                  <dyn:value>3</dyn:value>
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>MWNDEST</dyn:name>
                  <dyn:value>12</dyn:value>
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>LDOPERATORCODE</dyn:name>
                  <dyn:value>13</dyn:value>
                </dyn:characteristic>
                <dyn:characteristic>
                  <dyn:name>NOTIFYPHONE</dyn:name>
                  <dyn:value>11122233344455</dyn:value>
                </dyn:characteristic>
              </dyn:characteristics>
            </dyn:service>
          </dyn:services>
        </dyn:serviceRequest>
        <ngws:user>foobar</ngws:user>
      </ngws:startDynamicOrderSync>
    </soapenv:Body>
  </soapenv:Envelope>
```

This request indicate that the service that should be executed in the one identified by VAS- MOBILE-CREATE, and the parameters to activated this service are the ones that we got in the use case analysis.

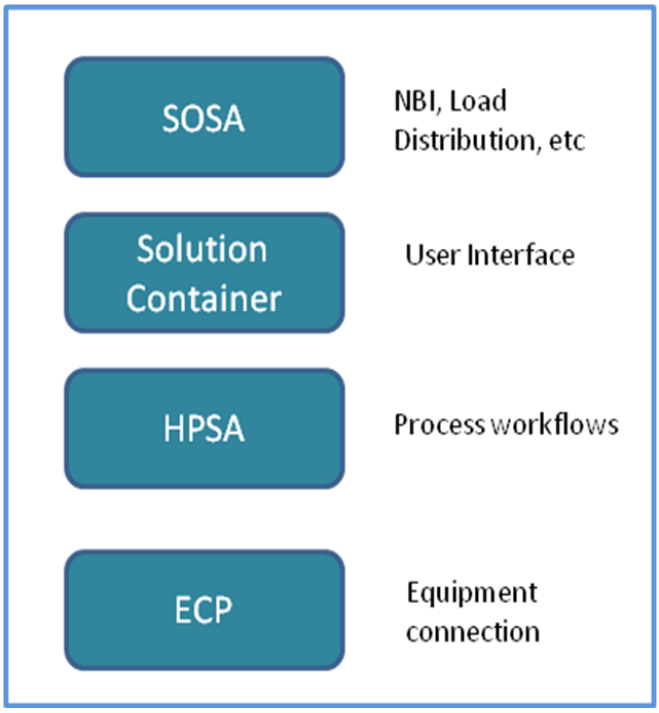
4 High performance architecture

Usually the Mobile Service Provider required a high performance in the activation process, due more of the mobiles services need to be activated instantly. Additionally the service provider need to manage a high volume of activation per second, this will depends on the Operator size.

MSA is designed to cover these both aspects of the mobile activation. The mains technical details of this product to fix these requirements are:

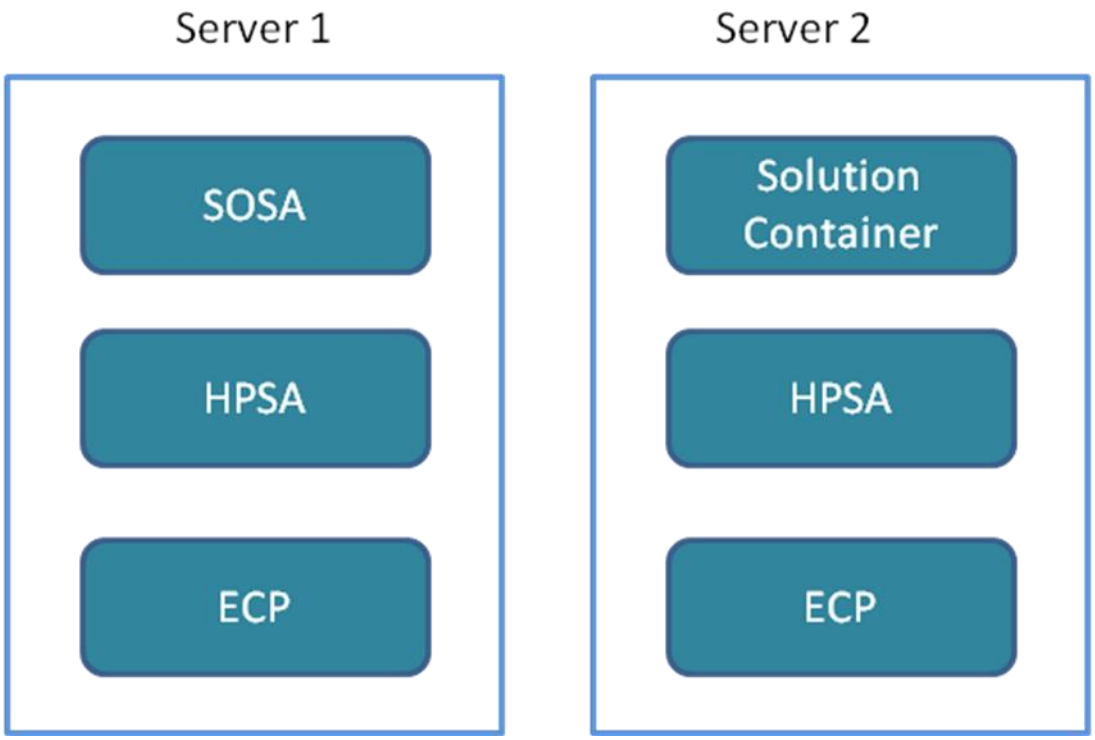
- The MSA workflows do not access to the database during the execution time, all the information needed by the workflow is query from the memory, using the smart bean search module.
- The interaction with the database is centralized in SOSA, in this way the number of connection with the database needed is lower and can be done in bulk. The interaction with the database is only to store the history information.
- For the command line activation, the connection with the target is doing using a connection pool, this characteristic avoid opening a new connection with the target for each activation request.
- For the webservice activation, the Webservice Connectivity Module provides an efficient mechanism to send webservice request to the target.
- In the northbound interface SOSA make an efficient load distribution using a queues system as well as controlling the avalanche.

Next picture shows the processes that need to run MSA:



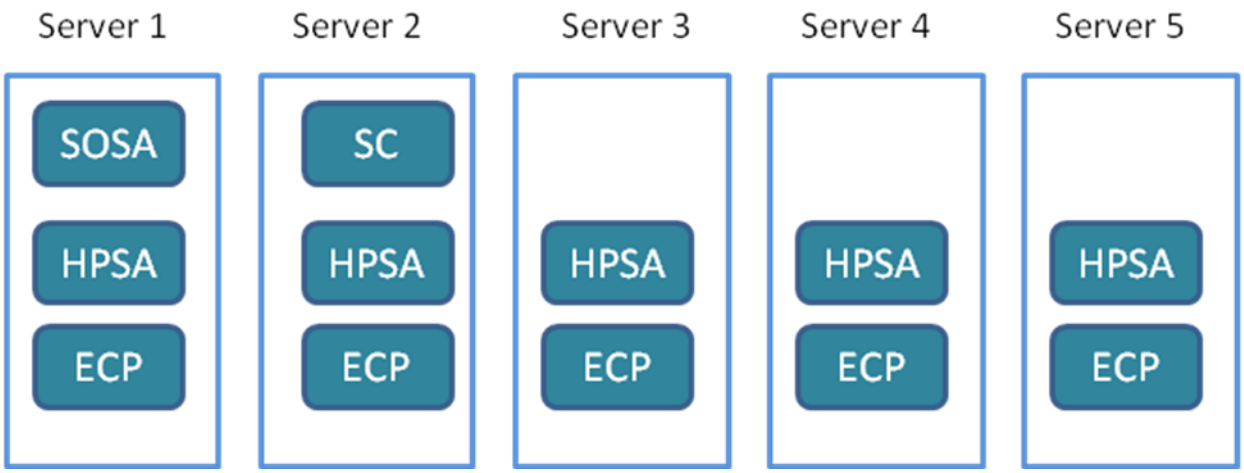
All this processes are java process that can run independently hence the process can run in the same or in a different physical server.

Next picture show the recommended process distribution using two physical servers:



The number of the servers depends on the customer requirements, this architecture can grow horizontally, and it means that the number of HPSA and ECP can be increasing to increase the performance, in a lineal way.

For a huge volume of activations, like 500 requests per second the number of servers should be next to 5, can be supposed that each HPSA+ ECP can support around 100 requests per second. This estimation is made base on the services offer by MSA out of the box. Next picture shows the recommended architecture for this use case:



Note: for all this cases, it supposed that the database runs in a different server.

5 Template management

MSA provides a solution to manage and cache ECP command templates.

A "Command Template" is a string which complies with a certain syntax through which an Operation is expressed, for the ECP Module to interpret and process it, usually with the purpose of automating a human interactive session on the Target System. The "Command Template" states the commands needed to perform the process (and usually to roll it back too), with specific information on every command, such as possible command outputs and their meaning (error, success) and the control flow which determines their execution order, among other things.

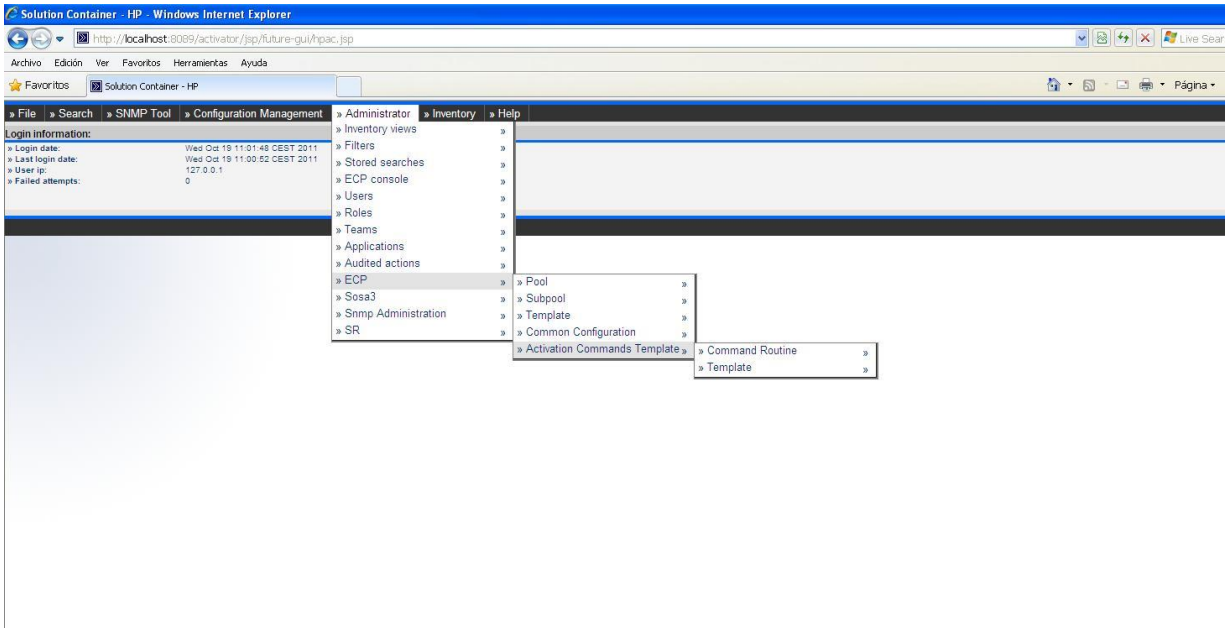
For more information about ECP and Command Templates see the document ECP.pdf, this document is available with the Extension Pack installation kit.

The main advantages provided by this module are:

- The templates are stored in the database, this feature provided a centralized management of the templates, and this is very helpful when MSA is installed on a cluster.
- The templates can be managed from a user interface integrated in the Solution Container.
- The template are related with the target OS version and Type, this feature allow managing different version of the target for the same service.
- This module provide a cache module and a node to query the template from this cache in the workflow avoiding to access to the database.
- A change in a template can be applied on the fly only and from the user interface.
- This module provide a mechanism to export/import a template in an xml file, this feature is very helpful to migrate a template from the development environment to the production environment.

5.1 Managing a template from the user interface

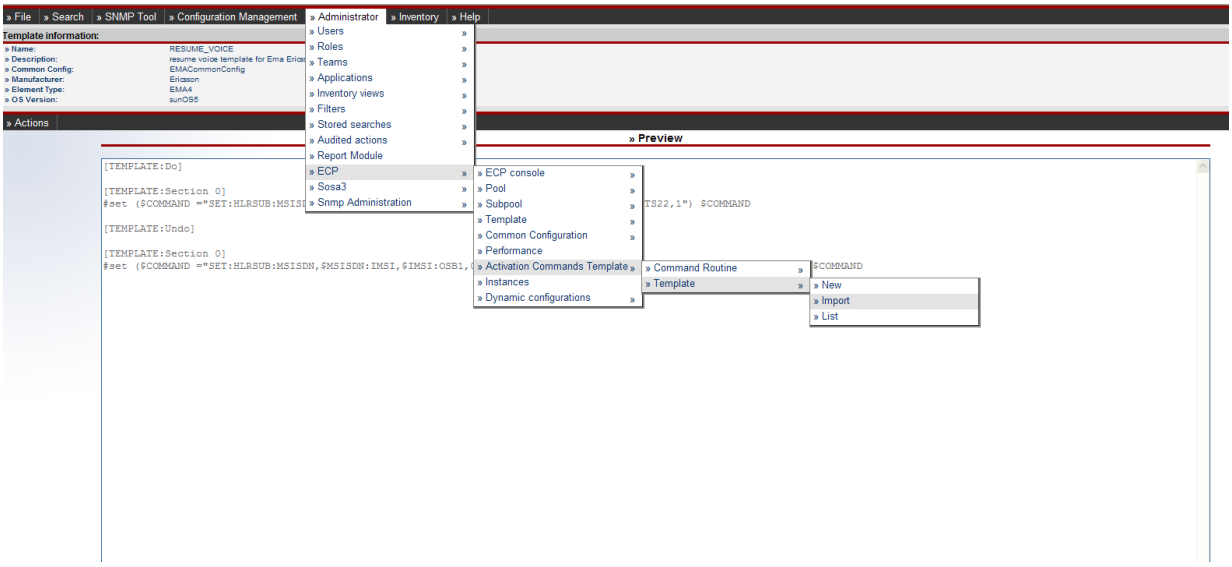
Since the ECP Template Model Web UI is deployed inside the SC (solutions container), to access it you have to open the SC in a supported web browser and go to "*`Administrator -> ECP -> Activation Commands Template`*" as shown below:



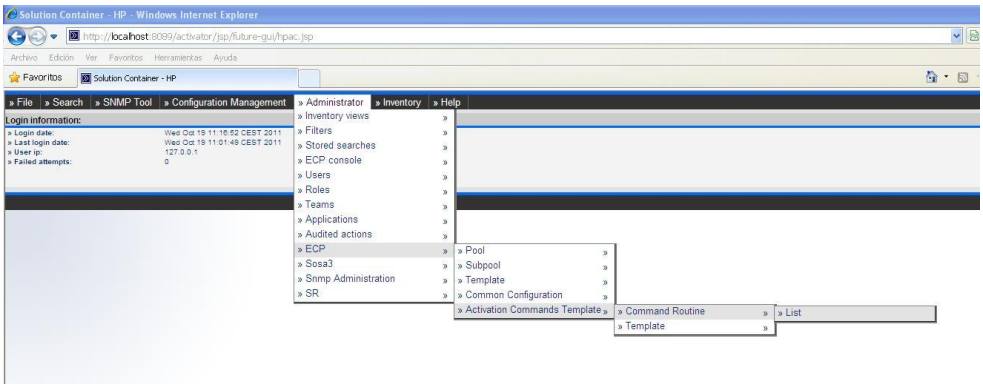
This application manages two types of objects, Command Routines and Templates. A command routine is basically a part of a template. The templates are divided in sections an each section is composed by two command routine.

Command Routines has three available operations: Search (List), View details and Edit. Templates has the following available operations: Search (List), Create, View details, Edit and Delete.

For templates, only the Search (List), Import and the Create operations can be accessed from the menus of the solutions container (see picture below). For accessing to View details, Edit, Export and Delete operations the user must first use the Search (List) and select an item listed there.



For command routines, only the Search (List) operation can be accessed from the menus of the solutions container (see picture below). For accessing to View details and Edit operations the user must first use the Search (List) and select an item listed there.

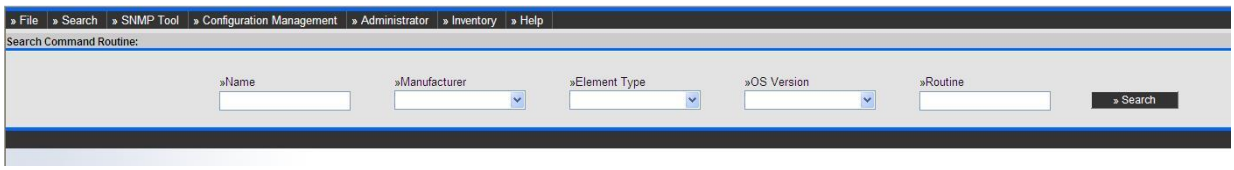


5.1.1 CommandRoutine Operations

5.1.1.1 Search/List

To access the Search (List) page, go to "`Administrator -> ECP -> Activation Commands Template -> Command Routine -> List`"

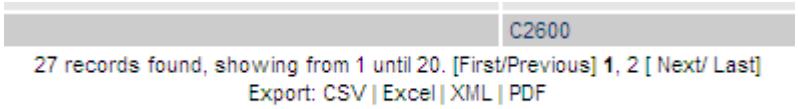
The search page will be displayed, empty for now. At the top-most part you will find the fields you can use to filter your search:



Note that it is not mandatory to fill any of them. You may leave them empty to get a list of all the items, or you may enter value in any of them, in which case the list of results will be filtered by them.

Once you hit the "`Search`" button, the results will be listed. You can click any of them to go to the Edition page, from where you will be able to edit the content for that item.

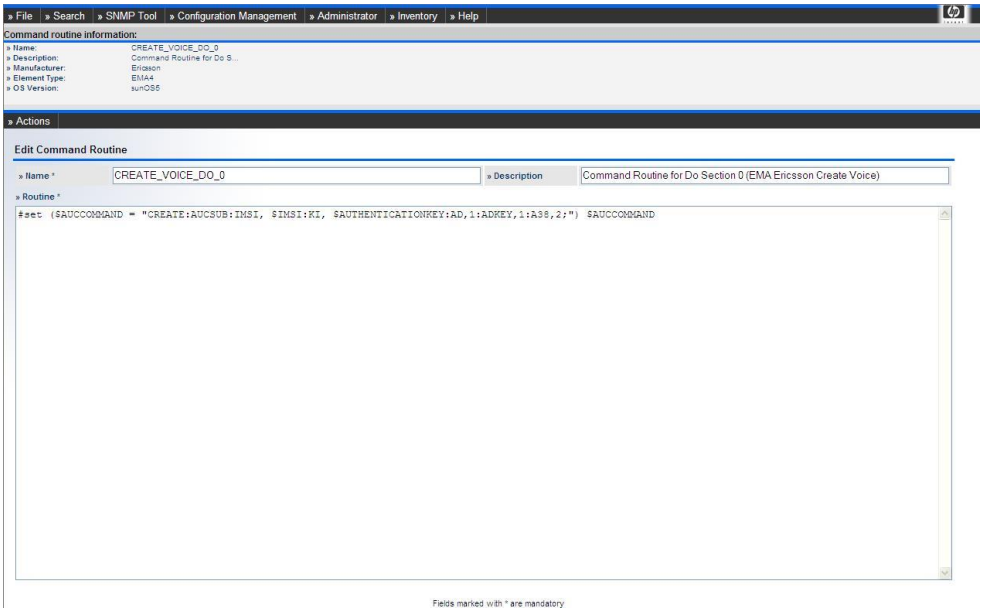
The bottom-most controls in the page allows you to navigate through the result pages (if there is more than 1) and even export the list to several file formats, including PDF and Excel spreadsheets:



5.1.1.2 Edit

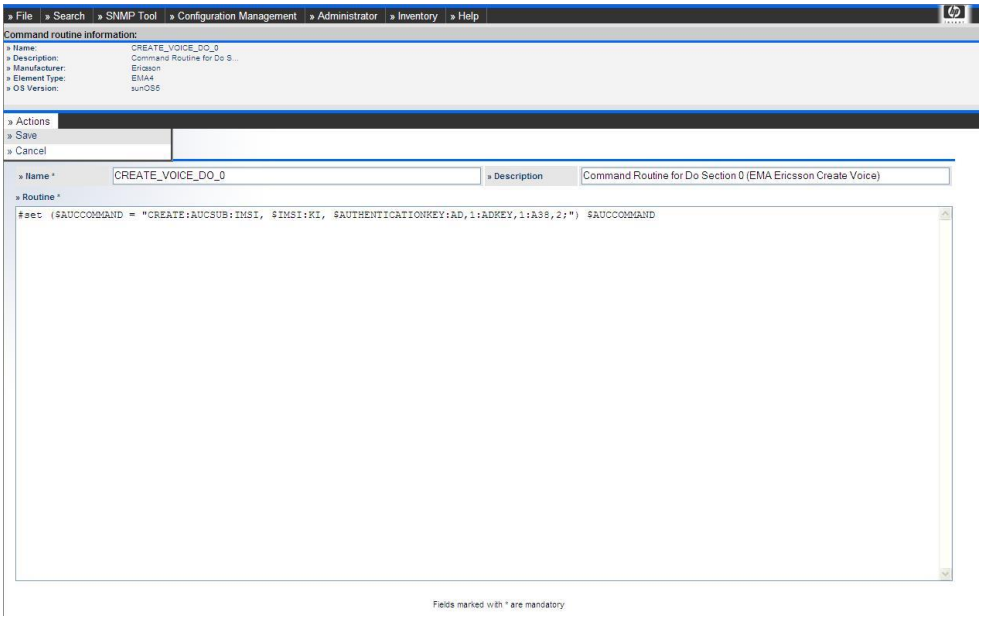
The Edit operation for a command routine is only accessible from the list page. To access the Edition page of an item perform a Search Operation and click the desired item from the results list of the search.

A formulary will be presented to you. Note the mandatory fields are marked with an asterisk'*', you cannot empty those fields.



Once you have edited the formulary, click "`Actions -> Save`" to persist the changes. If everything goes OK, you will be carried back to the List page of command routines.

If you want to cancel the edition, click "`Actions -> Cancel`. You will be carried back to the List page of command routines without any change on the previous command routine.



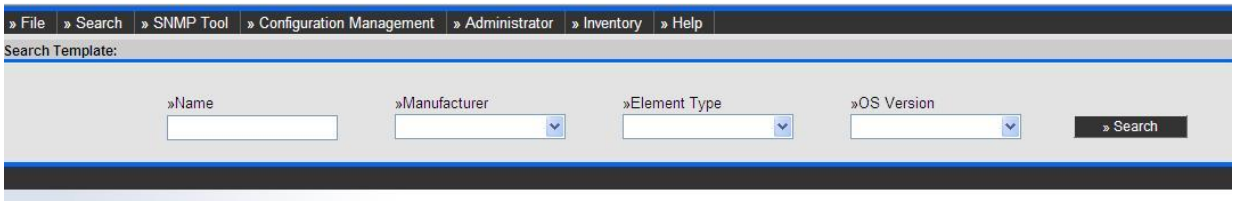
5.1.2 Template Operations

In this section are described the operation available for the templates.

5.1.2.1 Search/List

To access the Search (List) page, go to "`Administrator -> ECP -> Activation Commands Template -> Template -> List`"

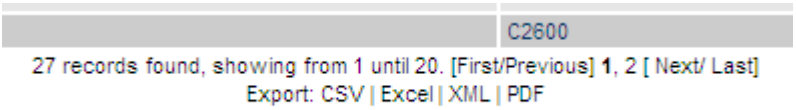
The search page will be displayed, empty for now. At the top-most part you will find the fields you can use to filter your search:



Note that it is not mandatory to fill any of them. You may leave them empty to get a list of all the items, or you may enter value in any of them, in which case the list of results will be filtered by them.

Once you hit the "`Search`" button, the results will be listed. You can click any of them to go to the Preview page, from where you can view a full preview of the template with all of its sections and command routines.

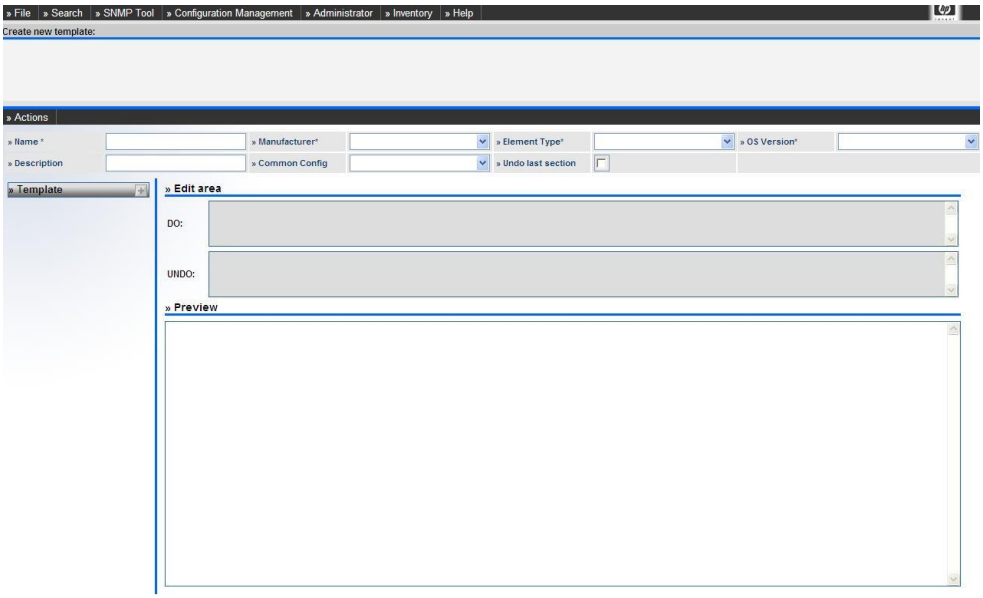
The bottom-most controls in the page allows you to navigate through the result pages (if there is more than 1) and even export the list to several file formats, including PDF and Excel spreadsheets:



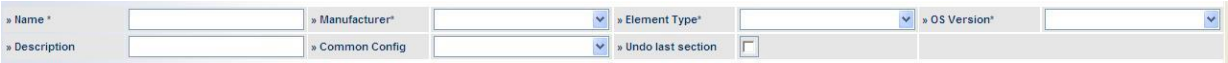
5.1.2.2 Create

To access the Creation page, go to "`Administrator -> ECP -> Activation Commands Template -> Template ->` New"

The creation page will be displayed, empty for now.



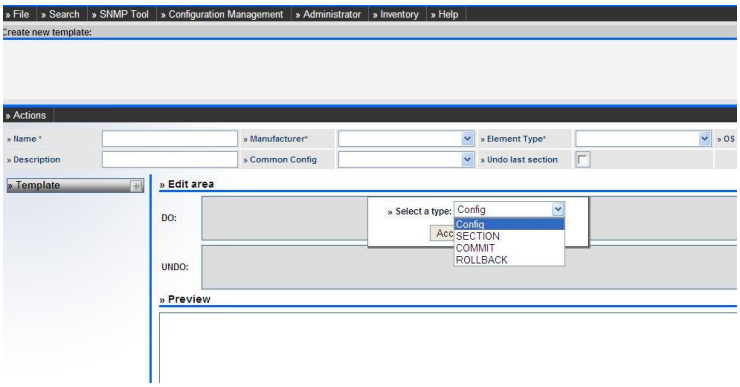
Creation page is divided in two main parts. At the top of the form, user can fill the common fields of the template: Name, Manufacturer, Element Type, OS Version, Description, Common Config and Undo last Section.



The second part of the web is a dynamically representation of the template structure.

Left side represents the tree, with all the sections that are part of the template. Right side has a top part in which the command routines for the current selected section, can be edited. And bottom part is a dynamic preview of the template. This preview will be updated automatically when user change any element of the template.

New sections can be added clicking on the button on right side of the tree representation of the template.

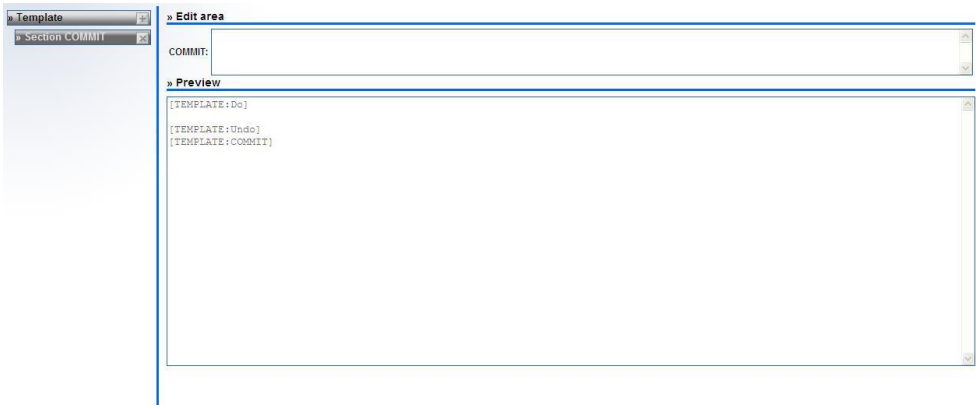


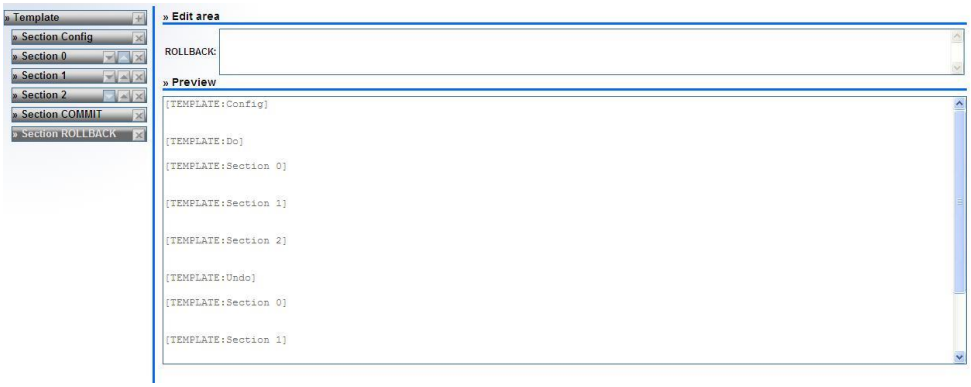
There are two different types of sections:

Non common sections: By default, those are 3 fixed sections: CONFIG, COMMIT and ROLLBACK.

Those sections can't be ordered and have a predefined order, specified on its data table of the data base of HPSA. User only can add one equal common non section in the same template.

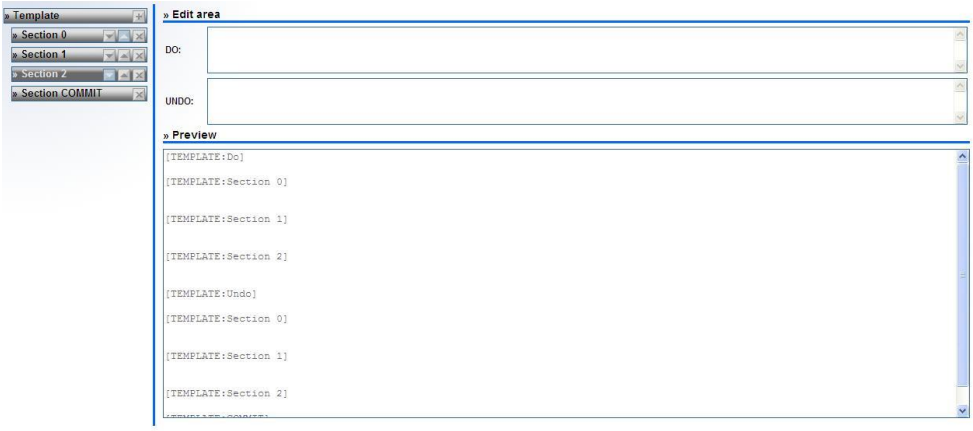
Non common sections only display one command routine area.





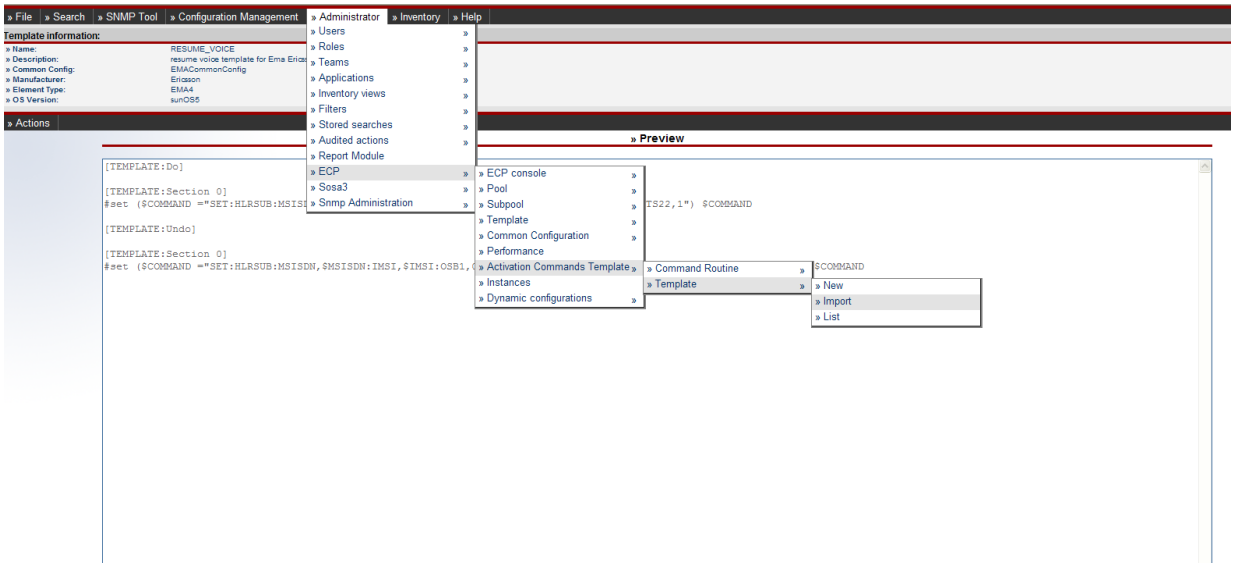
Common sections: user can add as many common sections as he want. Each common section added will have a new sequence number to identify it. This kind of sections can be ordered clicking on the arrows on right side of each section.

Common sections displays do and undo command routine areas.

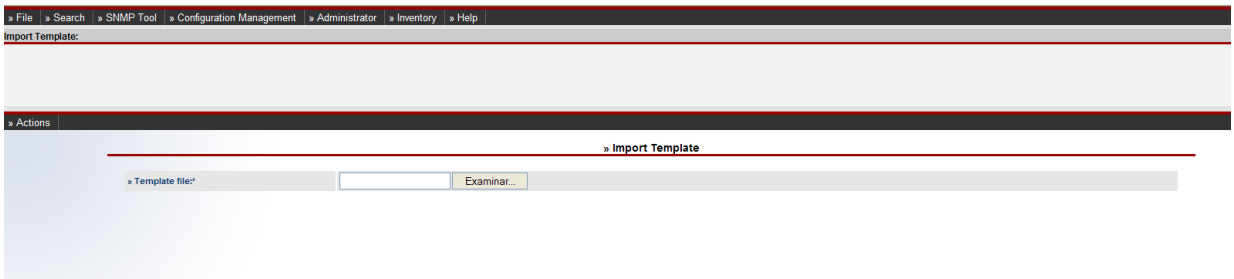


5.1.2.3 Import

To access the Search (List) page, go to "`Administrator -> ECP -> Activation Commands Template -> Template ->` **`Import`**"

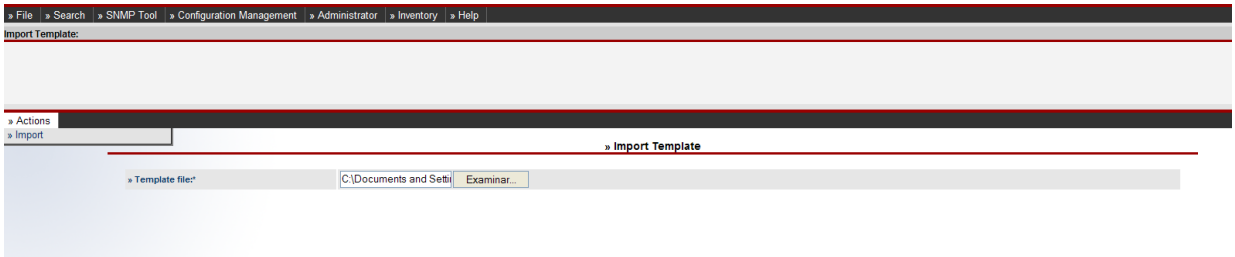


The import page will be displayed, empty for now.

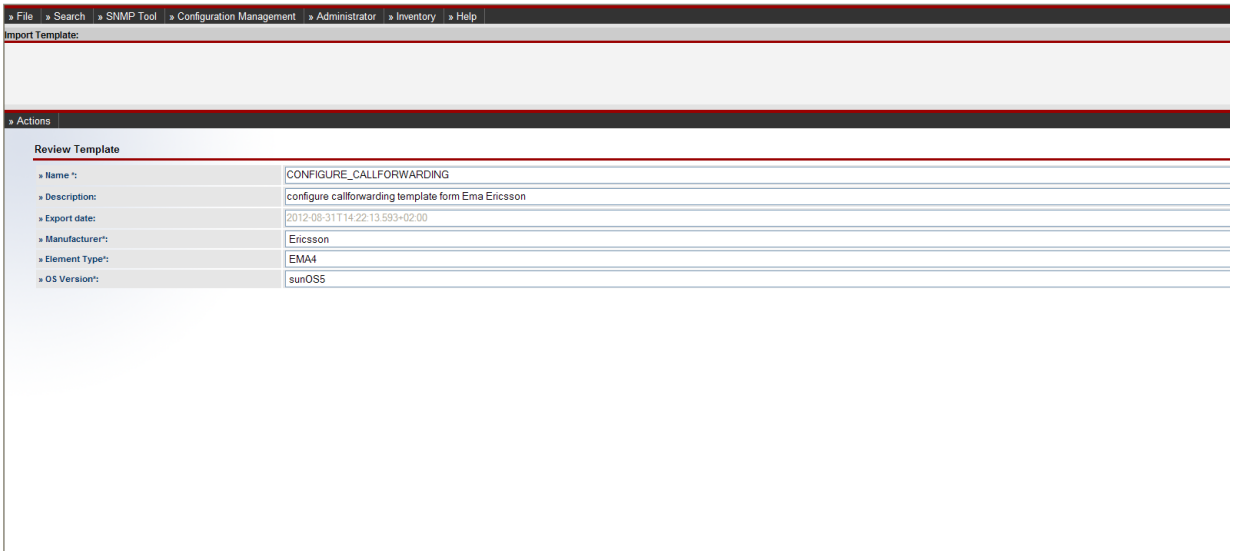


You can select the template for importing and create the template on it.

After selecting the file, click "*Actions -> Import*".



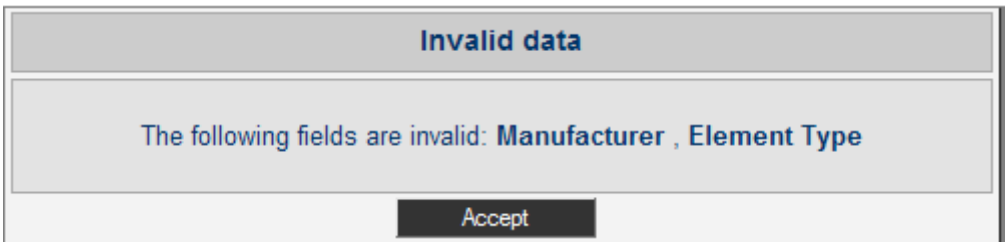
When you selected a template the system allows review the template. In this page user can change the template parameters: Name, Description, Export date, Manufacturer, Element Type and OS Version.



In this formulary the mandatory fields are marked with an asterisk'*', you cannot empty those fields.

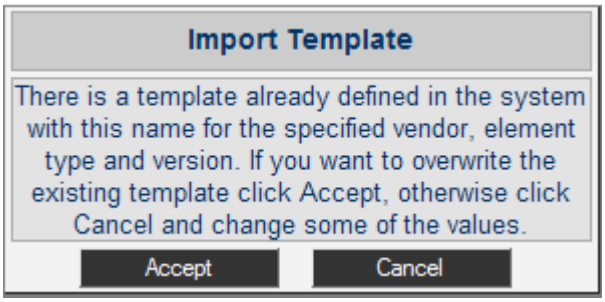
When you try import a template three cases can happen:

1) When some parameter mandatory is empty the application throws error indicating the missing parameter.



2) Try import a template with same name, vendor, OS version and element type than other template that exists in the application.

   The system alerts about this problem. Throws the following message:

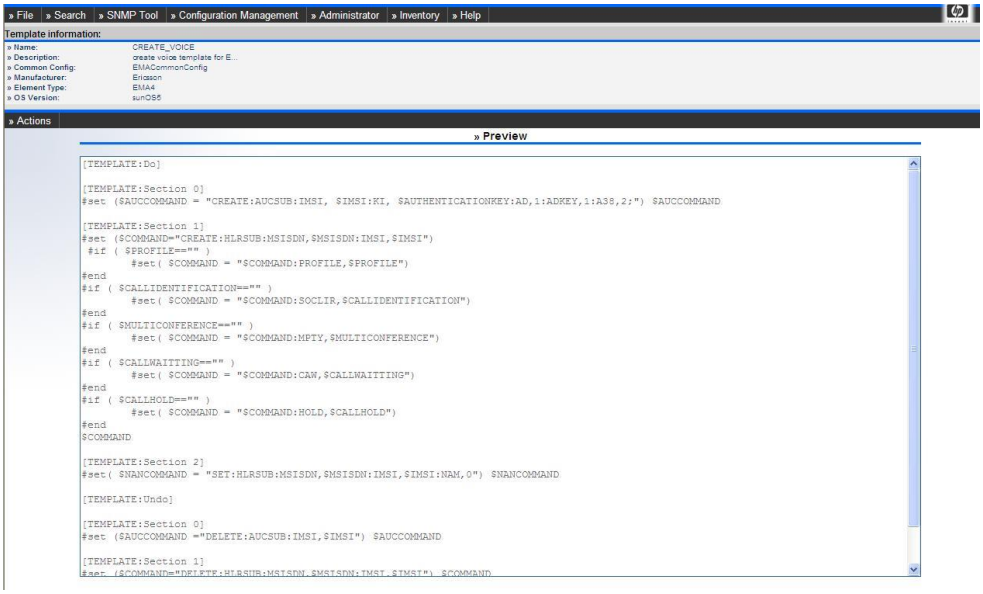


3) The import is successful. The application shows Preview of template.

5.1.2.4 Edit

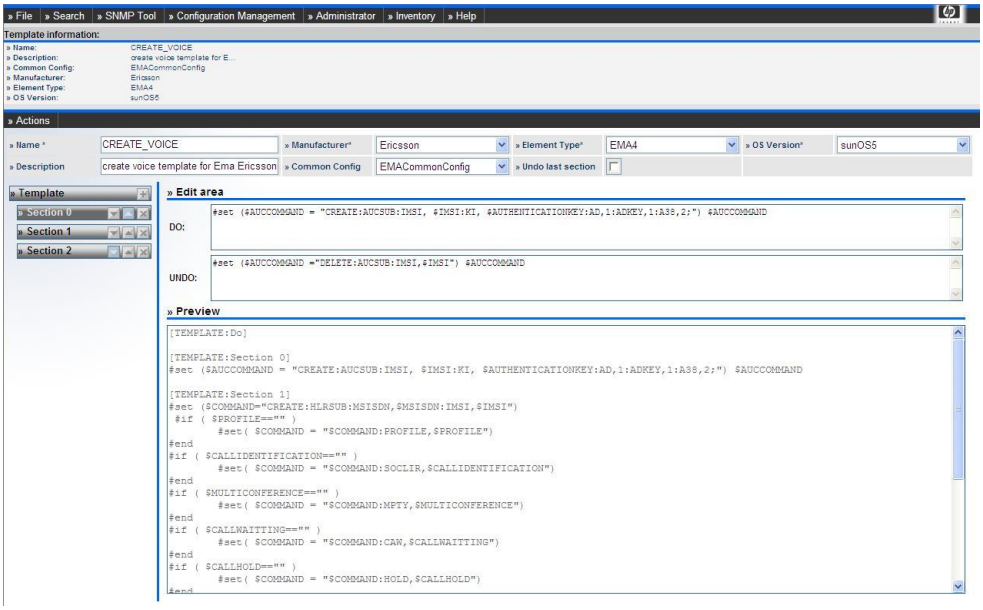
The Edit operation for an item is only accessible from the Preview page of each item. To access the Preview page of an item perform a Search Operation and click the desired item from the results list of the search.

The preview page will be displayed, with the structure of the template.



Once you are in the Preview page of the desired item, click "`Actions -> Edit`":

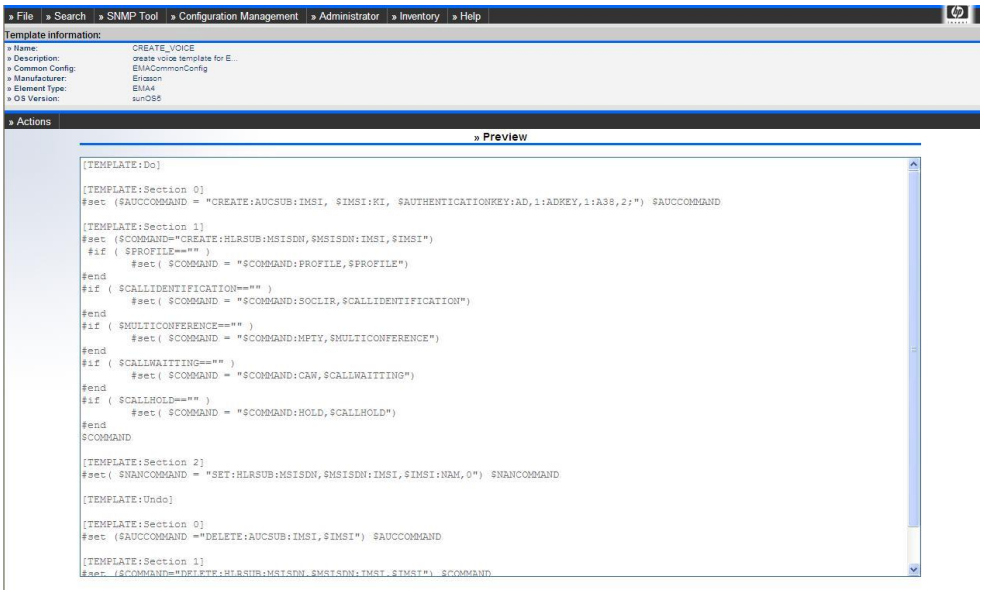
A formulary will be presented this is the same used for creation, so the behavior is the same. Please refer to the create template section of this manual to Know more about it.



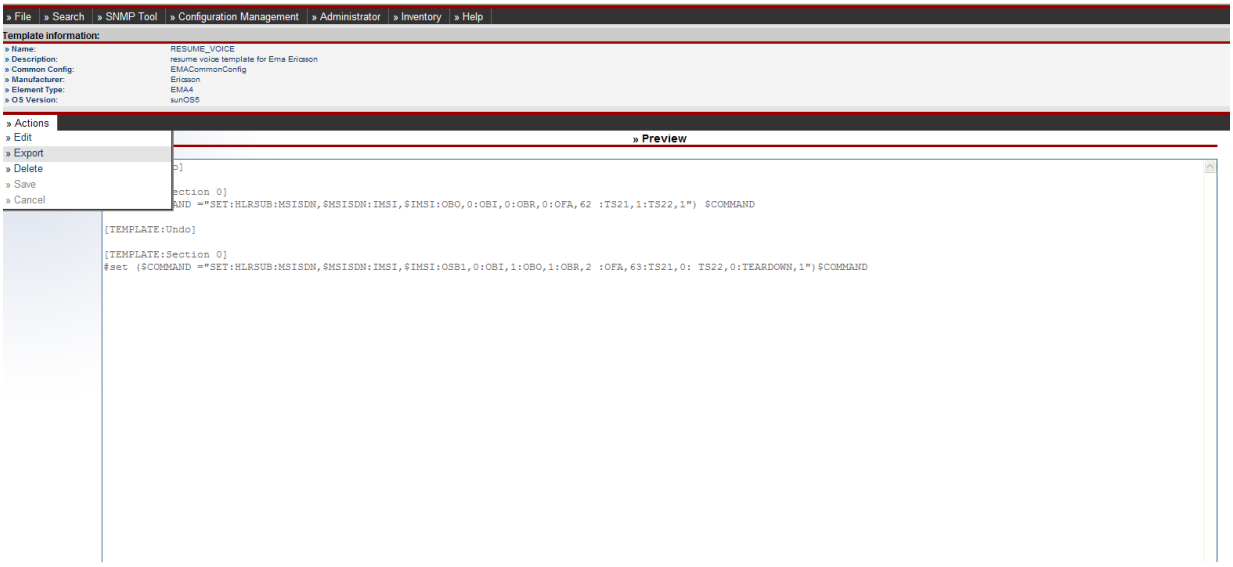
5.1.2.5 Export

The Export operation for an item is only accessible from the Preview page of each item. To access the Preview page of an item perform a Search Operation and click the desired item from the results list of the search.

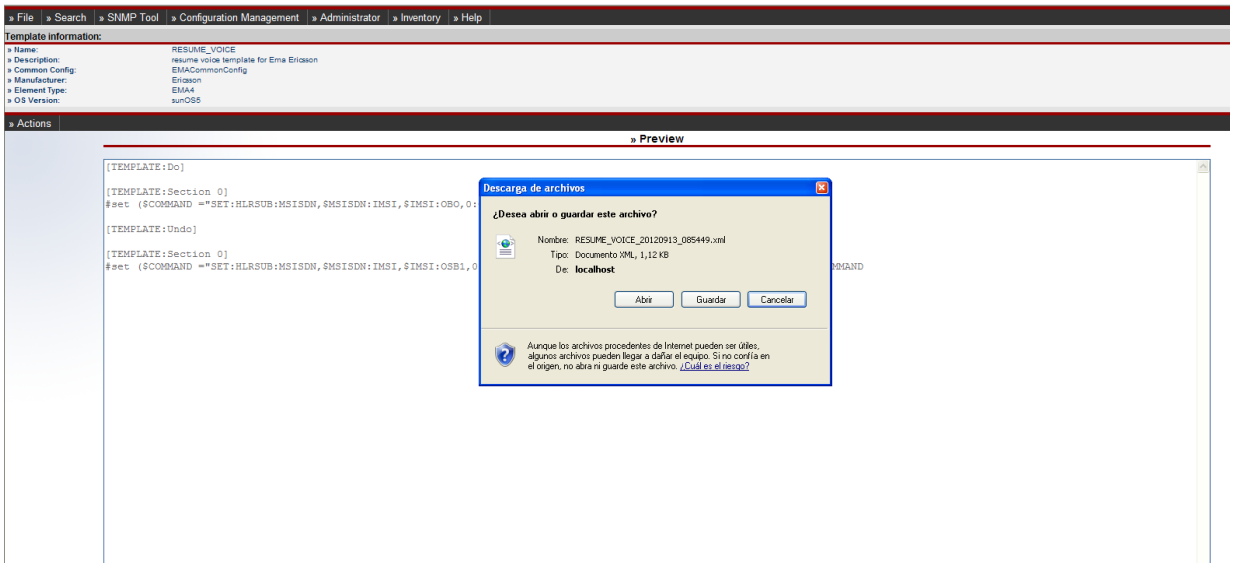
The preview page will be displayed, with the structure of the template.



Once you are in the Preview page of the desired item, click "*Actions -> Export*":



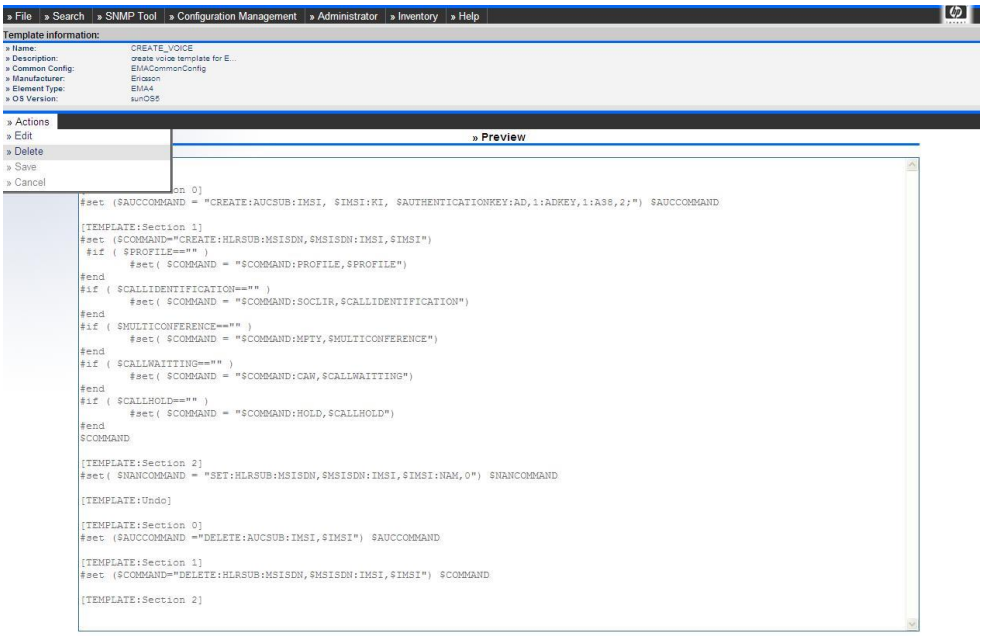
This utility lets will open or save the file in him own file system.



5.1.2.6 Delete

The Delete operation for an item is only accessible from the Preview page of each template. To access the Preview page of an item perform a Search Operation and click the desired item from the results list of the search.

Once you are in the Preview page of the desired item, click "`Actions -> Delete`".



This will delete the selected Template, as well as all its relationships with Sections and Command Routines. A confirmation dialog will be displayed.

5.2 Querying Command Templates from Workflows

QueryCommandTemplate is the node is responsible for consulting the command template from the commands cache. The class name for this node is com.hp.ov.activator.mwfm.component.tmpc.QueryCommandTemplate and it recognizes the following configuration parameters:

Parameter	Required	Description	Default
module	Yes	The name of the module invoked	None
result_status	Yes	The name of the case-packet variable in which the status will be stored	None
name	Yes	The name of the service for which are querying the template	None
manufacturer	Yes	The name of the manufacturer of the target in which the service is going to be activated	None
element_type_group	Yes	The name of the manufacturer of the networkelement in which the service is going to be activated	None
os_version_group	Yes	The name of the osversiongroup of the networkelement in which the service is going to be the name of the case-packet variable in which the template will be stored. The template contains all the commands for the service activation	None
Template	Yes	The name of the case-packet variable in which the template will be stored. The template contains all the commands for the service activation	None
common_config	Yes	The name of the case-packet variable in which the common config name will be stored	None

6 Web Service Connectivity

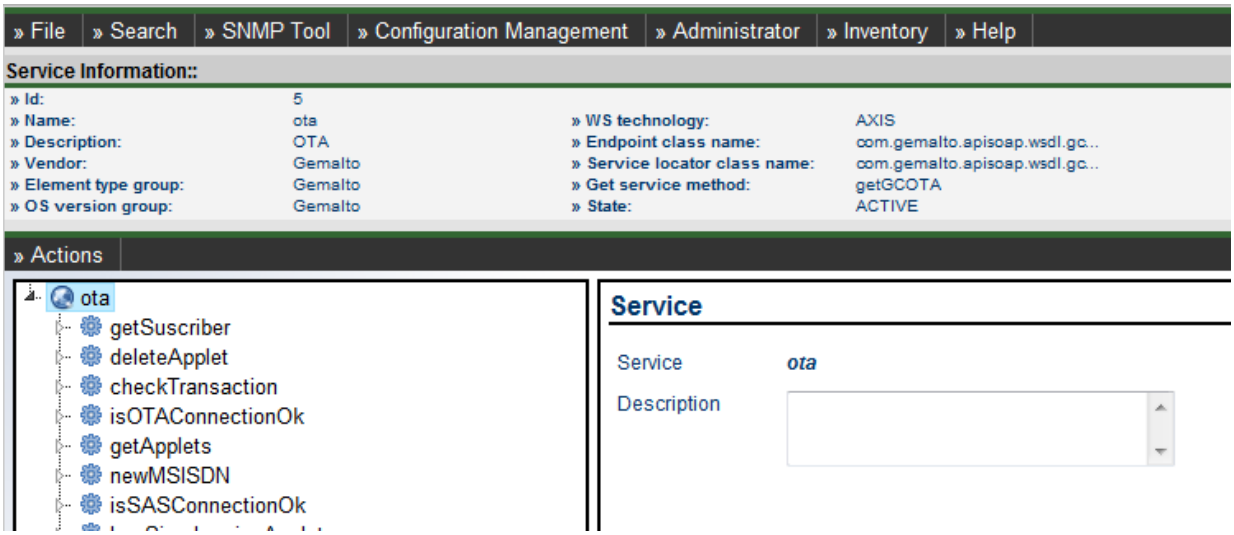
The WSC (Web Service Connectivity) feature of MSA allows generating service descriptors from a given WSDL file to invoke web service operations from within workflows. In order to get this feature to work the WSC module must be defined in the mwfm.xml file as explained in the Installation and Administration Guide.

6.1 Customizing Service Descriptors

Service descriptors define how the WSC module makes requests to a given web service endpoint using the stub generated for it.

When generating a service for a given WSDL file, a descriptor is automatically generated along with the corresponding service stub (Java client code required to perform requests to the service). These descriptors contain information about the supported operations, their corresponding overloads (if any) and input/output/fault parameters based on the Java methods in the framework-generated stub and their arguments.

Automatically generated descriptors can then be modified in order to add aliases for parameters or operations to ease workflow development. In order to edit the service descriptor corresponding to a given service, select *Actions → Service → Edit Descriptor* from the service information view in the WSC application of the Solution Container (refer to the installation and administration reference for more information on this) and you will be presented with a screen like the following:



From here you can customize the service descriptor. The tree view in the left represents the descriptor structure. In order to customize a descriptor element, just select it from the tree and its details will appear in the right pane.

The topmost element denoted by the icon represents the **service** itself, and only the description (for documentation purposes) can be modified.

Under it are the service **methods**, represented by the icon. Methods correspond to web service operations and have the following editable fields:

- **Alias**: an alternative name for the method that can be used in workflows to refer to it when using the *WSCRequestNode*.

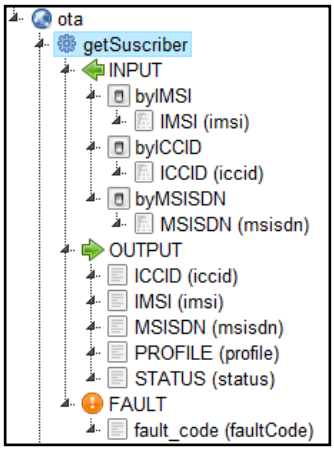
- **Description**: for documentation purposes.

Apart from this, methods may have input/output and fault parameters as seen in the example below:

The INPUT group contains input parameters. They may be optionally grouped into request configurations, represented by the icon, if the method is overloaded (i.e. there are several operations with the same name in the original WSDL).

The OUTPUT group contains output parameters.

The FAULT group contains an alternate collection of output parameters in the case the invocation results in a fault response.



Only parameters and request configurations are selectable, the group items are there for tree structure purposes.

More detailed explanations of the different elements follow:

Request configurations represent the different variants of a given method. Each of them has a different set of input parameters. From the corresponding detail form the name of the request configuration may be customized so it can be used from workflows instead of the default *methodname#N*. Non-overloaded methods have only one request configuration and thus it is not displayed in the tree, with its input parameters shown just under the INPUT group.

Input parameters may be represented by the , or icons depending if they represent basic, bean or array member types respectively. Upon selecting those the following fields will be displayed:

- **Path**: the parameter path. More information on paths below. The path is displayed for informative purposes and it cannot be modified.
- **Class**: the Java class name corresponding to the parameter, displayed for informative purposes only if it is defined in the descriptor.
- **Alias**: the parameter alias. An input parameter can be referenced by its alias instead of the complete path. Aliases must be unique inside a given request configuration. When a parameter is aliased, the alias will be displayed in the tree in place of the parameter path, which will be displayed between parentheses aside it.
- **Default Value**: default value for the parameter if it is not specified when making the request.
- **Description**: for documentation purposes.
- **Mandatory**: if checked, the parameter must be specified when making a request. Mandatory parameters are displayed in **bold** in the tree.
- **Encrypted**: if checked, the value assigned to this parameter at request time will be assumed to be encrypted, and thus it will be decrypted before sending it to the target equipment.

Note: for array member parameters only the path, class and description fields will be displayed as they cannot be aliased.

Output/fault parameters may be represented by the , or icons depending if they represent basic, bean or array member types respectively. Upon selecting those the following fields will be displayed:

- **Path**: the parameter path. More information on paths below. The path is displayed for informative purposes and it cannot be modified.

- **Alias**: the parameter alias. An output/fault parameter can be referenced by its alias instead of the complete path. Aliases must be unique inside a given OUTPUT/FAULT group. Only aliased parameters will be included in the output parameter map when using *output_attr_map* in *WSCRequestNode*. When a parameter is aliased, the alias will be displayed in the tree in place of the parameter path, which will be displayed between parentheses aside it.
- **Description**: for documentation purposes.

Note: for array member parameters only the path and description fields will be displayed as they cannot be aliased.

Once you are done customizing the descriptor you can save it by selecting *Actions → Descriptor → Save*. To discard all changes made to the descriptor, select *Actions → Descriptor → Cancel*.

6.2 Paths

Paths used to reference attributes are based on the following syntax:

```
path_element1.path_element2…path_elementN
```

Where the sequence of the path elements represents a full path inside the web service request or response/fault body to locate the attribute. Each element in the path must correspond to the name of a property of the previous object, while array indexes are expressed in the form *object#index*. So, the following path

```
foo.bar.baz#2.qux
```

would translate to

```
foo.getBar().getBaz()[2].getQux()
```

in Java code.

6.3 Making Web Service Requests from Workflows

In order to make requests to web services from workflows through the WSC module, a new node by the name WSCRequestNode is included. The class name for this node is com.hp.ov.activator.mwfm.component.wsc.WSCRequestNode and it recognizes the following configuration parameters:

Parameter	Required	Description	Default
equipment_target	Yes	The name of a named endpoint or the actual endpoint URL	None
module	Yes	The name of the WSCModule to be used for making the request	None
service_name	Yes	The name of the service to be invoked	None
method_name	Yes	The name of the method to be invoked (web service method name or alias)	None
request_configuration	No	The name of the request configuration to use.	default
exec_time	No	The name of variable where the execution time is stored.	None

request_trace	No	The name of variable where the trace of the request is stored.	None
response_trace	No	The name of variable where the trace of the response is stored.	None
throw_excep	No	Used to throw exceptions or use RET_VALUE and RET_TEXT to return information about the errors without launching exception.	true
input_attr_map	No	A map containing input attributes values. The map keys are interpreted as attribute alias or paths referencing request elements (input attributes) that must be assigned the corresponding values.	None
output_attr_map	No	A map specifying where output attributes values should be stored. The map keys are interpreted as attribute alias or paths referencing response elements (output attributes) and the corresponding values are the names of the variables where the attribute value must be stored.	None
input_attr_name0 **input_attr_name1** **...** **input_attr_nameN**	No	An attribute alias or path referencing a request sub-element (input parameter).	None
input_attr_value0 **input_attr_value1** **...** **input_attr_valueN**	No	Value to be assigned to the corresponding request attributes. It may be a variable or a constant.	None
output_attr_name0 **output_attr_name1...** **output_attr_nameN**	No	An attribute alias or path referencing a response sub-element (output parameter).	None
output_attr_value0 **output_attr_value1...** **output_attr_valueN**	No	The name of the case-packet variable where the content of the corresponding output attribute should be stored.	None

For more information on how input/output parameter paths work, please refer to the previous section.

In case a specific attribute name is defined in both the input/output_attribute_map and directly specified as an input/output_attribute_attr_name/value pair, the value specified for the pair takes precedence.

Note that in order to make requests using a certain WSC service it has to be properly generated and in the ACTIVE state. Both services and named endpoints are loaded along with the HPSA configuration, so after making any changes to them the HPSA configuration must be reloaded in order for them to take effect.

What follows is an example of a WSC invocation from one of the workflows shipped with the MSA solution:

```
<Process-Node>
  <Name>Call WS</Name>
```

```
  <Action>
    <Class-Name>com.hp.ov.activator.mwfm.component.wsc.WSCRequestNode</Class-Name>
    <Param name="exec_time" value="constant:invocation_time"/>
    <Param name="input_attr_map" value="S_INPUT"/>
    <Param name="method_name" value="constant:deleteUser"/>
    <Param name="module" value="wsc_module_name"/>
    <Param name="request_trace" value="COMMAND_SENT"/>
    <Param name="response_trace" value="NE_OUTPUT"/>
    <Param name="service_name" value="constant:voicemail"/>
    <Param name="target_target" value="beanEndpoint.Targettarget"/>
    <Param name="throw_excep" value="false"/>
    <Param name="input_attr_name0" value="operID"/>
    <Param name="input_attr_value0" value="Targetuser"/>
    <Param name="output_attr_name0" value="response"/>
    <Param name="output_attr_value0" value="error_code"/>
    <Param name="input_attr_name1" value="operPwd"/>
    <Param name="input_attr_value1" value="Targetpassword"/>
  </Action>
  <Next-Node>Switch</Next-Node>
</Process-Node>
```

Here we are passing the S_INPUT parameter map exactly as provided by SOSA as the input parameter map; this along with descriptor customization can make workflow development much easier and manageable.

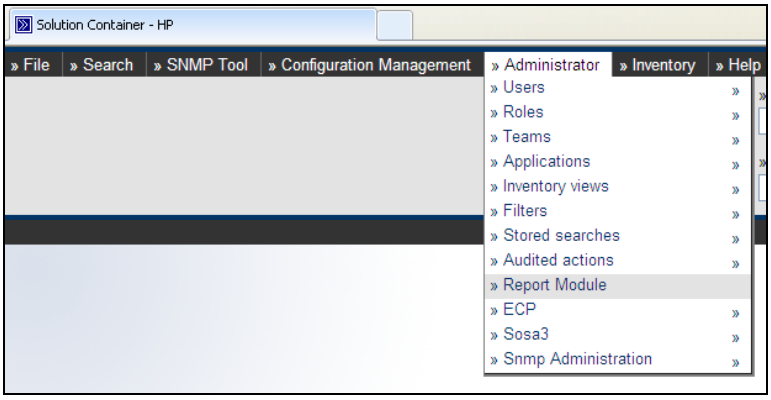
7 Report Module

The report module allows for defining arbitrary reports on database information, being able to specify filter conditions and how the query results are presented.

The report are defined in an xml file, where basically are defined the queries used for each report, the format of this xml file are described in the section 7.2 of this document.

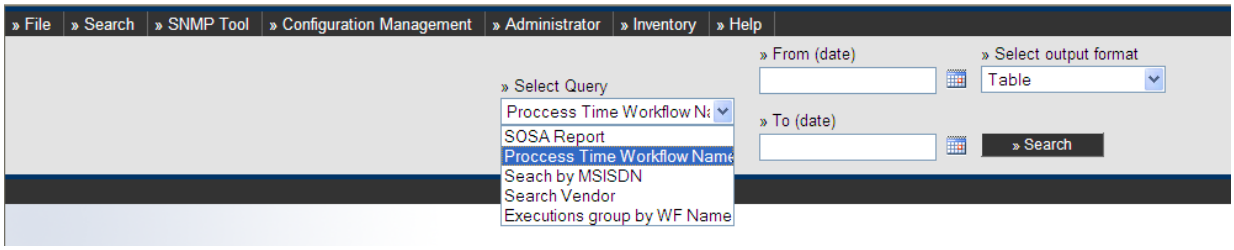
7.1.1 Accessing the Report Module Web UI

Report Module (RM) is a web application part of the Solution Container (SC). To access RM open the SC in a supported web browser and go to "`Administrator -> Report Module`" as shown below:

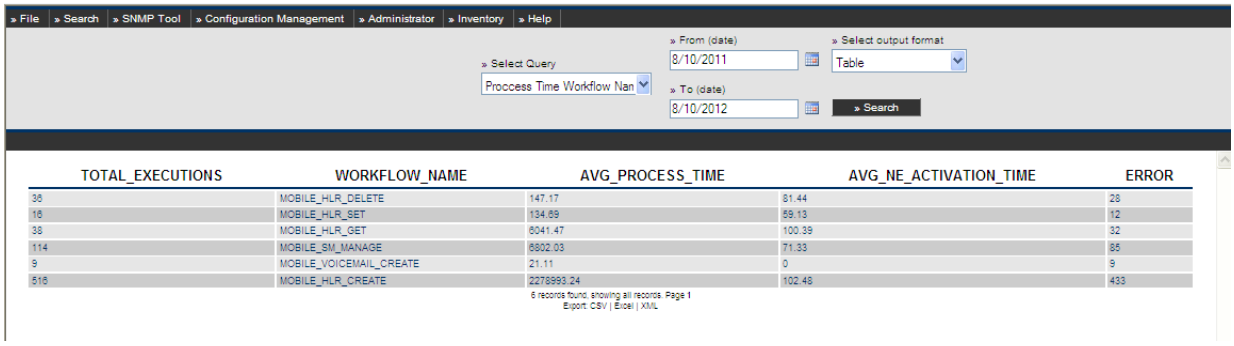


7.1.2 Available Operations

There is only one possible operation to execute through this module: Search operation. RM will execute the query selected by the user with the input and output parameters the user has previously set.



Once you hit the "`Search`" button, the results will be listed, it will forward to the resulting data page with the format specified by the user.



The input form is automatically generated based on the configuration file.

7.2 Report Module configuration

This module reads an xml file with all required information about the queries definition.

This file is called "ReportModuleQueryDefinition.xml" and it's on the next path:
$JBOSS_HOME/standalone/deployments/hpsa.ear/ep.war/WEB-INF/classes/reportmodule.
Note that this file is deployed in the WEB-INF server directory, so it is needed to restart the server in order to see any change that it is committed in this configuration file.

7.2.1 Query Section

The ReportModuleQueryDefinition file can have as many query sections as it is needed. Those sections represent each possible operation that could be executed in the Report Module UI.

Code example:

```
  <query>
    <name>Proccess Time Workflow Name</name>
    <description>Proccess Time Workflow Nam</description>
    <order>20</order>
    <maxlengthresult>200</maxlengthresult>
    <sql><![CDATA[
        select sum(total) as Total_Executions,WORKFLOW_NAME_DO as Workflow_Name,sum(AVG_PROC_TIME) as
AVG_Process_Time,
              sum(AVG_NE_ACT) as AVG_NE_Activation_Time,sum(error) as Error
              from
            (SELECT  COUNT(*) TOTAL,
            WORKFLOW_NAME_DO,
            ROUND(AVG(PROCESSING_TIME),2) AVG_PROC_TIME,
            ROUND(AVG(NE_ACT_TOTAL_TIME),2) AVG_NE_ACT,
            0 AS ERROR
                      FROM HISTORY_HBM_SA
            WHERE ( (START_TIME BETWEEN TO_DATE ('$From$', 'DD-MM-YY HH24:MI:SS')
            AND TO_DATE ('$To$', 'DD-MM-YY HH24:MI:SS')))
        GROUP BY WORKFLOW_NAME_DO
        UNION ALL
        SELECT 0 TOTAL,
           WORKFLOW_NAME_DO,
           0 AVG_PROC_TIME,
           0 AVG_NE_ACT,
           COUNT(*) AS ERROR
        FROM HISTORY_HBM_SA
        WHERE ( (START_TIME BETWEEN TO_DATE ('$From$', 'DD-MM-YY HH24:MI:SS')
          AND  TO_DATE ('$To$', 'DD-MM-YY HH24:MI:SS')))
        AND (sosa_code <> 0)
     GROUP BY WORKFLOW_NAME_DO )
          group by workflow_name_do
    ]]></sql>
    <input>
      <parameter>
        <name>From</name>
        <type>date</type>
```

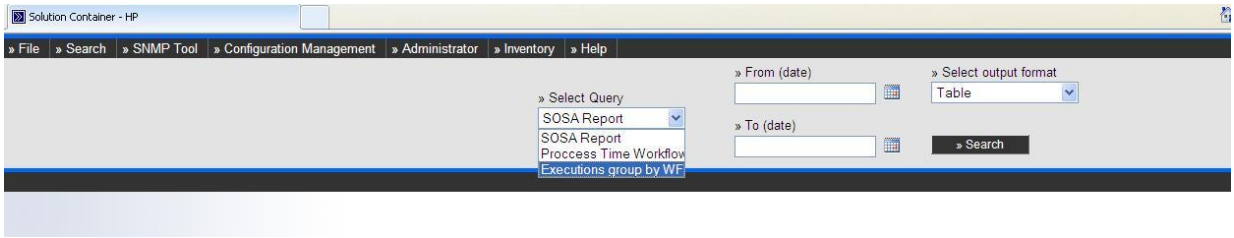
```
      <description>From Date</description>
    </parameter>
    <parameter>
      <name>To</name>
      <type>date</type>
      <description>To Date</description>
    </parameter>

      </input>
          <redirections>
                  <redirect>
                          <jsp>table.jsp</jsp>
                          <type>Table</type>
                  </redirect>
          </redirections>
  </query>
```

7.2.1.1 Name Section

Name section is the identifier of the query block. This name appear in UI on select field to allow the user to select the desired query to execute

```
<name>Proccess Time Workflow Name</name>
```



7.2.1.2 Description Section

Description section is a long description of the query. No more utility.

```
<description>Proccess Time Workflow Nam</description>
```

7.2.1.3 Order Section

Order section allows the user to specify the order that queries will have on select field in the application.

```
<order>20</order>
```

7.2.1.4 Max Length Result section

User can define the max number of results will be shown as a result of a query.

```
<maxlengthresult>200</maxlengthresult>
```

7.2.1.5 SQL Section

This section contains the SQL query that can be defined to recover the information desired.

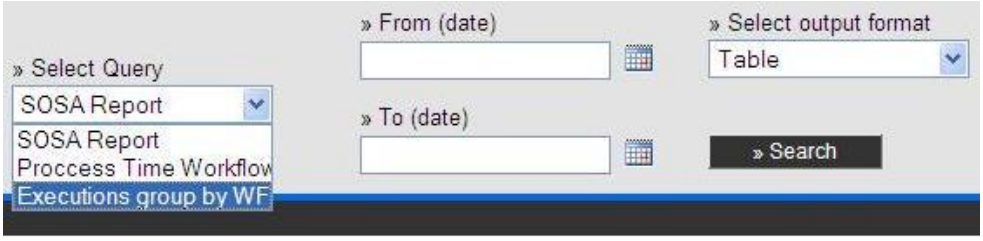
Note that *it is very important* to use the CDATA tag if you want to use any “special” character like <, > or similar in the query, because the configuration file it’s an xml file and user needs to escape this kind of “special” characters.

```
<sql><![CDATA[
                .........
]]></sql>
```

7.2.1.6 Input Section

This section contains as many input parameters as needed as criteria in the query. UI draw an input field for each parameter is defined. If field is a date field, a calendar component will be rendered next to it. Boolean parameter will be rendered as a radio decision input.

```
    <parameter>
  <name>From</name>
  <type>date</type>
  <description>From Date</description>
</parameter>
<parameter>
  <name>To</name>
  <type>date</type>
  <description>To Date</description>
</parameter>
```



On query sections, parameters must be used as it's shown below, between $ symbol:

```
WHERE ( (START_TIME BETWEEN TO_DATE ('$From$', 'DD-MM-YY HH24:MI:SS'ription>From Date</description>
```

7.2.1.7 Redirections Section

This section allows defining different types of data result representation.

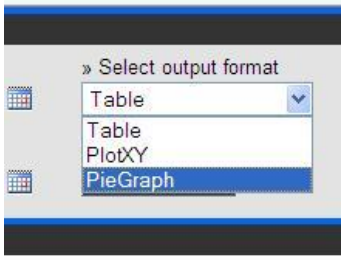
```
<redirections>
        <redirect>
                <jsp>table.jsp</jsp>
                <type>Table</type>
        </redirect>
</redirections>
```

In this section it is defined the JSP previously implemented to be displayed with the result, and the type of representation required.

It is possible to define as many different redirections as needed. For each redirection defined, one option will be filled on the select output format of the UI.

```
<redirections>
        <redirect>
                <jsp>table.jsp</jsp>
                <type>Table</type>
        </redirect>
        <redirect>
                <jsp>/jsp/reportmodule/customjsp/mobileSosaHistoryAxisGraph.jsp</jsp>
```

```
                <type>PlotXY</type>
            </redirect>
            <redirect>
                <jsp>/jsp/reportmodule/customjsp/mobileSosaHistoryPieGraph.jsp</jsp>
                <type>PieGraph</type>
            </redirect>
        </redirections>
```



7.3 Redirections (Types of Representations)

In previous section it was defined that this application allows define different types of data result representation.

By default, users always have table representation. But users can implement new different output format representations.

Any new representation must be implemented in a new JSP file. This new JSP file must be located in the path: $JBOSS_HOME/server/diagnostic/deploy/hpovact.sar/activator.war/jsp/reportmodule/customjsp This directory contains all customized implemented files.

An example of a pie chart will be:

```
<%@page import="java.math.BigDecimal"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@ page import="com.hp.reportmodule.utils.CustomRowSetDynaClass"%>
<%@ page import="java.util.List"%>
<%@ page import="java.util.Iterator"%>
<%@ page import="org.apache.commons.beanutils.DynaBean"%>
<%@ page import="com.hp.reportmodule.reportmodulequerydefinition.Parameter;"%>
<html>
<script src="/activator/javascript/hputils/datetimepicker.js"></script>
<script language="javascript" type="text/javascript" src="/activator/javascript/reportmodule/jquery.js"></script>
<script language="javascript" type="text/javascript" src="/activator/javascript/reportmodule/jquery.jqplot.js"></script>
<script language="javascript" type="text/javascript" src="/activator/javascript/reportmodule/excanvas.js"></script>
<script language="javascript" type="text/javascript" src="/activator/javascript/reportmodule/plugins/jquery.jqplot.js"></script>
<script language="javascript" type="text/javascript"
src="/activator/javascript/reportmodule/plugins/jqplot.pieRenderer.js"></script>

<%
        String useRandomColor = (String) request.getSession().getAttribute(
                                        com.hp.ov.activator.mwfm.futuregui.servlet.Constants.USE_RANDOM_COLOR);
        String mainColor = (String) request.getSession().getAttribute(
                                        com.hp.ov.activator.mwfm.futuregui.servlet.Constants.APP_MAIN_COLOR);

        CustomRowSetDynaClass rsdc = (CustomRowSetDynaClass) request.getSession().getAttribute("rsdc");
        SimpleDateFormat sdf = new SimpleDateFormat();
        SimpleDateFormat sdf2 = new SimpleDateFormat();
```

```
            Object[] values = rsdc.getColumn("OK").getColumnValues().toArray();
            Object[] values2 = rsdc.getColumn("KO").getColumnValues().toArray();
%>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<link rel="stylesheet" type="text/css" href="/activator/javascript/reportmodule/jquery.jqplot.css" />
<link rel="stylesheet" href="/activator/css/future-gui/estilos<%=(useRandomColor.equals(com.hp.spain.futuregui.Constants.TRUE)) ?
mainColor: ""%>.css">
<link rel="stylesheet" href="/activator/css/future-
gui/subestilos<%=(useRandomColor.equals(com.hp.spain.futuregui.Constants.TRUE)) ? mainColor: ""%>.css">
<style type="text/css">
body {
            overflow: scroll;
}
</style>
<style type="text/css" media="screen">
.jqplot-axis {          font-size: 1em;}

.jqplot-axis {          font-size: 1em;}
.jqplot-legend {     font-size: 2em;  }
</style>
<title>Plot</title>
</head>
<body>

<script type="text/javascript" language="javascript">
                        $(document).ready(
function(){
            $.jqplot.config.enablePlugins = true;
            var totalOK=0;
            var totalError=0;
    <%
    for (int i=0; i<rsdc.getRows().size(); i++){
    %>
      totalOK=totalOK+<%=values[i]%>;
      totalError=totalError+<%=values2[i]%>;
    <%
            }
            %>
            var data=[['OK',totalOK],['ERROR',totalError]];
            var plot = jQuery.jqplot ('chart1', [data],
                        {
                                    gridPadding: {top:0, bottom:38, left:0, right:0},
                                    seriesDefaults:{renderer:$.jqplot.PieRenderer, trendline:{show:false},
                                                                rendererOptions: { padding: 8, showDataLabels: true}
                                                                },
                                    legend:{show:true, placement: 'insideGrid', rendererOptions: {numberRows: 1},
                                                        location:'ne',
                                                        marginTop: '15px'
                                                        }
                        }
            );

            });


</script>
            <?php include "nav.inc"; ?>
            <div id="chart1"style="margin-top: 20px; margin-left: 20px; width: 1000px; height: 400px;"></div>
            <div style="padding-top: 20px">          </div>
</body>
</html>
```

Note that for this application it has been used a graphic library called **jqplot.**
User must implement its own JSP files with this library. You can found more information about jqplot in the next URL:
http://www.jqplot.com/

jqplot usage example: http://www.jqplot.com/docs/files/usage-txt.html
jqplot examples: http://www.jqplot.com/tests/

8 SmartBean Search Module

This is an HPSA module intended to cache some information in the memory to avoid access to the database. This functionality is used to improve the performance in the workflow.

The information that will be cache in memory need to be defined into the configuration table, this configuration table is stored in the database.

Additionally this

8.1 Indexed Bean Configuration

SBS Module provisions SBSM_CONF table with the configuration of the beans that must be cached in memory by the module.

SBSM_CONF is defined in the configuration inventory file and it will be provisioned with the following mandatory fields:

- *Classname*: is a String that contains the class name of the bean which has to be cached.
- *Indexes*: is a String that contains the Indexes used for caching the bean data.

If *Indexes* field has different values, they will be separated by ":", and the different indexes for a bean will be separated by ";" as it is showed in the following example.

```
INSERT INTO SBSM_CONF (CLASSNAME, INDEXES, ISPARENT__) VALUES ('com.hp.ov.activator.cr.inventory.NetworkElement',
'Name:Cluster;NetworkElementId', 0);
```

8.2 Workflow Nodes

There are three nodes from which workflow invocations could be fulfilled:

- SBSGetBeanNode.
- SBSGetBeanWithPatternNode.
- SBSGetBeansNode.

In this section it will be described what the SBSM nodes do as well as its configuration reference.

8.2.1 SBSGetBeanNode

SBSGetBeanNode enables workflows to recover a bean instance using the values of fields indexed in the cache as key, so the key provided has to be the same as the key of the object returned.

Parameter	Required	Description	Default	Type
module	Yes	The name of the module to be used.	None	String
bean	Yes	QName of bean class.	None	String
index_field0, index_field1, ... index_fieldN	Yes (at least index_field0)	The name of the fields used to index the beans (in the same order they have been provisioned in database).	None	String
index_value0,	Yes (at least	The value for each field used	None	String

index_value1, ... index_valueN	index_value0)	as criteria.		
wildcard_field0, wildcard_field1, ... wildcard_fieldN	No	The name of the fields used as wildcard variants if a result has not been found with the index_field and index_value criteria. Note that this wildcard_fields can only be a subset of index_fields provided.	None	String
wildcard_value0 , wildcard_value1 , ... wildcard_value N	No	The value for each field when the wildcard is used.	None	String
wildcard_mode	No	Mode used to generate different criteria combinations attending to wildcards modifiers provided. The value only can be ALL_IN_ONE (only one variant is generated replacing all the values of the wildcards at the same time) or SEQUENTIAL (the number of variant is related to the number of wildcard_fields provided). See example bellow in order to understand the variants generation mechanism.	None	String
result	Yes	Name of variable where bean instance will be stored.	None	Object
config	No	Map with dynamic configuration of node. All key-value pairs can overwrite the values provided statically in the workflow.	None	Object
throw_excep	No	Is used to throw exception or use RET_VALUE and RET_TEXT to return information about the errors without launching exceptions.	true	Boolean

The following example shows how to invoke SBS Get Bean Node:

```
<Process-Node>
   <Name>Query platform to activate</Name>
   <Action>
      <Class-Name> com.hp.ov.activator.mwfm.component.sbsm.SBSGetBeanNode</Class-Name>
```

```
        <Param name="bean" value="constant:com.hp.activation.mobile.inventory.VOICEMAIL_ZTE_MAP"/>
        <Param name="module" value="constant:SmartBeanSearch"/>
        <Param name="result" value="constant:beanVoiceMail"/>
        <Param name="throw_excep" value="false"/>
        <Param name="index_field0" value="constant:Mdn"/>
        <Param name="index_value0" value="plataform_name"/>
        <Param name="index_field1" value="Targetrole"/>
        <Param name="index_value1" value="system"/>
    </Action>
</Process-Node>
```

8.2.1.1 Wildcard Feature

When the search for a certain cached bean has been performed with the *index_field* and *index_value* criteria and no results have been found, *SBSGetBeanNode* gives the opportunity to use wildcards fields as search criteria.

There are different strategies in order to generate alternative variants of search criteria, depending on the value *wildcard_mode* parameter takes:

- ALL_IN_ONE: if only one variant is generated replacing all the values of the wildcards at the same time.
- SEQUENTIAL: if the number of variant is related to the number of *wildcard_fields* provided.

Let's take the following values and wildcards as an example:

index_field0	fieldA	**index_value0**	valueA
index_field1	fieldB	**index_value1**	valueB
index_field2	fieldC	**index_value2**	valueC
wildcard_field0	fieldA	**wildcard_value0**	DEFAULT
wildcard_field1	fieldC	**wildcard_value1**	default

If *wildcard_mode* has *ALL_IN_ONE* value, the variant generated would be only:

```
DEFAULT:valueB:default
```

Otherwise, if *wildcard_mode* gets SEQUENTIAL value, the variants generated would be:

```
DEFAULT:valueB:valueC
DEFAULT:valueB:default
```

In this case, the order of *wildcard_fields* is used as sequence in the replacement of original values.

This feature can also be used by the following node (SBSGetBeanWithPatternNode).

8.2.2 SBSGetBeanWithPatternNode

SBSGetBeanWithPatternNode enables workflows to recover a bean instance using the pattern of fields indexed in the cache as key. With this node, the field keys can be read during the caching process as a regular expression and the values provided in the node will try to match one of the regular expressions used as index. It can be useful to define a mapping.

Parameter	Required	Description	Default	Type
module	Yes	The name of the module to be used.	None	String
bean	Yes	QName of bean class.	None	String
index_field0,	Yes (at least	The name of fields used to	None	String

index_field1, ... index_fieldN	index_field0)	index the beans (in the same order what they have been provisioned in database).		
index_value0, index_value1, ... index_valueN	Yes (at least index_value0)	The value for each field used as criteria.	None	String
wildcard_field0, wildcard_field1, ... wildcard_fieldN	No	The name of fields used as wildcard variants if a result has not been found with the index_field and index_value criteria. Note that wildcard_fields only can be a subset of index_fields provided.	None	String
wildcard_value0 , wildcard_value1 , ... wildcard_value N	No	The value for each field when the wildcard is used.	None	String
wildcard_mode	No	Mode used to generate different criteria combinations attending to wildcards modifiers provided. The values only can be ALL_IN_ONE (only one variant is generated replacing all the values by the wildcards at the same time) or SEQUENTIAL (the number of variant is related to the number of wildcard_fields provided). See example bellow in order to understand the variants generation mechanism.	None	String
result	Yes	Name of variable where bean instance will be stored.	None	Object
config	No	Map with dynamic configuration of node. All key-value pairs can overwrite the values provided statically in the workflow.	None	Object
throw_excep	No	Is used to throw exception	true	Boolean

		or use RET_VALUE and RET_TEXT to return information about the errors without launching exceptions.		

The following example shows how to invoke SBS Get Bean with Pattern Node:

```
<Process-Node>
   <Name>Query platform to activate</Name>
   <Action>
      <Class-Name>com.hp.ov.activator.mwfm.component.sbsm. SBSGetBeanWithPatternNode</Class-Name>
      <Param name="bean" value="constant:com.hp.activation.mobile.inventory.VOICEMAIL_ZTE_MAP"/>
      <Param name="module" value="constant:SmartBeanSearch"/>
      <Param name="result" value="constant:beanVoiceMail"/>
      <Param name="throw_excep" value="false"/>
      <Param name="index_field0" value="constant:Mdn"/>
      <Param name="index_value0" value="plataform_name"/>
      <Param name="index_field1" value="Targetrole"/>
      <Param name="index_value1" value="system"/>
   </Action>
</Process-Node>
```

8.2.3 SBSGetBeansNode

SBSGetBeansNode enables workflows to recover an array of bean instances using the values of fields indexed in the cache as key.

Parameter	Required	Description	Default	Type
module	Yes	The name of the module to be used.	None	String
bean	Yes	QName of bean class.	None	String
index_field0, index_field1, ... index_fieldN	Yes (at least index_field0)	The name of fields used to index the beans. All the index_field provided have to be included simultaneously at least in one index registered for the bean.	None	String
index_value0, index_value1, ... index_valueN	Yes (at least index_value0)	The value for each field used as criteria.	None	String
result	Yes	Name of variable where array of bean instances will be stored.	None	Object
config	No	Map with dynamic configuration of node. All key-value pairs can overwrite the values provided statically in the workflow.	None	Object
throw_excep	No	Is used to throw exception or use RET_VALUE and	true	Boolean

		RET_TEXT to return information about the errors without launching exceptions.		

The following example shows how to invoke SBS Get Beans Node:

```
<Process-Node>
   <Name>Query platforms to activate</Name>
   <Action>
      <Class-Name> com.hp.ov.activator.mwfm.component.sbsm. SBSGetBeansNode</Class-Name>
      <Param name="bean" value="constant:com.hp.activation.mobile.inventory.VOICEMAIL_ZTE_MAP"/>
      <Param name="module" value="constant:SmartBeanSearch"/>
      <Param name="result" value="constant:beanVoiceMailArray"/>
      <Param name="throw_excep" value="false"/>
      <Param name="index_field0" value="constant:Mdn"/>
      <Param name="index_value0" value="plataform_name"/>
   </Action>
</Process-Node>
```

9 Bulk Activations

The bulk activations protocol adapter allows customers to load a request file composed by multiple activation requests that will be loaded by SOSA. In order to do so, the operator must place a bulk activation request file in the specified *request* directory, usually accessible through FTP, and SOSA will then detect it and start the processing. Additionally, there is a complementary Solution Container application for this feature allowing users to upload, schedule and cancel bulk activation requests through the web interface. Once processing of the request file has been completed, a result file with the original requests and their corresponding responses will be placed in the *results* directory as well as the original request file will be archived. Optionally, a bulk activation report may be sent via e-mail to the specified recipients including the results file as an attachment.

9.1 Protocol Adapter Configuration

In order to use the bulk activations feature, the bulk protocol adapter must be configured in SOSA:

```
<ProtocolAdapter className="com.hp.ov.activator.mobile.sosa.protocoladapters.bulk.BulkProtocolAdapter"
name="BULK_PA">

    <Parameter name="db.pool.name" value="db_sosa_catalog"/>

    <Parameter name="bulk.archive.directory" value="/path/to/bulk/archive"/>
    <Parameter name="bulk.request.directory" value="/path/to/bulk/requests"/>
    <Parameter name="bulk.result.directory" value="/path/to/bulk/results"/>

    <Parameter name="bulk.default.username" value="ftpuser"/>
    <Parameter name="bulk.mail.from" value="sosa@yourcompany.com"/>
    <Parameter name="bulk.mail.server.authenticate" value="true"/>
    <Parameter name="bulk.mail.server.hostname" value="mail.yourcompany.com"/>
    <Parameter name="bulk.mail.server.password" value="secret"/>
    <Parameter name="bulk.mail.server.port" value="25"/>
    <Parameter name="bulk.mail.server.secure" value="false"/>
    <Parameter name="bulk.mail.server.username" value="smtpuser"/>
    <Parameter name="bulk.mail.subject" value="Bulk Activations Report"/>

    <!-- Fine tuning

    <Parameter name="bulk.disable.persistence.for.services" value="MCA,SDP,PTT"/>
    <Parameter name="bulk.enable.persistence.for.services" value="MNP,HLR"/>
    <Parameter name="bulk.max.attachment.size" value="20971520"/>
    <Parameter name="bulk.monitor.interval" value="5000"/>
    <Parameter name="bulk.override.persistence" value="false"/>
    <Parameter name="bulk.so.timeout" value="60000"/>
-->

</ProtocolAdapter>
```

This protocol adapter supports the following configuration parameters:

Parameter	Mandatory	Default value	Description
db.pool.name	Yes		Database pool name
bulk.archive.directory	Yes		Filesystem path to the archive directory where processed activation requests will be stored in the form ORIGINAL_FILENAME

			_YYYYMMDD-HHMMSS
bulk.request.directory	Yes		Filesystem path to the requests directory where activation requests must be placed for processing
bulk.result.directory	Yes		Filesystem path to the results directory where activation reports will be output in the form ORIGINAL_FILENAME_result_YYYYMMDD-HHMMSS
bulk.default.username	No	bulk	The default username that will be assigned to request files not originated from the Solution Container application
bulk.mail.from	Yes		Sender e-mail address for activation reports
bulk.mail.server.authenticate	No	False	Whether to use authentication for connecting with the SMTP server
bulk.mail.server.hostname	No		SMTP server hostname
bulk.mail.server.password	No		SMTP server password
bulk.mail.server.port	No	25	SMTP server port
bulk.mail.server.secure	No	False	Whether to establish a secure (SSL/TLS) connection to the SMTP server
bulk.mail.server.username	No		SMTP server username
bulk.mail.subject	No	Bulk Activations Report	Message subject to be used in mailed activation reports
bulk.disable.persistence.for.services	No		Comma-separated list of service names for which SOSA persistence will be forcefully disabled when processing requests, no matter what is specified in the service catalog
bulk.enable.persistence.for.services	No		Comma-separated list of service names for which SOSA persistence will be forcefully enabled when processing requests, no matter what is specified in the service catalog
bulk.max.attachment.size	No		Maximum filesize (in bytes) for attached result files in mailed activation reports
bulk.monitor.interval	No	60000	Interval of time (in ms) between checks for new activation requests
bulk.override.persistence	No		Can be “enable” or “disable”. If specified, persistence will be forcefully enabled or disabled no matter what is specified in the service catalog for orders originated from the protocol adapter

9.2 Request File Format

Bulk activation request files are ASCII text files that must adhere to the following structure:

```
BEGIN
<request parameters>
COMMANDS
<activation requests>
END
```

9.2.1 Request Parameters

Request parameters allow configuring several aspects of the bulk activations request and may be specified in the *BEGIN* section of the request file in the form

```
PARAMETER_NAME:PARAMETER_VALUE;
```

The following request parameters are supported:

Parameter	Default value	Description
MAIL_ADDRESS		Comma-separated list of e-mail addresses where the activation report will be e-mailed on completion. If unspecified, no e-mail will be sent
MAIL_TEXT	Bulk activation job finished successfully.	A custom text for the mailed activation report
MAX_PARALLELISM	1	The maximum number of parallel threads used to process requests of this file. If this number is 1 (default value), requests will be processed sequentially, otherwise it is not guaranteed
RETRY		An expression in the form *C,N,I* allowing to specify that in case a request in the file fails with error code *C*, *N* retries should be attempted with a delay of *I* seconds between them. There may be as many RETRY parameter declarations as necessary.

9.2.2 Activation Requests

Activation requests must be placed in the *COMMANDS* section of the request file and must conform to the SCLP (SOSA Command-Line Protocol) specification:

```
SERVICE_ACTION_NAME.SERVICE_NAME.OPERATION PARAMS PARAM1=VALUE1, PARAM2=VALUE2, … PARAMn=VALUEn;
```

For example:

```
MOBILE.VOICEMAIL.CREATE PARAMS MSISDN=180561234567, IMSI=310150123456789, LANGUAGE=2, PASSWORD=secret;
```

In case there are no input parameters the PARAMS part of the command must be omitted:

```
SERVICE_ACTION_NAME.SERVICE_NAME.OPERATION;
```

9.3 Response File Format

At the time the bulk protocol adapter starts processing an input request file, the corresponding results file will be created in the *results* directory. The contents of this file are essentially the same as the corresponding input file, with the only difference of the responses being appended at the end of request command lines in the COMMANDS section. Command-response lines in the results file are added upon order finishing, so the ordering might differ from the input file if there is more than one processing thread. In case SOSA unexpectedly quits during the processing of a bulk request file, the corresponding incomplete results file will be used to resume the job without re-processing the already completed requests.

Responses adhere to the SCLP specification:

```
RESPONSE SOSA_CODE=SOSA_CODE, CODE=CODE PARAMS PARAM1=VALUE1, PARAM2=VALUE2, … PARAMn=VALUEn;
```

For example:

```
RESPONSE SOSA_CODE=0, CODE=0 PARAMS N=7;
```

In case there is an error which prevents the request from even being processed as a SOSA order, such as a bad SCLP command syntax or that the specified catalog service order or action does not exist, and so there is no SOSA code to be output, a response like the following is expected:

```
RESPONSE ERROR ERROR_MESSAGE;
```

For instance:

```
RESPONSE ERROR “Failed to start synchronous dynamic order: Could not find a suitable service order or
action in the catalog for NAME=FOO, TYPE=BAR, ACTION=BAZ”;
```

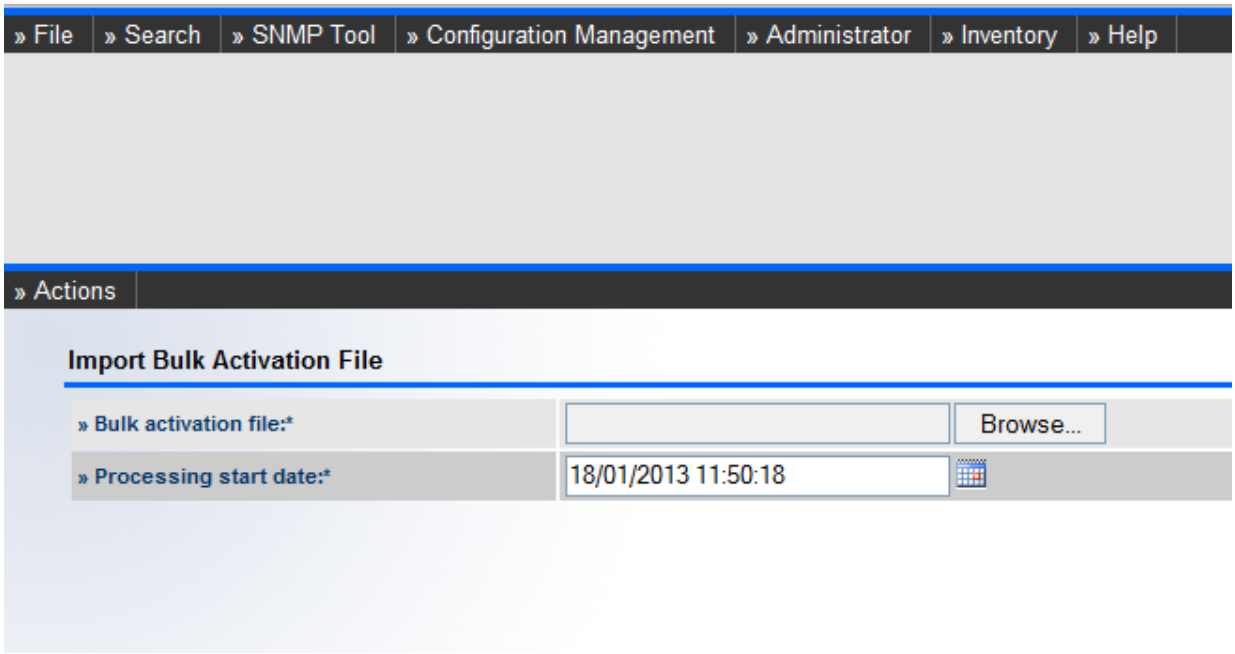
9.4 Web Interface

Apart from directly placing request files in the configured request directory, bulk activation request files may be uploaded through a Solution Container web interface. Opting for this approach also allows the operator to schedule the time at which the bulk activation will begin.

9.4.1 Importing a Bulk Activation Request File

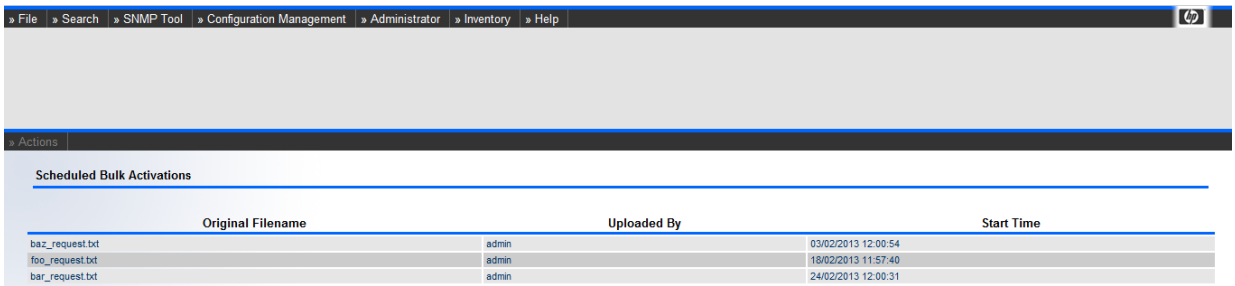
In order to import a bulk activation file, log into Solution Container and select *MSA* → *Bulk Activations* → *Import*.

You will then be presented with the Import Bulk Activation File form, where you can select the desired request file from your computer and may optionally schedule the processing start date. By default, the request will be processed as soon as it is uploaded. In order to confirm the operation click on *Actions* → *Import*.

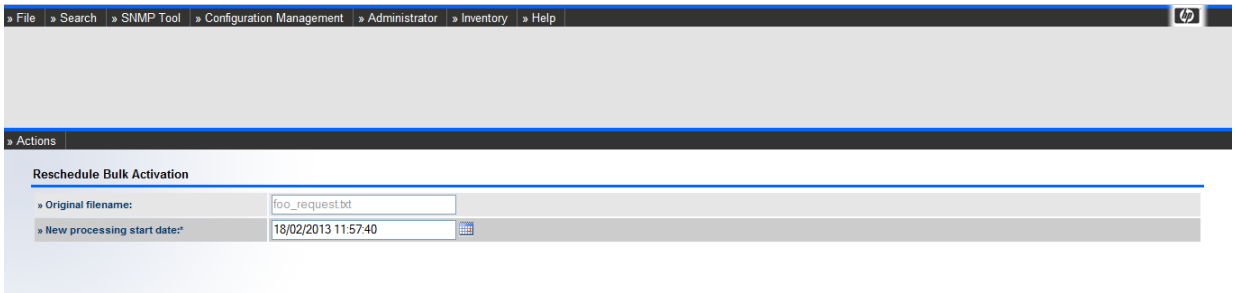


9.4.2 Listing Pending Bulk Activations

In order to list pending bulk activations select *MSA → Bulk Activations → List*. You will then be presented with the Scheduled Bulk Activations listing:



From here you can cancel or reschedule pending activations by selecting a row from the listing and clicking the appropriate option from the *Actions* menu.



Appendix

Voice Mail WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions targetNamespace="vms" xmlns:tns="vms"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:ns1="http://mail.dto.op.web.ngmail.zte.com"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc11="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soapenc12="http://www.w3.org/2003/05/soap-encoding"
xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="vms" elementFormDefault="qualified"
    attributeFormDefault="qualified">
      <xsd:element name="CreateUser">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="BoxNumber" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="OperID" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="OperPwd" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="ServiceClass" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="Language" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="Password" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="EntryAuthen" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="MessageNotify" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="UserName" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="Email" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="ZipCode" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="IDNumber" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="MWNType" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="MWNDest" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
            <xsd:element name="LDOperatorCode" type="xsd:string"
            nillable="true" minOccurs="1" maxOccurs="1" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="CreateUserResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="out" type="xsd:int" minOccurs="1"
            maxOccurs="1" />
          </xsd:sequence>
        </xsd:complexType>
```

```
    </xsd:element>
    <xsd:element name="DeleteUser">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="BoxNumber" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="OperID" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="OperPwd" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="DeleteUserResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="out" type="xsd:int" minOccurs="1"
          maxOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="QueryUser">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="BoxNumber" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="OperID" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="OperPwd" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="QueryUserResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="out"
          type="ns1:UserInfoWebServiceInfo" nillable="true"
          minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="ModifyUser">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="BoxNumber" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="OperID" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="OperPwd" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="UserStatus" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="ServiceClass" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="Language" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="Password" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="EntryAuthen" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
          <xsd:element name="MessageNotify" type="xsd:string"
          nillable="true" minOccurs="1" maxOccurs="1" />
```

```
        <xsd:element name="UserName" type="xsd:string"
        nillable="true" minOccurs="1" maxOccurs="1" />
        <xsd:element name="Email" type="xsd:string"
        nillable="true" minOccurs="1" maxOccurs="1" />
        <xsd:element name="ZipCode" type="xsd:string"
        nillable="true" minOccurs="1" maxOccurs="1" />
        <xsd:element name="IDNumber" type="xsd:string"
        nillable="true" minOccurs="1" maxOccurs="1" />
        <xsd:element name="MWNType" type="xsd:string"
        nillable="true" minOccurs="1" maxOccurs="1" />
        <xsd:element name="MWNDest" type="xsd:string"
        nillable="true" minOccurs="1" maxOccurs="1" />
        <xsd:element name="LDOperatorCode" type="xsd:string"
        nillable="true" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="ModifyUserResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="out" type="xsd:int" minOccurs="1"
        maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://mail.dto.op.web.ngmail.zte.com"
elementFormDefault="qualified"
attributeFormDefault="qualified">
  <xsd:complexType name="UserInfoWebServiceInfo">
    <xsd:sequence>
      <xsd:element name="IDNumber" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="LDOperatorCode" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="MWNDest" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="MWNType" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="email" type="xsd:string" minOccurs="0"
      nillable="true" />
      <xsd:element name="entryAuthen" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="language" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="messageNotify" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="password" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="resultcode" type="xsd:int"
      minOccurs="0" />
      <xsd:element name="serviceClass" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="userName" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="userStatus" type="xsd:string"
      minOccurs="0" nillable="true" />
      <xsd:element name="zipCode" type="xsd:string"
      minOccurs="0" nillable="true" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

```
</wsdl:types>
<wsdl:message name="CreateUserRequest">
  <wsdl:part name="parameters" element="tns:CreateUser" />
</wsdl:message>
<wsdl:message name="ModifyUserResponse">
  <wsdl:part name="parameters"
  element="tns:ModifyUserResponse" />
</wsdl:message>
<wsdl:message name="QueryUserResponse">
  <wsdl:part name="parameters" element="tns:QueryUserResponse" />
</wsdl:message>
<wsdl:message name="ModifyUserRequest">
  <wsdl:part name="parameters" element="tns:ModifyUser" />
</wsdl:message>
<wsdl:message name="CreateUserResponse">
  <wsdl:part name="parameters"
  element="tns:CreateUserResponse" />
</wsdl:message>
<wsdl:message name="QueryUserRequest">
  <wsdl:part name="parameters" element="tns:QueryUser" />
</wsdl:message>
<wsdl:message name="DeleteUserRequest">
  <wsdl:part name="parameters" element="tns:DeleteUser" />
</wsdl:message>
<wsdl:message name="DeleteUserResponse">
  <wsdl:part name="parameters"
  element="tns:DeleteUserResponse" />
</wsdl:message>
<wsdl:portType name="IvmsbossPortType">
  <wsdl:operation name="CreateUser">
    <wsdl:input name="CreateUserRequest"
    message="tns:CreateUserRequest" />
    <wsdl:output name="CreateUserResponse"
    message="tns:CreateUserResponse" />
  </wsdl:operation>
  <wsdl:operation name="DeleteUser">
    <wsdl:input name="DeleteUserRequest"
    message="tns:DeleteUserRequest" />
    <wsdl:output name="DeleteUserResponse"
    message="tns:DeleteUserResponse" />
  </wsdl:operation>
  <wsdl:operation name="QueryUser">
    <wsdl:input name="QueryUserRequest"
    message="tns:QueryUserRequest" />
    <wsdl:output name="QueryUserResponse"
    message="tns:QueryUserResponse" />
  </wsdl:operation>
  <wsdl:operation name="ModifyUser">
    <wsdl:input name="ModifyUserRequest"
    message="tns:ModifyUserRequest" />
    <wsdl:output name="ModifyUserResponse"
    message="tns:ModifyUserResponse" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="IvmsbossHttpBinding"
type="tns:IvmsbossPortType">
  <wsdlsoap:binding style="document"
  transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="CreateUser">
    <wsdlsoap:operation soapAction="" />
    <wsdl:input name="CreateUserRequest">
      <wsdlsoap:body use="literal" />
    </wsdl:input>
```

```
      <wsdl:output name="CreateUserResponse">
        <wsdlsoap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="DeleteUser">
      <wsdlsoap:operation soapAction="" />
      <wsdl:input name="DeleteUserRequest">
        <wsdlsoap:body use="literal" />
      </wsdl:input>
      <wsdl:output name="DeleteUserResponse">
        <wsdlsoap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="QueryUser">
      <wsdlsoap:operation soapAction="" />
      <wsdl:input name="QueryUserRequest">
        <wsdlsoap:body use="literal" />
      </wsdl:input>
      <wsdl:output name="QueryUserResponse">
        <wsdlsoap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ModifyUser">
      <wsdlsoap:operation soapAction="" />
      <wsdl:input name="ModifyUserRequest">
        <wsdlsoap:body use="literal" />
      </wsdl:input>
      <wsdl:output name="ModifyUserResponse">
        <wsdlsoap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Ivmsboss">
    <wsdl:port name="IvmsbossHttpPort"
    binding="tns:IvmsbossHttpBinding">
      <wsdlsoap:address location="http://10.130.48.146:88/vmsboss/Ivmsboss" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```