Mobicents USSD Gateway

User Guide

Amit Bhayani `<amit.bhayani (at) gmail.com>`
Bartosz Baranowski `<baranowb (at) gmail.com>`

Mobicents USSD Gateway: User Guide

by Amit Bhayani and Bartosz Baranowski

Abstract

This is guide to USSD Gateway application. It also introduces shortly notion of USSD services.

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the **cat my_next_bestselling_novel** command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press **Enter** to execute the command.
>
> Press **Ctrl**+**Alt**+**F1** to switch to the first virtual terminal. Press **Ctrl**+**Alt**+**F7** to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose System > Preferences > Mouse from the main menu bar to launch Mouse Preferences. In the Buttons tab, click the Left-handed mouse check box and click Close to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a gedit file, choose Applications > Accessories > Character Map from the main menu bar. Next, choose Search > Find… from the Character Map menu bar, type the name of the character in the Search field and click Next. The character you sought will be highlighted in the Character Table. Double-click this highlighted character to place it in the Text to copy field and then click the Copy button. Now switch back to your document and choose Edit > Paste from the gedit menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the > shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select Mouse from the Preferences sub-menu in the System menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or *`Proportional Bold Italic`*

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *`username@domain.name`*** at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *`file-system`*** command remounts the named file system. For example, to remount the `/home` file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

1.2. Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

2. Provide feedback to the authors!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in the the Issue Tracker [http://code.google.com/p/ussdgateway/issues/list], against the product Mobicents JAIN SLEE USSD Gateway Application, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: USSDGateway_Application_User_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

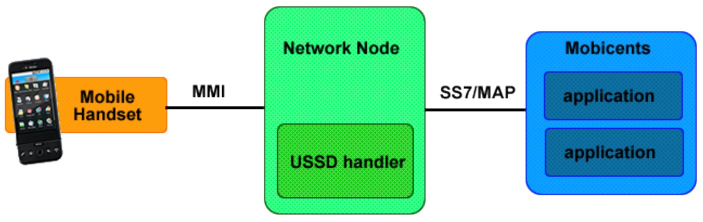
Chapter 1. Introduction to Mobicents JAIN SLEE USSD Gateway Application

USSD stands for Unstructured Supplementary Service Data what is a capability of GSM mobile phone much like the Short Message Service (SMS). But there is a difference between USSD and SMS handling.

SMS uses `store and forward` method of message delivery. Short Message is delivered first to Sender's Short Message Service Center (SMSc) which will try to deliver the message to recipient. So SMS does not guarantee that message will be delivered instantly.

USSD information is sent from mobile handset directly to application platform handling service. So USSD suppose to establish a real time session between mobile handset and application handling the service. The concept of real time session is very useful for constructing an interactive menu driven application.

A user who is dialing USSD service number initiates dialog with USSD handling application deployed on the Mobicents Platform as depicted on the figure below. The "Network Node" depicted could be MSC, HLR or VLR. The Mobicents Platform integrates with "Network Node" using MAP protocol.



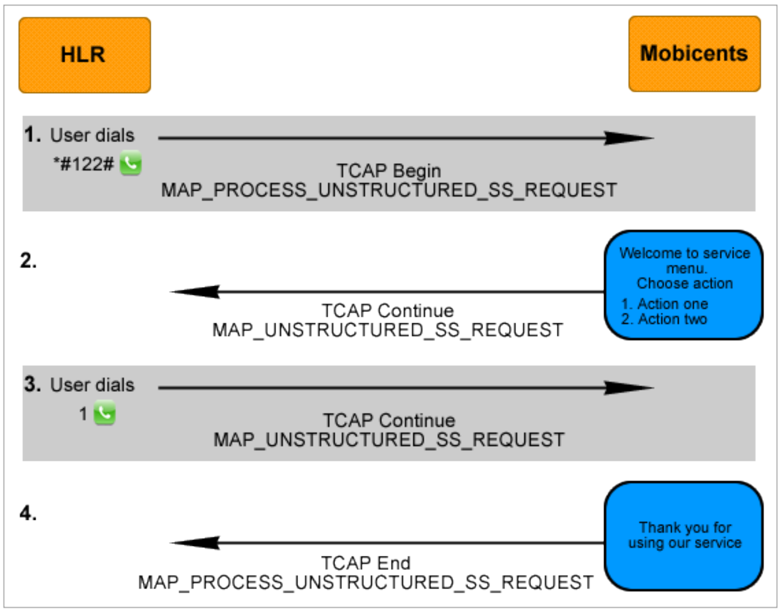
General interworking diagram

The detailed description of the allowed MMIs or phone number which user can dial is presented in `3GPP TS 22.090`. In the user's home network the following number range is defined for USSD services: *`1, 2 or 3 digits from the set (*, #) followed by 1X(Y), where X=any number 0-4, Y=any number 0-9, then, optionally "*" followed by any number of any characters, and concluding with # SEND`*

For example user can dial *#122# to reach a specific USSD service which is deployed in the home network. The application in its order can reply with menu.

One of the biggest benefits is that this service is always available even when user is currently in roaming.

Below diagram depicts typical MAP message flow for implementing data transfer between "Network Node" and Mobicents platform to implement menu driven application. For more information on mobile- (and network-) initiated USSD operations and the use of MAP USSD services, refer to [3GPPTS 24.090] in the References section.


HLR
Mobicents
1. User dials *#122#
TCAP Begin
MAP_PROCESS_UNSTRUCTURED_SS_REQUEST
2.
Welcome to service menu. Choose action 1. Action one 2. Action two
TCAP Continue
MAP_UNSTRUCTURED_SS_REQUEST
3. User dials 1
TCAP Continue
MAP_UNSTRUCTURED_SS_REQUEST
4.
Thank you for using our service
TCAP End
MAP_PROCESS_UNSTRUCTURED_SS_REQUEST

Message flow

Mobile initiated USSD service starts when user dials USSD string *#122#.

- The Network sends TCAP Begin message with Component MAP_PROCESS_UNSTRUCTURED_SS_REQUEST to the Mobicents platform. The Mobicents platform invokes USSD application logic .
- Application request additional information from user (action one or action two) via MAP_UNSTRUCTURED_SS_REQUEST encapsulated in TCAP Continue message. At this time TCAP Dialogue starts.
- Application receives user's selection of the action.
- Application performs its logic and sends a response back to the user. At this time application do not want to get additional information from the user and it sends response using MAP_PROCESS_UNSTRUCTURED_SS_REQUEST and terminates TCAP dialogue.

1.1. USSD Gateway

Existing MSC, VLR, and HLR network elements are proprietary and run on non-standard operating environments located in trusted operator's zones that make it difficult to build and deploy new applications. Also, these network elements do not provide the tools and interfaces needed to access and retrieve data from content providers over Internet. The USSD Gateway connects to the MSC, VLR, or HLR and enables the flow of USSD messages to be extended to an open, standards-based application server located in the IP network. The AS also provides the tools and interfaces to enable access to the content providers through the Internet.

Mobicents implementation of USSD Gateway is first and only open source USSD Gateway available as of today. The Mobicents USSD Gateway makes use of HTTP and SMPP* protocol between gateway and Value Added Service Modules or third party applications. Mobicents USSD Gateway receives the USSD request from subscriber handset/device via GSM Signaling network, these requests are translated to SIP or SMPP* depending on the rules set by the user and then routed to corresponding Value Added Service (VAS) or 3rd party application.

JBoss Drools is used to derive the protocol between Gateway and USSD Application and also the information of the server (for example IP, port etc) where these applications are deployed.

SMPP

SMPP protocol is in roadmap and will be implemented in next release

Chapter 2. Setup

2.1. Pre-Install Requirements and Prerequisites

Ensure that the following requirements have been met before continuing with the install.

2.1.1. Hardware Requirements

The Application doesn't change the Mobicents JAIN SLEE Hardware Requirements, refer to Mobicents JAIN SLEE documentation for more information.

Note

Note that application makes use of Resource Adaptors - this implies that RAs requirements must be taken into consideration!

Also be aware that each Resource Adaptor may have some specific hardware requirements!

2.1.2. Software Prerequisites

The Application requires Mobicents JAIN SLEE properly set, with:

- HTTP Client
- MAP

Resource Adaptors deployed.

Note

Note MAP Resource Adaptor - has some specific software requirements! Please refer to MAP RA document in JSLEE Guide

2.2. Mobicents JAIN SLEE USSD Gateway Source Code

2.2.1. Release Source Code Building

1. **Downloading the source code**

   Important

   GIT is used to manage its source code. Instructions for using GIT, including install, can be found at http://git-scm.com/documentation

   Use GIT to clone repository, the base URL is https://code.google.com/p/ussdgateway/, then to checkout specific release version(tag) use **git checkout tag_name**, lets consider release-1.0.0-SNAPSHOT.

```
[usr]$ git clone https://code.google.com/p/ussdgateway/
[usr]$ cd ussdgateway
[usr]$ git checkout release-1.0.0-SNAPSHOT
```

2. **Building the source code**

   Important

   Maven 2.0.9 (or higher) is used to build the release. Instructions for using Maven2, including install, can be found at http://maven.apache.org

   Use Maven to build the binary.

```
        [usr]$ cd ussdgateway-1.0.0-SNAPSHOT
        [usr]$ mvn install
```

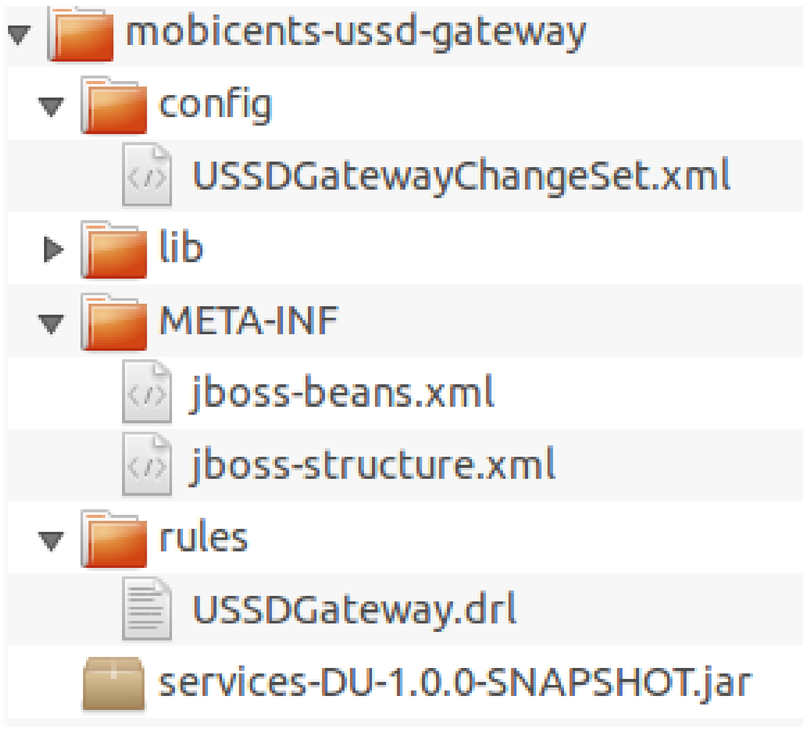
   Once the process finishes you should have the `ussdgateway-1.0.0-SNAPSHOT/core/bootstrap/target/mobicents-ussd-gateway` directory, if Mobicents JAIN SLEE is installed and environment variable JBOSS_HOME is pointing to its underlying JBoss Application Server directory, then the `mobicents-ussd-gateway` will also be deployed in the container.

2.2.2. Development Trunk Source Building

Similar process as for Section 2.2.1, “Release Source Code Building”, the only change is don't switch to specific tag.

2.3. Folder structure of Mobicents JAIN SLEE USSD Gateway

Installing Mobicents USSD Gateway creates a mobicents-ussd-gateway directory that contains gateway configuration, libraries required for boot and running, example rules definition file (.drl) etc. You need to know your way around the distribution layout to locate the drools file's to add new rules. The figure "view of Mobicens USSD Gateway" illustrates the installation directory of the Gateway.



Mobicents USSD Gateway

2.4. Rule engine configuration

Important

USSD Gateway uses `Drools` as rule engine to perform decisions, it is important to understand JBoss Drools [http://downloads.jboss.com/drools/docs/5.0.1.26597.FINAL/drools-expert/html_single/]

Engine is fed with `DRL` files having reference to fact. `DRL` file contains set of rules which perform operations on facts passed into engine. USSD Gateway `DRL` file defines rules to match initial USSD string to set of values identifying protocol and address of peer to which messages should be forwarded.

Fact is simple POJO class. USSD Gateway fact looks like

```
package org.mobicents.ussdgateway.rules;

import java.io.Serializable;

/**
 * Acts as Fact for Rules
 *
 */
public class Call implements Serializable {
    // Initial string, its like #123*
    private String ussdString;

    private boolean isHttp;
    private boolean isSmpp;

    // to be used with other protocols
    private String genericUrl;

    public Call(String ussdString) {
        this.ussdString = ussdString;
    }

    public String getUssdString() {
        return ussdString;
    }

    public boolean isHttp() {
        return isHttp;
    }

    public void setHttp(boolean isHttp) {
        this.isHttp = isHttp;
    }

    public boolean isSmpp() {
        return isSmpp;
    }

    public void setSmpp(boolean isSmpp) {
        this.isSmpp = isSmpp;
    }

    /**
     * @return the genericUrl
     */
    public String getGenericUrl() {
        return genericUrl;
    }

    /**
     * @param genericUrl
     *            the genericUrl to set
     */
    public void setGenericUrl(String genericUrl) {
        this.genericUrl = genericUrl;
    }
    @Override
    public String toString() {
       return "Call [ussdString=" + ussdString + ", isHttp=" + isHttp + ", isSmpp=" + isSmpp + ",
 genericUrl="
                  + genericUrl + "]";
    }

}
```

Rule engine can be fed with static `.drl` file or use `Guvnor` to dynamically create and maintain `.drl`

Rule engine (`Drools`) is configured with `USSDGatewayChangeSet.xml` file. Its content alters how rule set is loaded and maintained within engine. There are two ways of maintaining rules:

locally
: rules are loaded from designated file as explained in Section 2.5, “Local file configuration”. Configuration file should look as follows:

```
<change-set xmlns='http://drools.org/drools-5.0/change-set'
    xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'
    xs:schemaLocation='http://drools.org/drools-5.0/change-set.xsd'>
    <add>

        <resource
                     source='file:/home/vic/mobicents-jainslee-2.7.0.FINAL-jboss-5.1.0.GA/
jboss-5.1.0.GA/server/default
                /deploy/mobicents-ussd-gateway/rules/'
            type='DRL' />
    </add>
</change-set>
```

points to subdirectory in current application which is scanned for rule files.

remotely
: rules are managed by `Guvnor`. Guvnor configuration is explpained in Section 2.6, “Guvnor configuration” Configuration file should look as follows:

```
<change-set xmlns='http://drools.org/drools-5.0/change-set'
    xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'
    xs:schemaLocation='http://drools.org/drools-5.0/change-set.xsd'>
    <add>

        <resource source='http://localhost:8080/drools-guvnor/
            org.drools.guvnor.Guvnor/package/ussdGateway/LATEST.drl' type='DRL' />
    </add>
```

```
</change-set>
```

points to `Guvnors` latest rule file. Note that path after package `MUST` match your custom created package inside `Guvnor` .

2.5. Local file configuration

Rule file name is `USSDGateway.drl`. File content looks as follows:

```
package org.mobicents.ussdgateway.rules

import org.mobicents.ussdgateway.rules.Call;

rule "USSDGateway1"

    when
        $c : Call( ussdString == "*123#" )
    then
        $c.setHttp( true );
        $c.setGenericUrl( "http://localhost:8080/ussddemo/test" );

end
```

import of fact POJO
definition of rule
condition to enter rule clause. It accesses fact property `ussdString` and matches it against `#123*` value, if it matches engine jumps to `then` part
rule part which sets defined HTTP peer as destination for messages
end of rule `USSDGateway1` rule

The folder `rules` is scanned every 60 seconds and if any changes made to `USSDGateway.drl` or new `.drl` file added, engine will automatically deploy changed/new file and re-create the Knowledge Base

2.6. Guvnor configuration

Important

USSD Gateway Application uses `Gunvor` to manage system wide rule set in consistent way, it is important to understand Guvnor [http://downloads.jboss.com/drools/docs/5.0.1.26597.FINAL/drools-guvnor/html_single/]

`Guvnor` is deployed along with USSD Gateway Application. To access it simply go to `http://<your server>/drools-guvnor/` . This will bring initial info screen or login screen - depends on configuration.

If you have not configured the security you can directly login without providing any user id or password.

2.6.1. Creating resources

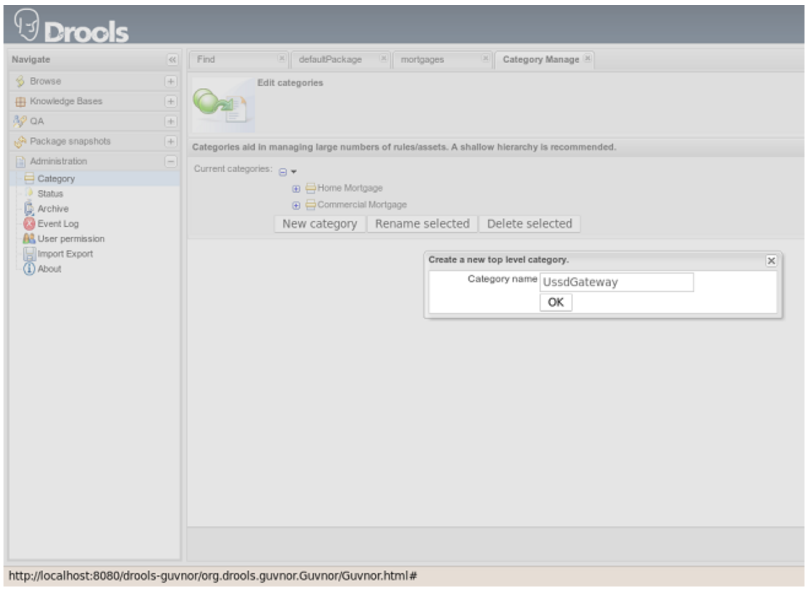
> **Note**
>
> `Guvnor` requires upload for fact model and creation of some resources before it can perform its tasks.

In case `Guvnor` has not been used(it is a new repository) you will get a message asking if you would you like to install a sample repository? Its upto you to install the sample repository. If you say yes, you would get sample repository which you can refer to have better understanding of Guvnor

Once you log-in follow the bellow steps:

1. **Create a category specific to USSD gateway.**

   Go to Administration > Category > New Category . Enter Category name as `UssdGateway` .

   Categories aid in managing large numbers of rules/assets. A shallow hierarchy is recommended.

   Create a new top level category.

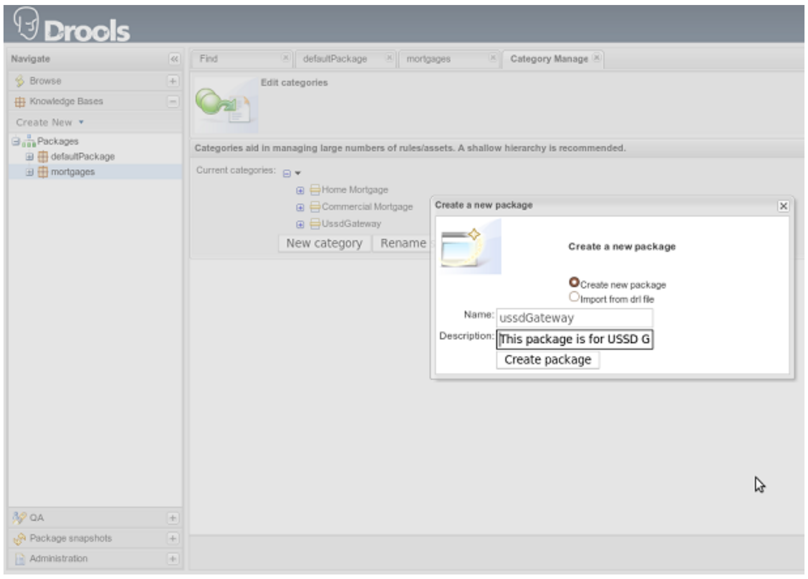
   Category name: UssdGateway

   http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/Guvnor.html#

   Guvnor category

2. **Create package for fact model.**

   Rules need a fact model (object model) to work off, so next you will want to go to the Package management feature. Go to Knowledge Bases > Create New > New Package . Type `ussdGateway` (note that this name `MUST` match package in `USSDGatewayChangeSet.xml` file).



Create a new package

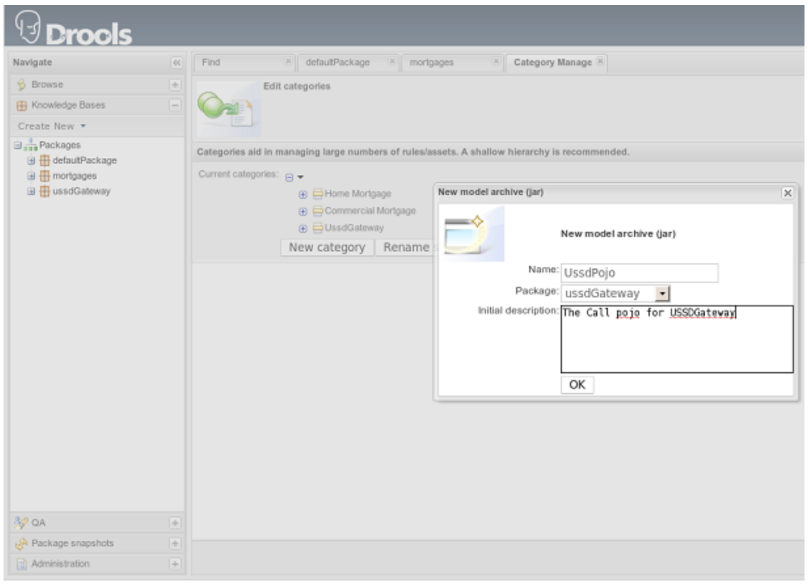
Name: ussdGateway

Description: This package is for USSD G

Guvnor package

3. **Upload fact model.**

   To upload a model, use ussdgateway-domain-x.y.z.jar which has the fact model (Call.java API) that you will be using in your rules. When you are in the model editor screen, you can upload a jar file, choose the package name from the list that you created in the previous step. Go to Knowledge Base > Create New > Upload POJO Model Jar . On the screen enter name as `UssdPojo` , select package `ussdGateway` and add the description, click Ok .



Name: UssdPojo

Package: ussdGateway

Initial description: The Call pojo for USSDGateway

Guvnor fact model upload

   Browse in newly open window and point to `${JBOSS.HOME}/server/default/deploy/mobicents-ussd-gateway/lib/ussdgateway-domain-x.y.z.jar` .

4. **Edit your package configuration.**

   Now edit your package configuration (you just created) to import the fact types you just uploaded (add import statements), and save the changes. Go to Knowledge Bases and click on `ussdGateway` package. Click on Save and validate configuration button.

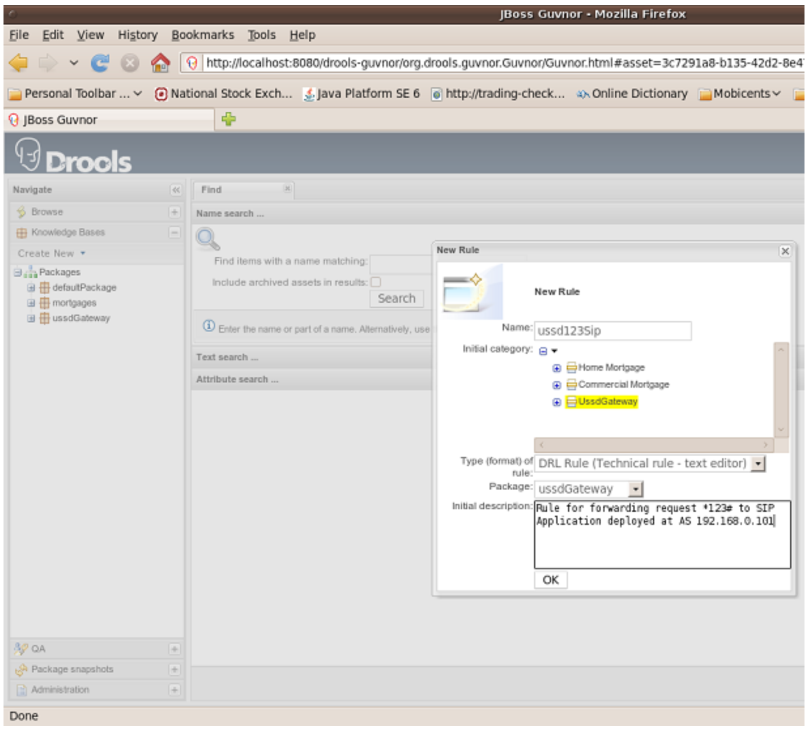
This concludes configuration of `Guvnor` . Note that this has to be done only once.

2.6.2. Creating rules

`Guvnor` allows to create rules and edit previously existing ones. Changes done with `Guvnor` are automaticly propagated to all clients. To create rule follow procedure below:

1. **Create rule.**

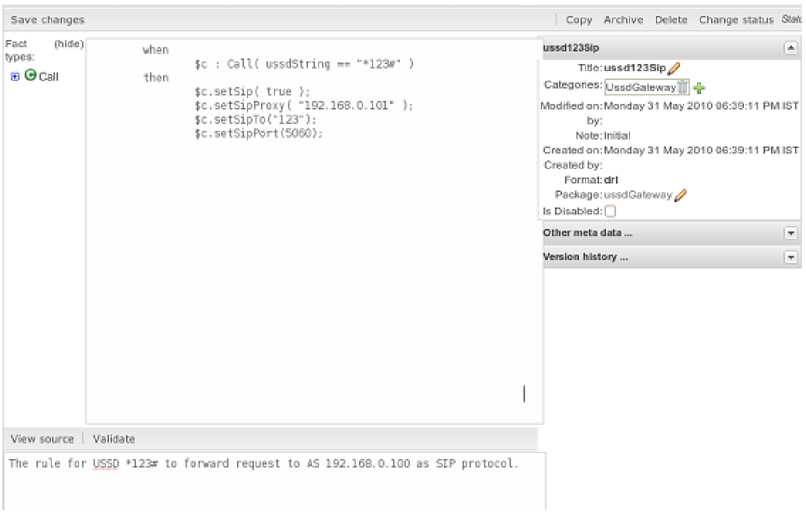
   Go to Knowledge Bases> Create New > New Rule. Enter Name as `ussd123Sip`, click on `UssdGateway` Initial Category. Select DRL Rule (Technical rule - text editor), actually you can use any editor here that you are comfortable with. Select `ussdGateway` as package. Enter description and click `Ok`.



Name: ussd123Sip
Initial category: UssdGateway
Type (format) of rule: DRL Rule (Technical rule - text editor)
Package: ussdGateway
Initial description: Rule for forwarding request *123# to SIP Application deployed at AS 192.168.0.101

Guvnor new rule

2. **Edit rule.**



```
when
        $c : Call( ussdString == "*123#" )
then
        $c.setSip( true );
        $c.setSipProxy( "192.168.0.101" );
        $c.setSipTo("123");
        $c.setSipPort(5060);
```

The rule for USSD *123# to forward request to AS 192.168.0.100 as SIP protocol.

ussd123Sip

Title: ussd123Sip
Categories: UssdGateway
Modified on: Monday 31 May 2010 06:39:11 PM IST
by:
Note: Initial
Created on: Monday 31 May 2010 06:39:11 PM IST
Created by:
Format: drl
Package: ussdGateway
Is Disabled: ☐

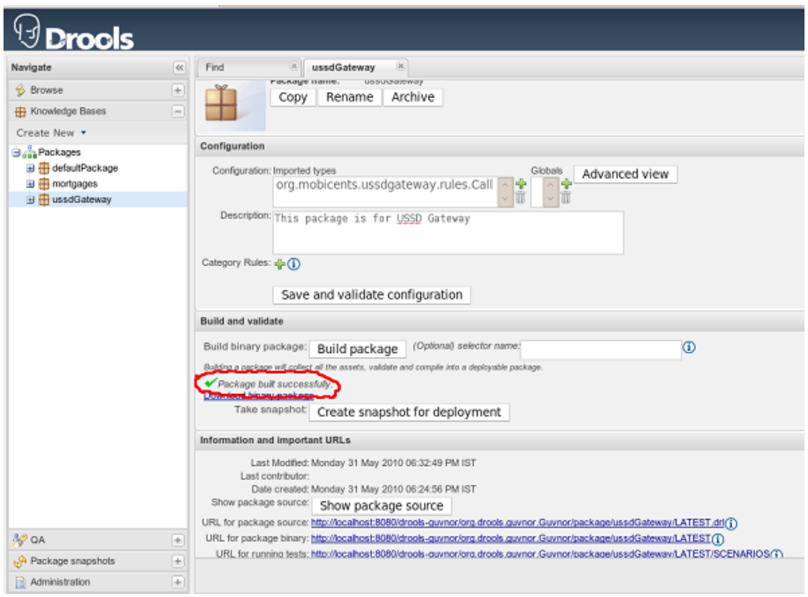
Guvnor edit rule

3. **Accept rule.**

   Click on Validate to validate the Rules you just defined. Once done with rule editing, you can check in the changes (save) by clicking on Save Changes

4. **Rebuild and validate package**

   After you have edited some rules in ussdGateway package, you can click on the ussdGateway package, open the package, and build the whole package.



Package name: ussdGateway

Configuration

Configuration: Imported types: org.mobicents.ussdgateway.rules.Call
Globals:
Description: This package is for USSD Gateway
Category Rules:

Build and validate

Build binary package: Build package (Optional) selector name:
Building a package will collect all the assets, validate and compile into a deployable package.
Package built successfully.
Download binary package
Take snapshot: Create snapshot for deployment

Information and important URLs

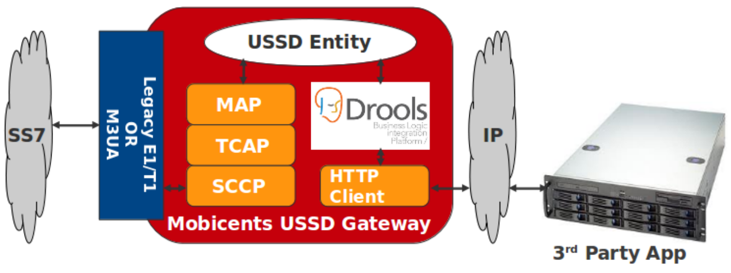
Last Modified: Monday 31 May 2010 06:32:49 PM IST
Last contributor:
Date created: Monday 31 May 2010 06:24:56 PM IST
Show package source: Show package source
URL for package source: http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/ussdGateway/LATEST.drl
URL for package binary: http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/ussdGateway/LATEST
URL for running tests: http://localhost:8080/drools-guvnor/org.drools.guvnor.Guvnor/package/ussdGateway/LATEST/SCENARIOS

Guvnor new rule

Chapter 3. Design Overview

USSD Gateway is JAIN SLEE 1.1 Application. It is capable of forwarding USSD messages to desired peer.

Following diagram depicts top design overview:



USSD Gateway Design overview

USSD Gateway provides Load Balancing and Fault Tolerance of applications. Two 3rd Party Application Servers can be paired to provide Fault-Tolerance on the Gateway and GSM Network level.

Important

Currently gateway supports following protocols for proxying:

- HTTP

Chapter 4. HTTP Transfer Mechanism

USSD Gateway supports implementation of HTTP 1.1 standards and acts as HTTP Client invoking (HTTP POST) the HTTP Application deployed on 3rd Party Application Server. The HTTP Request carries XML payload with USSD specific information.

HTTP callback makes 3rd Party Application agnostic to Operating System, Programming Language and Framework.

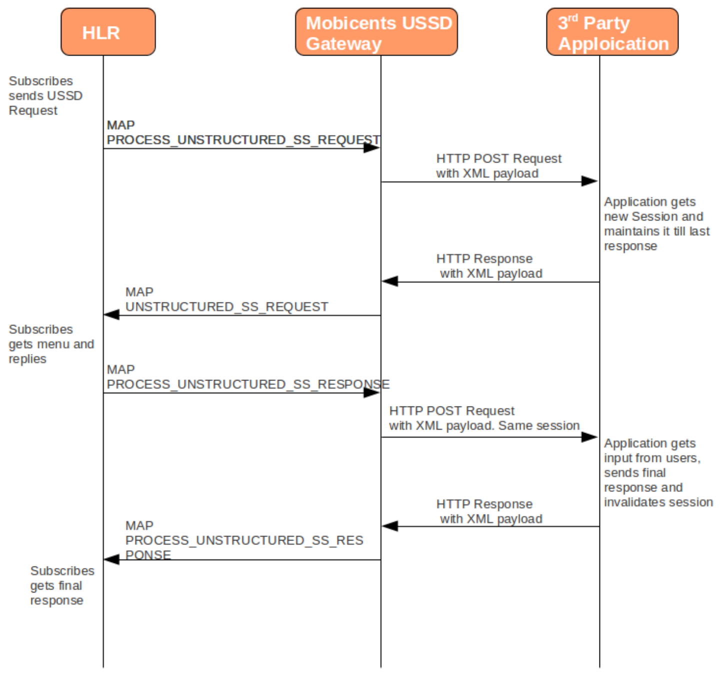
3rd Party Application can be either of following technologies on any OS

- Apache Tomcat, JBoss AS, Oracle Application Server, IBM Websphere etc for JSP/Servlet on Java
- PHP
- Microsoft IIS for ASP

HTTP errors are supported and recognized by the USSD Gateway

4.1. HTTP Message Structure

Below diagram gives example message sequence for interacting with USSD Gateway HTTP API


HLR
Mobicents USSD Gateway
3rd Party Apploication
Subscribes sends USSD Request
MAP PROCESS_UNSTRUCTURED_SS_REQUEST
HTTP POST Request with XML payload
Application gets new Session and maintains it till last response
HTTP Response with XML payload
MAP UNSTRUCTURED_SS_REQUEST
Subscribes gets menu and replies
MAP PROCESS_UNSTRUCTURED_SS_RESPONSE
HTTP POST Request with XML payload. Same session
Application gets input from users, sends final response and invalidates session
HTTP Response with XML payload
MAP PROCESS_UNSTRUCTURED_SS_RESPONSE
Subscribes gets final response

Mobicents HTTP message flow

4.1.1. HTTP payload for MAP_PROCESS_UNSTRUCTURED_SS_REQUEST

XML Payload sent to 3rd Party Application by USSD Gateway for received MAP_PROCESS_UNSTRUCTURED_SS_REQUEST will be

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog type="BEGIN" id="1234">
    <destinationReference nai="1" npi="6" number="204208300008002"/>
    <originationReference nai="1" npi="1" number="31628968300"/>
    <processUnstructuredSSRequest invokeId="0" dataCodingScheme="15" string="*234#">
        <msisdn nai="1" npi="1" number="79273605819"/>
        <alertingPattern size="1">
            <value value="6"/>
        </alertingPattern>
    </processUnstructuredSSRequest>
</dialog>
```

The XML structure is similar to actual SS7 MAP messages. `<dialog>` acts as parent tag. `type` attribute defines the state of `Dialog`. Following fours states are defined

- BEGIN : Indicates this is first message in new USSD dialog identified by attribute `id`. Through out the life of this dialog the id remains same.
- CONTINUE : Indicates this is continuing dialog
- END : Indicates this is end of dialog. Both Application as well as USSD Gateway can end dialog
- ABORT : Indicates dialog is aborted. Both Application as well as USSD Gateway can abort dialog. Abort never carries any message

`destinationReference` and `originationReference` are optional and will be included only if MAP Dialog has these values.

Next is the actual MAP message of dialog. For USSD Gateway six type of messages are defined

processUnstructuredSSRequest
: This message is always sent by USSD GW to Application as HTTP POST request. Application should always send back `processUnstructuredSSResponse` indicating that this is last message of this dialog or can also send `unstructuredSSRequest` indicating that Application expect's more response from user (menu structure)

  `<msisdn>` tag is optional and included only if actual MAP message received by USSD Gateway carries this value. This is MSISDN of user who originated this request.

> **No MSISDN**
>
> If <msisdn> is not included in `processUnstructuredSSRequest`, `originationReference` will be included in dialog and this will be the MSISDN of user originating request.

`<alertingPattern>` is also optional.

processUnstructuredSSResponse
: This message is always sent by Application to USSD GW as response to received `processUnstructuredSSRequest` or `unstructuredSSResponse`. This should always be the last message in dialog.

unstructuredSSRequest
: This message is always sent by Application to USSD GW in response to received `processUnstructuredSSRequest` or `unstructuredSSResponse`. This indicates that application is expecting some response from user.

unstructuredSSResponse
: This message is always sent by USSD GW to Application in HTTP POST request. This is response to `unstructuredSSRequest` sent by Application earlier.

unstructuredSSResponse
: unstructuredSSNotifyRequest : Not implemented yet

unstructuredSSResponse
: unstructuredSSNotifyResponse : Not implemented yet

All message type has mandatory `invokeId` attribute helping to relate the response to request. For example `processUnstructuredSSResponse` will have same `invokeId` as carried by `processUnstructuredSSRequest`. Hence if application is multi level menu, it should store `invokeId` received in `processUnstructuredSSRequest` in HTTP Session to use later. Also for every new request in same dialog, `invokeId` should be incremented by 1. For example when application sends `unstructuredSSRequest` to received `processUnstructuredSSRequest` with invokeId==0, it should set the invokeId==1 in `unstructuredSSRequest`

Attributes `dataCodingScheme` and `string` are also mandatory and represents the actual USSD Message. `dataCodingScheme` is the encoding parameter of the USSD Message.

> **USSD String length**
>
> In GSM 0902 160 octets is stated as the maximum length for the USSD string. However due to underlying signalling layers the maximum length of the USSD string depending on the message and can be less than 160

4.1.2. HTTP payload for MAP_PROCESS_UNSTRUCTURED_SS_RESPONSE

XML Payload sent to USSD Gateway by 3rd Party Application to send MAP_PROCESS_UNSTRUCTURED_SS_RESPONSE will be

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog type="END" id="1234">
    <processUnstructuredSSResponse invokeId="0" dataCodingScheme="15" string="Thank You!"/>
</dialog>
```

Dialog END

Notice that `processUnstructuredSSResponse` is last message in dialog and should always be carried in dialog type END

Important

dialog `id` and `invokeId` will be same as received in `processUnstructuredSSRequest`

4.1.3. HTTP payload for MAP_UNSTRUCTURED_SS_REQUEST

XML Payload sent to USSD Gateway by 3rd Party Application to send MAP_UNSTRUCTURED_SS_REQUEST will be

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog type="CONTINUE" id="1234">
      <unstructuredSSRequest invokeId="1" dataCodingScheme="15" string="USSD String : Hello
 World&#10; 1. Balance&#10; 2. Texts Remaining"/>
</dialog>
```

Important

dialog `id` will be same as received in `processUnstructuredSSRequest`. However `invokeId` is incremented by 1

4.1.4. HTTP payload for MAP_UNSTRUCTURED_SS_RESPONSE

XML Payload sent to 3rd Party Application by USSD Gateway for received MAP_UNSTRUCTURED_SS_RESPONSE will be

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog type="CONTINUE" id="1234">
    <unstructuredSSResponse invokeId="0" dataCodingScheme="15" string="1"/>
</dialog>
```

Chapter 5. Traces and Alarms

5.1. Tracers

USSD Gateway USSD Gateway Application creates following tracers:

Table 5.1. USSD Gateway Application Tracer and Log Categories

Sbb	Tracer name	LOG4J category
ParentSbb	USSD-Parent	javax.slee.SbbNotification[service=ServiceID[name=mobicents-ussdgateway,vendor=org.mobicents, version=1.0],sbb=SbbID[name=ParentSbb, vendor=org.mobicents,version=1.0]].USSD-Parent
SipSbb	USSD-CHILD-SipSbb	javax.slee.SbbNotification[service=ServiceID[name=mobicents-ussdgateway,vendor=org.mobicents, version=1.0],sbb=SbbID[name=SipSbb, vendor=org.mobicents,version=1.0]].USSD-CHILD-SipSbb
HttpClientSbb	USSD-CHILD-HttpClientSbb	javax.slee.SbbNotification[service=ServiceID[name=mobicents-ussdgateway,vendor=org.mobicents, version=1.0],sbb=SbbID[name=HttpClientSbb, vendor=org.mobicents,version=1.0]].USSD-CHILD-HttpClientSbb

Important

Spaces where introduced in `LOG4J category` column values, to correctly render the table. Please remove them when using copy/paste.

Appendix A. Revision History

Revision History

Revision 1.0	Wed June 2 2010	BartoszBaranowski

Creation of the Mobicents JAIN SLEE USSD Gateway Application User Guide.

Index