JBoss Communications
MAP Stack User Guide

by Amit Bhayani, Bartosz Baranowski, and Oleg Kulikov

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://bugzilla.redhat.com/bugzilla/], against the product **JBoss Communications MAP Stack** , or contact the authors.

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

Introduction to JBoss Communications MAP Stack

Important

For better understanding of this chapter please read GSM 09.02.

Mobile application part (MAP) is the protocol which is used to allow the GSM network nodes within the Network Switching Subsystem (NSS) to communicate with each other to provide services, such as roaming capability, text messaging (SMS), Unstructured Supplementary Service Data (USSD) and subscriber authentication. MAP provides an application layer on which to build the services that support a GSM network. This application layer provides a standardized set of services. MAP uses the services of the SS7 network, specifically the Signaling Connection Control Part (SCCP) and the Transaction Capabilities Application Part (TCAP)

MAP provides a number of services, which can be classified as

- Mobility services
- Location management services
- Paging and search
- Access management
- Handover services
- Authentication management
- Security management
- International mobile equipment identities management
- Subscriber management
- Identity management
- Fault recovery
- Subscriber information
- Operation and maintenance
- Subscriber tracing
- Other operation and maintenance

- Call handling
- Supplementary services–related services
- Short Message Service (SMS) management
- Network-requested Packet Data Protocol (PDP) context activation
- Location service management

The JBoss Communications implementation of MAP is only capable of Supplementary services–related services related to USSD as of today.

Within each of these service classifications, a number of messages are used to communicate between the various network entities as defined by each of the message sets. Within each message set is a predefined set of parameters. These parameters may be optional or mandatory, as defined by the standards.

1.1. Common MAP Services

Common MAP services are used with all the various services. They do not fit within any one category because they can be used in all the categories.

The messages consist of

- MAP_OPEN
- MAP_CLOSE
- MAP_DELIMITER
- MAP_U_ABORT
- MAP_P_ABORT
- MAP_NOTICE

1.2. Supplementary Services–Related Services

These services are used to manage supplementary services:

The messages consist of

- MAP_REGISTER_SS
- MAP_ERASE_SS
- MAP_ACTIVATE_SS
- MAP_DEACTIVATE_SS

- MAP_INTERROGATE_SS
- MAP_REGISTER_PASSWORD
- MAP_GET_PASSWORD
- MAP_PROCESS_UNSTRUCTURED_SS_REQUEST
- MAP_UNSTRUCTURED_SS_REQUEST
- MAP_UNSTRUCTURED_SS_NOTIFY
- MAP_SS_INVOCATION_NOTIFY
- MAP_REGISTER_CC_ENTRY
- MAP_ERASE_CC_ENTRY

Note

Of these Mobicents Impl of MAP only implements the Common MAP Services and USSD services MAP_PROCESS_UNSTRUCTURED_SS_REQUEST, MAP_UNSTRUCTURED_SS_REQUEST and MAP_UNSTRUCTURED_SS_NOTIFY

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 Stack doesn't change the JBoss Communications Hardware Requirements, however it requires SS7 card.

Note

Desired is high performance machine to process voice and data channels.

2.1.2. Software Prerequisites

The Stack depends on:

- JBoss Communications ASN library
- JBoss Communications Stream library
- JBoss Communications MTP library
- JBoss Communications SCCP library
- JBoss Communications TCAP library

2.2. JBoss Communications MAP Stack Source Code

2.2.1. Release Source Code Building

1. **Downloading the source code**

Important

Subversion is used to manage its source code. Instructions for using Subversion, including install, can be found at *http://svnbook.red-bean.com*

Use SVN to checkout a specific release source, the base URL is ?, then add the specific release version, lets consider 1.0.0.BETA2.

```
[usr]$ svn co ?/1.0.0.BETA2 isup-1.0.0.BETA2
```

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 binaries.

```
[usr]$ cd isup-1.0.0.BETA2
[usr]$ mvn install
```

Once the process finishes you should have the `binary` jar files in the `target` directory of `module`.

2.2.2. Development Trunk Source Building

Similar process as for *Section 2.2.1, "Release Source Code Building"*, the only change is the SVN source code URL, which is NOT AVAILABLE.

Design Overview

JBoss Communications MAP Stack TODO

Important

Be aware, JBoss Communications MAP Stack is subject to changes as it is under active development!

Protocol

4.1. API

JBoss Communications MAP Stack MAP builds abstraction layer over protocol definition. The MAP API is totally defined by Mobicents Media Server Team.

4.1.1. MAP Dialog overview

All interactions are handled by MAP Dialog. MAP Dialog doesn't maintain any FSM and is on top of TCAP Dialog. MAP uses the terms of ITU-T TCAP: a Begin message (with one invoke) initiates the transaction, and an End message (with a return-result or return-error) ends the transaction. It is defined by following interface:

```
package org.mobicents.protocols.ss7.map.api;

import org.mobicents.protocols.ss7.map.api.dialog.AddressString;
import org.mobicents.protocols.ss7.map.api.dialog.MAPUserAbortChoice;
import org.mobicents.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSIndication;
import org.mobicents.protocols.ss7.map.api.service.supplementary.USSDString;

public interface MAPDialog {

    /**
     * Returns this Dialog's ID. Thiis ID is actually TCAP's Dialog ID.
     * {@link org.mobicents.protocols.ss7.tcap.api.tc.dialog.Dialog}
     *
     * @return
     */
    public Long getDialogId();

    /**
     * This is equivalent of MAP User issuing the MAP_DELIMITER Service Request.
     * send() is called to explicitly request the transfer of the MAP protocol
     * data units to the peer entities.
     */
    public void send() throws MAPException;

    /**
     * This is equivalent of MAP User issuing the MAP_CLOSE Service Request.
     * This service is used for releasing a previously established MAP dialogue.
```

```
     * The service may be invoked by either MAP service-user depending on rules
     * defined within the service-user.
     *
     * <br/>
     *
     * If prearrangedEnd is false, all the Service Primitive added to MAPDialog
     * and not sent yet, will be sent to peer.
     *
     * <br/>
     *
     * If prearrangedEnd is true, all the Service Primitive added to MAPDialog
     * and not sent yet, will not be sent to peer.
     *
     * @param prearrangedEnd
     */
    public void close(boolean prearrangedEnd) throws MAPException;

    /**
     * This is equivalent to MAP User issuing the MAP_U_ABORT Service Request.
     *
     * @param userReason
     */
    public void abort(MAPUserAbortChoice mapUserAbortChoice)
            throws MAPException;

    /**
     * Add's a new Process Unstructured SS Request as Component.
     *
     * @param ussdDataCodingScheme
     *            The Data Coding Scheme for this USSD String as defined in GSM
     *            03.38
     * @param ussdString
     *            Ussd String
     * @param msisdn
     *            The MSISDN in {@link AddressString} format. This is optional
     * @throws MAPException
     */
    public void addProcessUnstructuredSSRequest(byte ussdDataCodingScheme,
            USSDString ussdString, AddressString msisdn) throws MAPException;

    /**
     * Add's a new ProcessUnstructured SS Response as Component.
     *
     * @param invokeId
```

```
     *          The original invoke ID retrieved from
     *          {@link ProcessUnstructuredSSIndication}
     * @param lastResult
     *          Specify if this Result is last - true, or there would be
     *          follow-up results - false
     * @param ussdDataCodingScheme
     *          The Data Coding Scheme for this USSD String as defined in GSM
     *          03.38
     * @param ussdString
     *          Ussd String {@link USSDString}
     * @throws MAPException
     */
    public void addProcessUnstructuredSSResponse(long invokeId,
            boolean lastResult, byte ussdDataCodingScheme, USSDString ussdString)
            throws MAPException;

    /**
     * Add's a new Unstructured SS Request
     *
     * @param ussdDataCodingScheme
     *          The Data Coding Scheme for this USSD String as defined in GSM
     *          03.38
     * @param ussdString
     *          Ussd String {@link USSDString}
     * @throws MAPException
     */
    public void addUnstructuredSSRequest(byte ussdDataCodingScheme,
            USSDString ussdString) throws MAPException;

    /**
     * Add's a new Unstructured SS Response
     *
     * @param invokeId
     *          The original invoke ID retrieved from
     *          {@link UnstructuredSSIndication}
     * @param lastResult
     *          Specify if this Result is last - true, or there would be
     *          follow-up results - false
     * @param ussdDataCodingScheme
     *          The Data Coding Scheme for this USSD String as defined in GSM
     *          03.38
     * @param ussdString
     *          Ussd String {@link USSDString}
     * @throws MAPException
```

```
     */
    public void addUnstructuredSSResponse(long invokeId, boolean lastResult,
            byte ussdDataCodingScheme, USSDString ussdString)
            throws MAPException;

}
```

4.1.2. Common MAP Services

Common MAP Services are represented by bellow interfaces

- MAP_OPEN : MAP_OPEN is represented by MAPOpenInfo Interface. The stack fires MAPOpenInfo event indicating begining of new MAP Dialog.

```
package org.mobicents.protocols.ss7.map.api.dialog;

import org.mobicents.protocols.ss7.map.api.MAPDialog;

public interface MAPOpenInfo {

    /**
     * Get the destination {@link AddressString}
     *
     * @return
     */
    public AddressString getDestReference();

    /**
     * Set the destination {@link AddressString}
     *
     * @param destReference
     */
    public void setDestReference(AddressString destReference);

    /**
     * Get the originating {@link AddressString}
     *
     * @return
```

```
     */
    public AddressString getOrigReference();

    /**
     * Set the originating {@link AddressString}
     *
     * @param origReference
     */
    public void setOrigReference(AddressString origReference);

    /**
     * Get the {@link MAPDialog} for which this event is fired
     *
     * @return
     */
    public MAPDialog getMAPDialog();

    /**
     * Set the {@link MAPDialog}
     *
     * @param mapDialog
     */
    public void setMAPDialog(MAPDialog mapDialog);

}
```

- MAP_CLOSE : MAP_CLOSE is represented by MAPCloseInfo Interface. The stack fires MAPCloseInfo event indicating end of MAP Dialog.

```
package org.mobicents.protocols.ss7.map.api.dialog;

import org.mobicents.protocols.ss7.map.api.MAPDialog;

public interface MAPCloseInfo {

    /**
     * Get the {@link MAPDialog} for which this event is fired
```

```
     *
     * @return
     */
    public MAPDialog getMAPDialog();

    /**
     * Set the {@link MAPDialog}
     *
     * @param mapDialog
     */
    public void setMAPDialog(MAPDialog mapDialog);

}
```

- MAP_DELIMITER : MAP_DELIMITER is not represented by any event and not fired from stack; rather the Service Message Event it-self is fired.
- MAP_U_ABORT : MAP_U_ABORT is represented by MAPUserAbortInfo interface. The stack fires MAPUserAbortInfo event indication that the peer application aborted the MAP Dialog

```
package org.mobicents.protocols.ss7.map.api.dialog;

import org.mobicents.protocols.ss7.map.api.MAPDialog;

public interface MAPUserAbortInfo {

    /**
     * Get the {@link MAPDialog} for which this event is fired
     *
     * @return
     */
    public MAPDialog getMAPDialog();

    /**
     * Set the {@link MAPDialog}
     *
     * @param mapDialog
```

```
     */
    public void setMAPDialog(MAPDialog mapDialog);

    /**
     * Get the {@link MAPUserAbortChoice} indicating the reason why peer aborted
     * the MAP Dialog
     *
     * @return
     */
    public MAPUserAbortChoice getMAPUserAbortChoice();

    /**
     * Set the {@link MAPUserAbortChoice}
     *
     * @param mapUsrAbrtChoice
     */
    public void setMAPUserAbortChoice(MAPUserAbortChoice mapUsrAbrtChoice);

}
```

- MAP_P_ABORT : MAP_P_ABORT is represented by MAPProviderAbortInfo. The stack fires MAPProviderAbortInfo event indicating that the stack aborted the dialog.

```
package org.mobicents.protocols.ss7.map.api.dialog;

import org.mobicents.protocols.ss7.map.api.MAPDialog;

public interface MAPProviderAbortInfo {

    /**
     * Get the {@link MAPDialog} for which this event is fired
     *
     * @return
     */
    public MAPDialog getMAPDialog();
```

```
    /**
     * Set the {@link MAPDialog}
     *
     * @param mapDialog
     */
    public void setMAPDialog(MAPDialog mapDialog);

    /**
     * Set the {@link MAPProviderAbortReason} indicating the reason why stack
     * aborted this MAP Dialog
     *
     * @param mapProvAbrtReas
     */
    public void setMAPProviderAbortReason(MAPProviderAbortReason mapProvAbrtReas);

    /**
     * get the {@link MAPProviderAbortReason} indicating the reason why stack
     * aborted this MAP Dialog
     *
     * @return
     */
    public MAPProviderAbortReason getMAPProviderAbortReason();

}
```

- MAP_NOTICE

4.1.3. Supplementary services–related services

Supplementary services–related service messages for USSD are represented by following interfaces

- MAP_PROCESS_UNSTRUCTURED_SS_REQUEST : MAP_PROCESS_UNSTRUCTURED_SS_REQUEST is represented by ProcessUnstructuredSSIndication. ProcessUnstructuredSSIndication event is fired by stack when it receives the USSD String for the first time.

```
package org.mobicents.protocols.ss7.map.api.service.supplementary;

import org.mobicents.protocols.ss7.map.api.dialog.AddressString;

public interface ProcessUnstructuredSSIndication extends USSDService {

    /**
     * Set the {@link AddressString} representing the MSISDN which sent this
     * USSD String
     *
     * @param msisdnAddr
     */
    public void setMSISDNAddressString(AddressString msisdnAddr);

    /**
     * Get the {@link AddressString} representing the MSISDN
     *
     * @return
     */
    public AddressString getMSISDNAddressString();
}
```

- MAP_UNSTRUCTURED_SS_REQUEST : MAP_UNSTRUCTURED_SS_REQUEST is represented by UnstructuredSSIndication. UnstructuredSSIndication is fired by the satck for consequent USSD String received.

```
package org.mobicents.protocols.ss7.map.api.service.supplementary;

public interface UnstructuredSSIndication extends  USSDService {
}
```

All the MAP Service messages implements the MAPMessage Interface.

```
package org.mobicents.protocols.ss7.map.api;

public interface MAPMessage {

    public long getInvokeId();

    public void setInvokeId(long invokeId);

    public MAPDialog getMAPDialog();

    public void setMAPDialog(MAPDialog mapDialog);


}
```

All service message specific to USSD implements the USSDService Interface which it-self extends MAPMessage interface

```
package org.mobicents.protocols.ss7.map.api.service.supplementary;

import org.mobicents.protocols.ss7.map.api.MAPMessage;

public interface USSDService extends MAPMessage {

    /**
     * This parameter contains the information of the alphabet and the language
     * used for the unstructured information in an Unstructured Supplementary
     * Service Data operation. The coding of this parameter is according to the
     * Cell Broadcast Data Coding Scheme as specified in GSM 03.38.
     *
     * @return
     */
    public byte getUSSDDataCodingScheme();
```

```
    /**
     *
     * @param ussdDataCodingSch
     */
    public void setUSSDDataCodingScheme(byte ussdDataCodingSch);

    /**
     * <p>
     * This parameter contains a string of unstructured information in an
     * Unstructured Supplementary Service Data operation. The string is sent
     * either by the mobile user or the network. The contents of a string sent
     * by the MS are interpreted by the network as specified in GSM 02.90.
     * </p>
     * <br/>
     * <p>
     * USSD String is OCTET STRING (SIZE (1..160))
     * </p>
     *
     * <br/>
     *
     * <p>
     * The structure of the contents of the USSD-String is dependent -- on the
     * USSD-DataCodingScheme as described in TS GSM 03.38.
     * </p>
     *
     *
     *
     * @return
     */
    public USSDString getUSSDString();

    /**
     *
     * @param ussdString
     */
    public void setUSSDString(USSDString ussdString);

}
```

4.2. Stack

MAP is part of SS7 protocol stack. AS such it relies on TCAP which in-turn relies on SCCP as means of transport. To create MAP stack you require properly configured SCCP layer. Please refer to *???* for details and examples.

JBoss Communications MAP is defined by provider and `stack` interfaces. Interfaces are defined as follows:

```
package org.mobicents.protocols.ss7.map.api;

public interface MAPStack {

   public MAPProvider getMAPProvider();

}
```

```
package org.mobicents.protocols.ss7.map.api;

import org.mobicents.protocols.ss7.map.api.dialog.AddressString;
import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;

public interface MAPProvider {

   public static final int NETWORK_UNSTRUCTURED_SS_CONTEXT_V2 = 1;

   /**
    * Creates a new Dialog. This is equivalent to issuing MAP_OPEN Service
    * Request to MAP Provider.
    *
    * @param applicationCntx
    *           This parameter identifies the type of application context
    *           being established. If the dialogue is accepted the received
    *           application context name shall be echoed. In case of refusal
    *           of dialogue this parameter shall indicate the highest version
    *           supported.
    *
    * @param destAddress
```

```
     *          A valid SCCP address identifying the destination peer entity.
     *          As an implementation option, this parameter may also, in the
     *          indication, be implicitly associated with the service access
     *          point at which the primitive is issued.
     *
     * @param destReference
     *          This parameter is a reference which refines the identification
     *          of the called process. It may be identical to Destination
     *          address but its value is to be carried at MAP level.
     *
     * @param origAddress
     *          A valid SCCP address identifying the requestor of a MAP
     *          dialogue. As an implementation option, this parameter may
     *          also, in the request, be implicitly associated with the
     *          service access point at which the primitive is issued.
     *
     * @param origReference
     *          This parameter is a reference which refines the identification
     *          of the calling process. It may be identical to the Originating
     *          address but its value is to be carried at MAP level.
     *          Processing of the Originating-reference shall be performed
     *          according to the supplementary service descriptions and other
     *          service descriptions, e.g. operator determined barring.
     * @return
     */
    public MAPDialog createNewDialog(MAPApplicationContext appCntx,
            SccpAddress origAddress, AddressString origReference,
            SccpAddress destAddress, AddressString destReference)
            throws MAPException;

    /**
     * Add MAP Dialog listener to the Stack
     *
     * @param mapDialogListener
     */
    public void addMAPDialogListener(MAPDialogListener mapDialogListener);

    /**
     * Remove MAP DIalog Listener from the stack
     *
     * @param mapDialogListener
     */
    public void removeMAPDialogListener(MAPDialogListener mapDialogListener);
```

```
    /**
     * Add MAP Service listener to the stack
     *
     * @param mapServiceListener
     */
    public void addMAPServiceListener(MAPServiceListener mapServiceListener);

    /**
     * Remove MAP Service listener from the stack
     *
     * @param mapServiceListener
     */
    public void removeMAPServiceListener(MAPServiceListener mapServiceListener);

    /**
     * Get the {@link MapServiceFactory}
     *
     * @return
     */
    public MapServiceFactory getMapServiceFactory();

    /**
     * Get {@link MAPDialog} corresponding to passed dialogId
     *
     * @param dialogId
     * @return
     */
    public MAPDialog getMAPDialog(Long dialogId);

}
```

Provider allows user to access stack facitlities, create dialogs, register as listener for incoming common service messages and register as listener for incoming USSD service messages. Listener's declares set of callbacks methods. They are defined as follows:

```
package org.mobicents.protocols.ss7.map.api;

import org.mobicents.protocols.ss7.map.api.dialog.MAPAcceptInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPCloseInfo;
```

```
import org.mobicents.protocols.ss7.map.api.dialog.MAPOpenInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPProviderAbortInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPRefuseInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPUserAbortInfo;

public interface MAPDialogListener {

    public void onMAPOpenInfo(MAPOpenInfo mapOpenInfo);

    public void onMAPAcceptInfo(MAPAcceptInfo mapAccptInfo);

    public void onMAPCloseInfo(MAPCloseInfo mapCloseInfo);

    public void onMAPRefuseInfo(MAPRefuseInfo mapRefuseInfo);

    public void onMAPUserAbortInfo(MAPUserAbortInfo mapUserAbortInfo);

    public void onMAPProviderAbortInfo(MAPProviderAbortInfo mapProviderAbortInfo);

}
```

```
package org.mobicents.protocols.ss7.map.api;

import org.mobicents.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSIndication;
import org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSIndication;

public interface MAPServiceListener {


    public void onProcessUnstructuredSSIndication(ProcessUnstructuredSSIndication procUnstrInd);

    public void onUnstructuredSSIndication(UnstructuredSSIndication unstrInd);


}
```

4.3. Configuration

MAP layer does not require any config. However it requires properly set TCAP and its sublayers. Please refer to `TCAP User Guide` for supported configuration options.

4.3.1. Dependencies

MAP depends on following:

- MTP
- SCCP
- TCAP

4.4. Example

```
import org.mobicents.protocols.ss7.map.api.MAPApplicationContext;
import org.mobicents.protocols.ss7.map.api.MAPDialog;
import org.mobicents.protocols.ss7.map.api.MAPDialogListener;
import org.mobicents.protocols.ss7.map.api.MAPProvider;
import org.mobicents.protocols.ss7.map.api.MAPServiceListener;
import org.mobicents.protocols.ss7.map.api.MapServiceFactory;
import org.mobicents.protocols.ss7.map.api.dialog.AddressNature;
import org.mobicents.protocols.ss7.map.api.dialog.AddressString;
import org.mobicents.protocols.ss7.map.api.dialog.MAPAcceptInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPCloseInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPOpenInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPProviderAbortInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPRefuseInfo;
import org.mobicents.protocols.ss7.map.api.dialog.MAPUserAbortInfo;
import org.mobicents.protocols.ss7.map.api.dialog.NumberingPlan;
import org.mobicents.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSIndication;
import org.mobicents.protocols.ss7.map.api.service.supplementary.USSDString;
import org.mobicents.protocols.ss7.map.api.service.supplementary.UnstructuredSSIndication;
import org.mobicents.protocols.ss7.sccp.parameter.SccpAddress;

public class MAPClient implements MAPDialogListener, MAPServiceListener {

    MAPProvider mapProvider;

    MapServiceFactory servFact = mapProvider.getMapServiceFactory();
```

```
SccpAddress destAddress = null;

// The address created by passing the AddressNature, NumberingPlan and
// actual address
AddressString destReference = servFact.createAddressString(
        AddressNature.international_number, NumberingPlan.land_mobile,
        "204208300008002");

SccpAddress origAddress = null;

AddressString origReference = servFact.createAddressString(
        AddressNature.international_number, NumberingPlan.ISDN,
        "31628968300");

public void run() throws Exception {

    // First create Dialog
    MAPDialog mapDialog = mapProvider.createNewDialog(
            MAPApplicationContext.networkUnstructuredSsContextV2,
            destAddress, destReference, origAddress, origReference);

    // The dataCodingScheme is still byte, as I am not exactly getting how
    // to encode/decode this.
    byte ussdDataCodingScheme = 0x0f;

    // USSD String: *125*+31628839999#
    // The Charset is null, here we let system use default Charset (UTF-7 as
    // explained in GSM 03.38. However if MAP User wants, it can set its own
    // impl of Charset
    USSDString ussdString = servFact.createUSSDString("*125*+31628839999#");


    AddressString msisdn = this.servFact
    .createAddressString(AddressNature.international_number,
            NumberingPlan.ISDN, "31628838002");

    mapDialog.addProcessUnstructuredSSRequest(ussdDataCodingScheme,
            ussdString, msisdn);

    // This will initiate the TC-BEGIN with INVOKE component
    mapDialog.send();
}

public void onMAPAcceptInfo(MAPAcceptInfo mapAccptInfo) {
```

```
        // TODO Auto-generated method stub

    }

    public void onMAPCloseInfo(MAPCloseInfo mapCloseInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPOpenInfo(MAPOpenInfo mapOpenInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPProviderAbortInfo(MAPProviderAbortInfo mapProviderAbortInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPRefuseInfo(MAPRefuseInfo mapRefuseInfo) {
        // TODO Auto-generated method stub

    }

    public void onMAPUserAbortInfo(MAPUserAbortInfo mapUserAbortInfo) {
        // TODO Auto-generated method stub

    }

    public void onProcessUnstructuredSSIndication(
            ProcessUnstructuredSSIndication procUnstrInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSIndication(UnstructuredSSIndication unstrInd) {
        // TODO Auto-generated method stub

    }

}
```

Appendix A. Revision History

Revision History
Revision 1.0 Wed June 2 2010 AmitBhayani
Creation of the JBoss Communications MAP Stack User Guide.

Index

F