MAP

Table of Contents

Mobile application part (MAP) is the protocol that is used to allow the network nodes within the Network Switching Subsystem (NSS) to communicate with each other in order 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 network. This application layer provides a standardized set of services. uses the services of the GSM network, specifically the Signaling Connection Control Part (SCCP) and the Transaction Capabilities Application Part (TCAP).

For better understanding of this chapter you must be familiar with the specifications defined in `GSM 09.02`. JSS7 stack supports MAP versions 1, 2, 3, 4 with java interfaces for a version 3GPP TS 29.002 V10.3.0 (2011-06)

MAP has full implementation for USSD, SMS and LMS (Location Management Service) services and partial implementation for Mobility Services (updateLocation, sendAuthenticationInfo, anyTimeInterrogation, checkIMEI operations). You can find the list of implemented MAP messages here: MAPMessagesImplemented.ods. The document uses color coding to represent the status of the implementation:

Green

Fully Implemented

Red

Interface available (but not yet implemented)

Yellow

Implementation in progress

Restcomm jSS7 Any contribution to implement other messages are welcome. We will provide you with all the help that you may require initially.

jSS7 MAP

The methods required to represent the MAP Protocol Stack are defined in the interface `org.restcomm.protocols.ss7.map.api.MAPStack` which exposes `org.restcomm.protocols.ss7.map.api.MAPProvider` that interacts directly with the `MAPStack`. This interface defines the methods that will be used by any registered MAP-User application.

A MAP-User application must implement interfaces to listen for MAP messages and also implement interfaces for dialogue and component handling primitives. One such interface is `org.restcomm.protocols.ss7.map.api.MAPDialogListener` which must be implemented for listening associated with dialog events. Others are one or more interfaces for listening associated with component events based on `org.restcomm.protocols.ss7.map.api.MAPServiceListener` interface.

Below is a list of MAP Service Listener classes. A MAP-User interested in listening for messages specific to a particular MAP Service must implement the corresponding MAPServiceListener.

LMS operations

`org.restcomm.protocols.ss7.map.api.service.lsm.MAPServiceLsmListener`

Mobility operations

`org.restcomm.protocols.ss7.map.api.service.mobility.MAPServiceMobilityListener`

OAM operations

`org.restcomm.protocols.ss7.map.api.service.oam.MAPServiceOamListener`

PDP context activation operations

`org.restcomm.protocols.ss7.map.api.service.pdpContextActivation.MAPServicePdpContextActivationListener`

SMS operations

`org.restcomm.protocols.ss7.map.api.service.sms.MAPServiceSmsListener`

Supplementary operations

`org.restcomm.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementaryListener`

A MAP-User interested in all the services must implement all the above Service Listener classes.

The `org.restcomm.protocols.ss7.map.MAPStackImpl` class is a concrete implementation of `MAPStack`. The MAP-User application gets a reference to `MAPProvider` by doing a JNDI lookup as explained in [_design_overview_ss7_service].

```
String providerJndiName = "java:/restcomm/ss7/map";
this.mapProvider = ((MAPProvider) ctx.lookup(providerJndiName));
...
```

To listen for incoming MAP Dialog and MAP Primitive messages, the MAP-User application should register the concrete implementation of `MAPDialogListener` with `MAPProvider`.

To listen for incoming MAP Service messages, the MAP-User application should register the concrete implementation of `MAPServiceListener` with the corresponding `MAPServiceBase` `. The following [class]`MAPServiceBase` services are exposed by the `MAPProvider`:

Call handling service

`org.restcomm.protocols.ss7.map.api.service.callhandling.MAPServiceCallHandling`

LSM service

`org.restcomm.protocols.ss7.map.api.service.lsm.MAPServiceLsm`

Mobility service

`org.restcomm.protocols.ss7.map.api.service.mobility.MAPServiceMobility`

OAM service

`org.restcomm.protocols.ss7.map.api.service.oam.MAPServiceOam`

PDP context activation service

`org.restcomm.protocols.ss7.map.api.service.pdpContextActivation.MAPServicePdpContextActivation`

SMS service

`org.restcomm.protocols.ss7.map.api.service.sms.MAPServiceSms`

Supplementary service

`org.restcomm.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementary`

```
public class UssdClientExample implements MAPDialogListener,
MAPServiceSupplementaryListener {
        ....
        mapProvider.addMAPDialogListener(this);
        mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);
        ....
}
```

Prior to using any MAP specific service, the corresponding service should be activated as shown below:

```
        ....
        // Make the supplementary service activated
        mapProvider.getMAPServiceSupplementary().acivate();
        ....
```

The MAP-User Application leverages:

- `MAPParameterFactory` to create instances of `USSDString`, `ISDNAddressString` and many other primitives that are used by MAP services.
- `MAPSmsTpduParameterFactory` to create instances of SMS TPDU primitives used for sending SMS messages like `SmsDeliverTpdu` or `SmsSubmitTpdu`.
- `MAPErrorMessageFactory` to create instances of MAP error messages like `MAPErrorMessageSystemFailure`.

```
        MapParameterFactory paramFact = mapProvider.getMapServiceFactory();
        USSDString ussdString = paramFact.createUSSDString("*125*+31628839999#",
                null);
        ISDNAddressString msisdn = paramFact.createISDNAddressString(
                AddressNature.international_number, NumberingPlan.ISDN,
                "31628838002");
```

jSS7 Sending a MAP request (processUnstructuredSS-Request as an example)

For sending a MAP request, you must do the following at the client side:

- Create a new MAP Dialog

```
// First create Dialog
SccpAddress origAddress = createLocalAddress();
ISDNAddressString origReference = client.getMAPProvider().getMAPParameterFactory().
    createISDNAddressString(AddressNature.international_number, NumberingPlan
.land_mobile, "31628968300");
SccpAddress destAddress = createRemoteAddress();
ISDNAddressString destReference = client.getMAPProvider().getMAPParameterFactory().
    createISDNAddressString(AddressNature.international_number, NumberingPlan
.land_mobile, "204208300008002");

currentMapDialog = mapProvider.getMAPServiceSupplementary().
    createNewDialog(MAPApplicationContext.getInstance(MAPApplicationContextName
.networkUnstructuredSsContext,
    MAPApplicationContextVersion.version2), origAddress,
    destReference, remoteAddress, destReference);
```

- Add an Invoke component (processUnstructuredSS-Request message)

```
// The dataCodingScheme is still byte, as I am not exactly getting how
// to encode/decode this.
byte ussdDataCodingScheme = 0x0f;
// 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 = paramFact.createUSSDString(ussdMessage, null);
ISDNAddressString msisdn = client.getMAPProvider().getMAPParameterFactory().
createISDNAddressString(AddressNature.international_number, NumberingPlan.ISDN,
"31628838002");
currentMapDialog.addProcessUnstructuredSSRequest(ussdDataCodingScheme, ussdString,
alertingPattern, msisdn);
```

- Send a TC-Begin message to the server peer

```
currentMapDialog.send();
```

- Wait for a response from the server

At the server side, when the TC-Begin message is received, the following sequence of events occur:

```
void MAPDialogListener. onDialogRequest(MAPDialog mapDialog, AddressString
destReference,
    AddressString origReference, MAPExtensionContainer extensionContainer);
```

This is the request for MAP Dialog processing. A MAP-User can reject the Dialog by invoking the `mapDialog.refuse()` method.

This is followed by the events (one or more) corresponding to the incoming primitives. In this case it is:

```
void MAPServiceSupplementaryListener.onProcessUnstructuredSSRequest
(ProcessUnstructuredSSRequest procUnstrReqInd);
```

When processing component-dependant messages, you can add response components. In this case it is processUnstructuredSS-Response as an example:

```
        USSDString ussdString = ind.getUSSDString();
        String request = ussdString.getString();

        // processing USSD request
        String response = "Your balans is 100$";

        // The dataCodingScheme is still byte, as I am not exactly getting how
        // to encode/decode this.
        byte ussdDataCodingScheme = 0x0f;
        USSDString ussdResponse = paramFact.createUSSDString(response, null);

        try {mapDialog.addProcessUnstructuredSSResponse(ind.getInvokeId(),
ussdDataCodingScheme, ussdResponse);
        } catch (MAPException e) {// TODO Auto-generated catch
blocke.printStackTrace();
        }
```

If preparing the response takes more time, you should return the control and prepare the answer asynchronously in a separate thread.

If error or reject primitives are included in a TCAP message, the following events occur:

```
public void onErrorComponent(MAPDialog mapDialog, Long invokeId, MAPErrorMessage
mapErrorMessage);
public void onProviderErrorComponent(MAPDialog mapDialog, Long invokeId,
MAPProviderError providerError);
public void onRejectComponent(MAPDialog mapDialog, Long invokeId, Problem problem);
```

After all incoming components have been processed, the event `onDialogDelimiter(MAPDialog mapDialog);` or in the case of TC-END, `onDialogClose(MAPDialog mapDialog)` is invoked. If all response components have been prepared you can tell the stack to send the response:

- `mapDialog.close(false);` - to send TC-END
- `mapDialog.send();` - to send TC-CONTINUE
- `mapDialog.close(true);` - sends TC-END without any components (prearrangedEnd)

Instead of `send()` and `close(boolean prearrangedEnd)` methods you can invoke `sendDelayed()` or `closeDelayed(boolean prearrangedEnd)` methods. These methods are similar to `send()` and `close()` methods, but when these methods are invoked from MAP Service message handlers (component handler methods called by stack while parsing incoming components), real sending and dialog closing will occur only when all incoming component events and `onDialogDelimiter()` ` or `onDialogClose()` is processed. If all response components have been prepared you should return the control and send a response when all components are ready.

In case of an error, you can terminate a MAP dialog in any method by invoking

- `mapDialog.abort(mapUserAbortChoice);` - sends TC-U-ABORT primitive

If there are no local actions or there is no response from a peer for a long time, a timeout occurs and the following methods are invoked:

- `MAPDialogListener.onDialogTimeout(MAPDialog mapDialog);`
- `MAPServiceListener.onInvokeTimeout(MAPDialog mapDialog, Long invokeId);`

In the `onDialogTimeout()` method you can invoke `mapDialog.keepAlive();` to prevent a Dialog from closing. For preventing an Invoke timeout you should invoke `resetInvokeTimer(Long invokeId);` before `onInvokeTimeout()` occurs.

jSS7 MAP Usage

```
package org.restcomm.protocols.ss7.map;

import javax.naming.InitialContext;
import javax.naming.NamingException;

import org.restcomm.protocols.ss7.map.api.MAPApplicationContext;
import org.restcomm.protocols.ss7.map.api.MAPApplicationContextName;
import org.restcomm.protocols.ss7.map.api.MAPApplicationContextVersion;
import org.restcomm.protocols.ss7.map.api.MAPDialog;
import org.restcomm.protocols.ss7.map.api.MAPDialogListener;
import org.restcomm.protocols.ss7.map.api.MAPException;
import org.restcomm.protocols.ss7.map.api.MAPMessage;
import org.restcomm.protocols.ss7.map.api.MAPParameterFactory;
import org.restcomm.protocols.ss7.map.api.MAPProvider;
import org.restcomm.protocols.ss7.map.api.datacoding.CBSDataCodingScheme;
import org.restcomm.protocols.ss7.map.api.dialog.MAPAbortProviderReason;
```

```
import org.restcomm.protocols.ss7.map.api.dialog.MAPAbortSource;
import org.restcomm.protocols.ss7.map.api.dialog.MAPNoticeProblemDiagnostic;
import org.restcomm.protocols.ss7.map.api.dialog.MAPRefuseReason;
import org.restcomm.protocols.ss7.map.api.dialog.MAPUserAbortChoice;
import org.restcomm.protocols.ss7.map.api.errors.MAPErrorMessage;
import org.restcomm.protocols.ss7.map.api.primitives.AddressString;
import org.restcomm.protocols.ss7.map.api.primitives.AlertingPattern;
import org.restcomm.protocols.ss7.map.api.primitives.IMSI;
import org.restcomm.protocols.ss7.map.api.primitives.ISDNAddressString;
import org.restcomm.protocols.ss7.map.api.primitives.MAPExtensionContainer;
import org.restcomm.protocols.ss7.map.api.primitives.USSDString;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.MAPDialogSupplementary;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementaryListen
er;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSRequest;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSResponse
;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyRequest;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyResponse;
import org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSRequest;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSResponse;
import org.restcomm.protocols.ss7.map.datacoding.CBSDataCodingSchemeImpl;
import org.restcomm.protocols.ss7.sccp.parameter.SccpAddress;
import org.restcomm.protocols.ss7.tcap.asn.ApplicationContextName;
import org.restcomm.protocols.ss7.tcap.asn.comp.Problem;

/**
 * A simple example show-casing how to use MAP stack. Demonstrates how new MAP
 * Dialog is craeted and Invoke is sent to peer.
 *
 * @author Amit Bhayani
 *
 */
public class UssdClientExample implements MAPDialogListener,
MAPServiceSupplementaryListener {

    private MAPProvider mapProvider;
    private MAPParameterFactory paramFact;

    public UssdClientExample() throws NamingException {
        InitialContext ctx = new InitialContext();
        try {String providerJndiName = "java:/restcomm/ss7/map";this.mapProvider =
((MAPProvider) ctx.lookup(providerJndiName));
        } finally {ctx.close();
```

```
        }
    }

    public MAPProvider getMAPProvider() {
        return mapProvider;
    }

    public void start() {
        // Listen for Dialog events
        mapProvider.addMAPDialogListener(this);
        // Listen for USSD related messages
        mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);

        // Make the supplementary service activated
        mapProvider.getMAPServiceSupplementary().acivate();
    }

    public void stop() {
        mapProvider.getMAPServiceSupplementary().deactivate();
    }

    public void sendProcessUssdRequest(SccpAddress origAddress, AddressString
origReference, SccpAddress remoteAddress,AddressString destReference, String
ussdMessage, AlertingPattern alertingPattern, ISDNAddressString msisdn)throws
MAPException {
        // First create Dialog
        MAPDialogSupplementary currentMapDialog = mapProvider
.getMAPServiceSupplementary().createNewDialog( MAPApplicationContext.getInstance
(MAPApplicationContextName.networkUnstructuredSsContext,
MAPApplicationContextVersion.version2), origAddress, destReference, remoteAddress,
destReference);

        CBSDataCodingScheme ussdDataCodingScheme = new CBSDataCodingSchemeImpl(0x0f);
        // 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 = paramFact.createUSSDString(ussdMessage, null, null);

        currentMapDialog.addProcessUnstructuredSSRequest(ussdDataCodingScheme,
ussdString, alertingPattern, msisdn);
        // This will initiate the TC-BEGIN with INVOKE component
        currentMapDialog.send();
    }

    public void onProcessUnstructuredSSResponse(ProcessUnstructuredSSResponse ind) {
        USSDString ussdString = ind.getUSSDString();
        try {String response = ussdString.getString(null);// processing USSD response
        } catch (MAPException e) {// TODO Auto-generated catch
blocke.printStackTrace();
        }
    }
```

```
    public void onErrorComponent(MAPDialog mapDialog, Long invokeId, MAPErrorMessage
mapErrorMessage) {
        // TODO Auto-generated method stub

    }

    public void onRejectComponent(MAPDialog mapDialog, Long invokeId, Problem problem,
boolean isLocalOriginated) {
        // TODO Auto-generated method stub

    }

    public void onInvokeTimeout(MAPDialog mapDialog, Long invokeId) {
        // TODO Auto-generated method stub

    }

    public void onMAPMessage(MAPMessage mapMessage) {
        // TODO Auto-generated method stub

    }

    public void onProcessUnstructuredSSRequest(ProcessUnstructuredSSRequest
procUnstrReqInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSRequest(UnstructuredSSRequest unstrReqInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSResponse(UnstructuredSSResponse unstrResInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyRequest(UnstructuredSSNotifyRequest
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyResponse(UnstructuredSSNotifyResponse
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }
```

```
    public void onDialogDelimiter(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequest(MAPDialog mapDialog, AddressString destReference,
AddressString origReference,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequestEricsson(MAPDialog mapDialog, AddressString
destReference, AddressString origReference,IMSI eriImsi, AddressString eriVlrNo) {
        // TODO Auto-generated method stub

    }

    public void onDialogAccept(MAPDialog mapDialog, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogUserAbort(MAPDialog mapDialog, MAPUserAbortChoice userReason
,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogProviderAbort(MAPDialog mapDialog, MAPAbortProviderReason
abortProviderReason,MAPAbortSource abortSource, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogClose(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }

    public void onDialogNotice(MAPDialog mapDialog, MAPNoticeProblemDiagnostic
noticeProblemDiagnostic) {
        // TODO Auto-generated method stub

    }

    public void onDialogRelease(MAPDialog mapDialog) {
    }

    public void onDialogTimeout(MAPDialog mapDialog) {
```

```
        // TODO Auto-generated method stub

    }

    @Override
    public void onDialogReject(MAPDialog mapDialog, MAPRefuseReason refuseReason
,ApplicationContextName alternativeApplicationContext, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }
}
```

```
package org.restcomm.protocols.ss7.map;

import javax.naming.InitialContext;
import javax.naming.NamingException;

import org.restcomm.protocols.ss7.map.api.MAPDialog;
import org.restcomm.protocols.ss7.map.api.MAPDialogListener;
import org.restcomm.protocols.ss7.map.api.MAPException;
import org.restcomm.protocols.ss7.map.api.MAPMessage;
import org.restcomm.protocols.ss7.map.api.MAPParameterFactory;
import org.restcomm.protocols.ss7.map.api.MAPProvider;
import org.restcomm.protocols.ss7.map.api.datacoding.CBSDataCodingScheme;
import org.restcomm.protocols.ss7.map.api.dialog.MAPAbortProviderReason;
import org.restcomm.protocols.ss7.map.api.dialog.MAPAbortSource;
import org.restcomm.protocols.ss7.map.api.dialog.MAPNoticeProblemDiagnostic;
import org.restcomm.protocols.ss7.map.api.dialog.MAPRefuseReason;
import org.restcomm.protocols.ss7.map.api.dialog.MAPUserAbortChoice;
import org.restcomm.protocols.ss7.map.api.errors.MAPErrorMessage;
import org.restcomm.protocols.ss7.map.api.primitives.AddressString;
import org.restcomm.protocols.ss7.map.api.primitives.IMSI;
import org.restcomm.protocols.ss7.map.api.primitives.MAPExtensionContainer;
import org.restcomm.protocols.ss7.map.api.primitives.USSDString;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.MAPDialogSupplementary;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.MAPServiceSupplementaryListen
er;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSRequest;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.ProcessUnstructuredSSResponse
;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyRequest;
import
org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSNotifyResponse;
import org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSRequest;
```

```
import
org.restcomm.protocols.ss7.map.api.service.supplementary.UnstructuredSSResponse;
import org.restcomm.protocols.ss7.map.datacoding.CBSDataCodingSchemeImpl;
import org.restcomm.protocols.ss7.tcap.asn.ApplicationContextName;
import org.restcomm.protocols.ss7.tcap.asn.comp.Problem;

/**
 * A simple example show-casing how to use MAP stack. Demonstrates how to listen
 * to incoming Dialog from peer and process the MAP messages and send response.
 *
 * @author Amit Bhayani
 *
 */
public class UssdServerExample implements MAPDialogListener,
MAPServiceSupplementaryListener {

    private MAPProvider mapProvider;
    private MAPParameterFactory paramFact;

    public UssdServerExample() throws NamingException {
        InitialContext ctx = new InitialContext();
        try {String providerJndiName = "java:/restcomm/ss7/map";this.mapProvider =
((MAPProvider) ctx.lookup(providerJndiName));
        } finally {ctx.close();
        }
    }

    public MAPProvider getMAPProvider() {
        return mapProvider;
    }

    public void start() {
        // Listen for Dialog events
        mapProvider.addMAPDialogListener(this);
        // Listen for USSD related messages
        mapProvider.getMAPServiceSupplementary().addMAPServiceListener(this);

        // Make the supplementary service activated
        mapProvider.getMAPServiceSupplementary().acivate();
    }

    public void stop() {
        mapProvider.getMAPServiceSupplementary().deactivate();
    }

    public void onProcessUnstructuredSSRequest(ProcessUnstructuredSSRequest ind) {

        USSDString ussdString = ind.getUSSDString();
        MAPDialogSupplementary currentMapDialog = ind.getMAPDialog();

        try {String request = ussdString.getString(null);
```

```
// processing USSD requestString response = "Your balans is 100$";
CBSDataCodingScheme ussdDataCodingScheme = new CBSDataCodingSchemeImpl(0x0f
);USSDString ussdResponse = paramFact.createUSSDString(response, null, null);
currentMapDialog.addProcessUnstructuredSSResponse(ind.getInvokeId(),
ussdDataCodingScheme, ussdResponse);
        } catch (MAPException e1) {// TODO Auto-generated catch
blocke1.printStackTrace();
        }
    }

    public void onDialogDelimiter(MAPDialog mapDialog) {
        // This will initiate the TC-END with ReturnResultLast component
        try {mapDialog.send();
        } catch (MAPException e) {// TODO Auto-generated catch
blocke.printStackTrace();
        }
    }

    public void onErrorComponent(MAPDialog mapDialog, Long invokeId, MAPErrorMessage
mapErrorMessage) {
        // TODO Auto-generated method stub

    }

    public void onRejectComponent(MAPDialog mapDialog, Long invokeId, Problem problem,
boolean isLocalOriginated) {
        // TODO Auto-generated method stub

    }

    public void onInvokeTimeout(MAPDialog mapDialog, Long invokeId) {
        // TODO Auto-generated method stub

    }

    public void onMAPMessage(MAPMessage mapMessage) {
        // TODO Auto-generated method stub

    }

    public void onProcessUnstructuredSSResponse(ProcessUnstructuredSSResponse ind) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSRequest(UnstructuredSSRequest unstrReqInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSResponse(UnstructuredSSResponse unstrResInd) {
```

```
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyRequest(UnstructuredSSNotifyRequest
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }

    public void onUnstructuredSSNotifyResponse(UnstructuredSSNotifyResponse
unstrNotifyInd) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequest(MAPDialog mapDialog, AddressString destReference,
AddressString origReference,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogRequestEricsson(MAPDialog mapDialog, AddressString
destReference, AddressString origReference,IMSI eriImsi, AddressString eriVlrNo) {
        // TODO Auto-generated method stub

    }

    public void onDialogAccept(MAPDialog mapDialog, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogReject(MAPDialog mapDialog, MAPRefuseReason refuseReason
,ApplicationContextName alternativeApplicationContext, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogUserAbort(MAPDialog mapDialog, MAPUserAbortChoice userReason
,MAPExtensionContainer extensionContainer) {
        // TODO Auto-generated method stub

    }

    public void onDialogProviderAbort(MAPDialog mapDialog, MAPAbortProviderReason
abortProviderReason,MAPAbortSource abortSource, MAPExtensionContainer
extensionContainer) {
        // TODO Auto-generated method stub
```

```
    }

    public void onDialogClose(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }

    public void onDialogNotice(MAPDialog mapDialog, MAPNoticeProblemDiagnostic
noticeProblemDiagnostic) {
        // TODO Auto-generated method stub

    }

    public void onDialogRelease(MAPDialog mapDialog) {
    }

    public void onDialogTimeout(MAPDialog mapDialog) {
        // TODO Auto-generated method stub

    }
}
```