XML Parser API

Table of Contents

1. Maven dependency 2
2. Simple example - Tree based Menu 4
3. EventsSerializeFactory 8
4. Dialog 10

Restcomm USSD GATEWAY exposes easy to use Java framework to serialize and deserialize the MAP messages. So if you are using Java Servlets for your business logic, this framework will make it very easy to compose your business logic. However using this framework is not mandatory and you are free to create your own XML parser.

Chapter 1. Maven dependency

If you are using maven to build your project define following dependencies to use XML Framework in your project

```
        <dependency>
            <groupId>org.mobicents.ussd</groupId>
            <artifactId>xml</artifactId>
            <version>6.1.5.GA</version>
        </dependency>
```

Note

Mobicents maintains the archieve in private repository. please add the following in your pom's repositories tag

```
        <repository>
          <id>mobicents-releases-repository</id>
          <name>Mobicents Releases Repository</name>
          <url>
http://telestax.artifactoryonline.com/telestax/releases</url>
          <releases>
            <enabled>true</enabled>
            <updatePolicy>never</updatePolicy>
          </releases>
          <snapshots>
            <enabled>false</enabled>
            <updatePolicy>never</updatePolicy>
          </snapshots>
        </repository>
        <repository>
          <id>mobicents-snapshots-repository</id>
          <name>Mobicents Snapshots Repository</name>

<url>http://telestax.artifactoryonline.com/telestax/snapshots</url>
          <releases>
            <enabled>false</enabled>
            <updatePolicy>never</updatePolicy>
          </releases>
          <snapshots>
            <enabled>true</enabled>
            <updatePolicy>never</updatePolicy>
          </snapshots>
        </repository>
```

Important

Mobicents Releases and Snapshots Repository are password protected. Ask support to get your password.

Once you have user name and password add following to your servers definition in your maven settings located in M2_HOME/conf/settings.xml

```
<server>
  <id>mobicents-releases-repository</id>
  <username>xxx</username>
  <password>xxx</password>
</server>
<server>
  <id>mobicents-snapshots-repository</id>
  <username>xxx</username>
  <password>xxx</password>
</server>
```

Chapter 2. Simple example - Tree based Menu

Below is the simple `HttpServlet` example to create a tree based menu structure. The flow of message is same as shown in Restcomm HTTP message flow.

```
    public class TestServlet extends HttpServlet {

    private static final Logger logger = Logger.getLogger(TestServlet.class);

    private EventsSerializeFactory factory = null;

    @Override
    public void init() {
            factory = new EventsSerializeFactory();
    }

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException {

    }

    @Override
    public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException {
        ServletInputStream is = request.getInputStream();
        try {
            Dialog original = factory.deserialize(is);
            HttpSession session = request.getSession(true);
            if (logger.isInfoEnabled()) {
                logger.info("doPost. HttpSession=" + session.getId() + " Dialog = " +
original);
            }

            USSDString ussdStr = null;
            byte[] data = null;
            final FastList<MAPMessage> capMessages = original.getMAPMessages();

            MessageType messageType = original.getTCAPMessageType();

            // This is initial request, if its not NTFY, we need session
            for (FastList.Node<MAPMessage> n = capMessages.head(), end = capMessages
                    .tail(); (n = n.getNext()) != end;) {
                final MAPMessage rawMessage = n.getValue();
                final MAPMessageType type = rawMessage.getMessageType();

                switch (messageType) {
```

```
                    case Begin:
                    switch (mapMessage.getMessageType()) {
                        case processUnstructuredSSRequest_Request:
                            ProcessUnstructuredSSRequest processUnstructuredSSRequest
= (ProcessUnstructuredSSRequest) mapMessage;
                            CBSDataCodingScheme cbsDataCodingScheme =
processUnstructuredSSRequest
                                .getDataCodingScheme();
                            if (logger.isInfoEnabled()) {
                                logger.info("Received
ProcessUnstructuredSSRequestIndication USSD String="
                                        + processUnstructuredSSRequest.
getUSSDString().getString());
                            }

                            session.setAttribute(
                                "ProcessUnstructuredSSRequest_InvokeId",
                                processUnstructuredSSRequest.getInvokeId());

                            //You business logic here and finally send back response

                            //Urdu
                            //CBSDataCodingScheme cbsDataCodingSchemeUrdu = new
CBSDataCodingSchemeImpl(72);
                            //ussdStr = new USSDStringImpl("\u062C\u0645\u064A\u0639
\u0627\u0644\u0645\u0633\u062A\u0639\u0645\u0644\u064A\u0646
\u0627\u0644\u0622\u062E\u0631\u064A\u0646 \u062A\u0645
\u0625\u0636\u0627\u0641\u062A\u0647\u0645",
                            //      cbsDataCodingSchemeUrdu, null);
                            //UnstructuredSSRequest unstructuredSSRequestIndication =
new UnstructuredSSRequestImpl(
                            //      cbsDataCodingSchemeUrdu, ussdStr, null, null);

                            //English

                            ussdStr = new USSDStringImpl(
                                "USSD String : Hello World\n 1. Balance\n 2. Texts
Remaining",
                                cbsDataCodingScheme, null);
                            UnstructuredSSRequest unstructuredSSRequestIndication =
new UnstructuredSSRequestImpl(
                                cbsDataCodingScheme, ussdStr, null, null);

                            original.reset();
                            original.setTCAPMessageType(MessageType.Continue);
                            original.addMAPMessage(unstructuredSSRequestIndication);

                            data = factory.serialize(copy);

                            response.getOutputStream().write(data);
                            response.flushBuffer();
```

```
                            break;
                        default:
                            // This is error. If its begin it should be only Process
                            // Unstructured SS Request
                            logger.error("Received Dialog BEGIN but message is not
ProcessUnstructuredSSRequestIndication. Message="
                                    + mapMessage);
                            break;
                        }

                    break;
                    case Continue:
                    switch (type) {
                    case unstructuredSSRequest_Response:
                        UnstructuredSSResponse unstructuredSSResponse =
(UnstructuredSSResponseImpl) rawMessage;

                        CBSDataCodingScheme cbsDataCodingScheme =
unstructuredSSResponse
                                .getDataCodingScheme();

                        long invokeId = (Long) session
                                .getAttribute(
"ProcessUnstructuredSSRequest_InvokeId");

                        USSDString ussdStringObj = unstructuredSSResponse
                                .getUSSDString();
                        String ussdString = null;
                        if (ussdStringObj != null) {
                            ussdString = ussdStringObj.getString(null);
                        }

                        logger.info("Received UnstructuredSSResponse USSD String="
                                + ussdString
                                + " HttpSession="
                                + session.getId() + " invokeId=" + invokeId);

                        cbsDataCodingScheme = new CBSDataCodingSchemeImpl(0x0f);
                        ussdStr = new USSDStringImpl("Thank You!", null, null);
                        ProcessUnstructuredSSResponse processUnstructuredSSResponse =
new ProcessUnstructuredSSResponseImpl(
                                cbsDataCodingScheme, ussdStr);
                        processUnstructuredSSResponse.setInvokeId(invokeId);

                        original.reset();
                        original.setTCAPMessageType(MessageType.End);
                        original.addMAPMessage(processUnstructuredSSResponse);
                        original.close(false);

                        data = factory.serialize(original);
```

```
                        response.getOutputStream().write(data);
                        response.flushBuffer();

                        try {
                            session.invalidate();
                        } catch (Exception e) {
                            session.invalidate();
                            logger.error("Error while invalidating HttpSession="
                                    + session.getId());
                        }
                        break;
                    default:
                        // This is error. If its begin it should be only Process
                        // Unstructured SS Request
                        logger.error("Received Dialog CONTINUE but message is not
UnstructuredSSResponseIndication. Message="
                                + rawMessage);
                        break;
                    }

                    break;

                case ABORT:
                    // The Dialog is aborted, lets do cleaning here

                    try {
                        session.invalidate();
                    } catch (Exception e) {
                        session.invalidate();
                        logger.error("Error while invalidating HttpSession=" +
session.getId());
                    }
                    break;
            }

        } catch (XMLStreamException e) {
            logger.error("Error while processing received XML", e);
        }

    }
}
```

Chapter 3. EventsSerializeFactory

This section provides the details for `EventsSerializeFactory`

```
public class EventsSerializeFactory {

    private static final String DIALOG = "dialog";
    private static final String TYPE = "type";
    private static final String TAB = "\t";

    final XMLBinding binding = new XMLBinding();

    public EventsSerializeFactory() {
        binding.setAlias(Dialog.class, DIALOG);
        binding.setClassAttribute(TYPE);
    }

    /**
     * Serialize passed {@link Dialog} object
     *
     * @param dialog
     * @return serialized byte array
     * @throws XMLStreamException
     *             Exception if serialization fails
     */
    public byte[] serialize(Dialog dialog) throws XMLStreamException {

        final ByteArrayOutputStream baos = new ByteArrayOutputStream();
        final XMLObjectWriter writer = XMLObjectWriter.newInstance(baos);

        try {

            writer.setBinding(binding);
            writer.setIndentation(TAB);

            writer.write(dialog, DIALOG, Dialog.class);
            writer.flush();
            byte[] data = baos.toByteArray();

            return data;
        } finally {
            writer.close();
        }
    }

    /**
     * De-serialize the byte[] into {@link Dialog} object
     *
     * @param data
     * @return de-serialized Dialog Object
```

```
     * @throws XMLStreamException
     *             Exception if de-serialization fails
     */
    public Dialog deserialize(byte[] data) throws XMLStreamException {
        final ByteArrayInputStream bais = new ByteArrayInputStream(data);
        final XMLObjectReader reader = XMLObjectReader.newInstance(bais);
        try {
            Dialog dialog = reader.read(DIALOG, Dialog.class);
            return dialog;
        } finally {
            reader.close();
        }
    }

    /**
     * De-serialize passed {@link InputStream} into {@link Dialog} object
     *
     * @param is
     * @return de-serialized Dialog Object
     * @throws XMLStreamException
     *             Exception if de-serialization fails
     */
    public Dialog deserialize(InputStream is) throws XMLStreamException {
        final XMLObjectReader reader = XMLObjectReader.newInstance(is);
        try {
            Dialog dialog = reader.read(DIALOG, Dialog.class);
            return dialog;
        } finally {
            reader.close();
        }
    }
}
```

- The `serialize` method serializes Dialog and retruns back byte array.
- The `deserialize` is overloaded method. Application can either pass `byte[]` or `InputStream` and de-serializes the stream of data to Dialog object.

Chapter 4. Dialog

This section provides the details for `XmlMAPDialog`

```
public class XmlMAPDialog implements Serializable {

    .....
    .....

    public XmlMAPDialog() {
        super();
    }

    /**
     *
     */
    public XmlMAPDialog(MAPApplicationContext appCntx, SccpAddress localAddress,
SccpAddress remoteAddress,
            Long localId, Long remoteId, AddressString destReference, AddressString
origReference) {
        this.appCntx = appCntx;
        this.localAddress = localAddress;
        this.remoteAddress = remoteAddress;
        this.localId = localId;
        this.remoteId = remoteId;

        this.destReference = destReference;
        this.origReference = origReference;
    }

    @Override
    public void abort(MAPUserAbortChoice mapUserAbortChoice) throws MAPException {
        this.mapUserAbortChoice = mapUserAbortChoice;
    }

    @Override
    public void addEricssonData(IMSI arg0, AddressString arg1) {
        // TODO Auto-generated method stub

    }

    @Override
    public boolean cancelInvocation(Long arg0) throws MAPException {
        throw new MAPException(new OperationNotSupportedException());
    }

    @Override
    public void close(boolean prearrangedEnd) throws MAPException {
        this.prearrangedEnd = prearrangedEnd;
    }
```

```
    @Override
    public void closeDelayed(boolean arg0) throws MAPException {
        throw new MAPException(new OperationNotSupportedException());
    }

    @Override
    public MAPApplicationContext getApplicationContext() {
        return this.appCntx;
    }

    @Override
    public SccpAddress getLocalAddress() {
        return this.localAddress;
    }

    @Override
    public Long getLocalDialogId() {
        return this.localId;
    }

    @Override
    public int getMaxUserDataLength() {
        return 0;
    }

    @Override
    public int getMessageUserDataLengthOnClose(boolean arg0) throws MAPException {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public int getMessageUserDataLengthOnSend() throws MAPException {
        // TODO Auto-generated method stub
        return 0;
    }

    @Override
    public AddressString getReceivedDestReference() {
        return this.destReference;
    }

    @Override
    public MAPExtensionContainer getReceivedExtensionContainer() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public AddressString getReceivedOrigReference() {
```

```
        return this.origReference;
    }

    @Override
    public SccpAddress getRemoteAddress() {
        return this.remoteAddress;
    }

    @Override
    public Long getRemoteDialogId() {
        return this.remoteId;
    }

    @Override
    public boolean getReturnMessageOnError() {
        return this.returnMessageOnError;
    }

    @Override
    public MAPServiceBase getService() {
        return null;
    }

    @Override
    public MAPDialogState getState() {
        return this.state;
    }

    @Override
    public MessageType getTCAPMessageType() {
        return this.messageType;
    }

    @Override
    public Object getUserObject() {
        return this.userObject;
    }

    @Override
    public void keepAlive() {
        // TODO Auto-generated method stub

    }

    @Override
    public void processInvokeWithoutAnswer(Long invokeId) {
        this.processInvokeWithoutAnswerIds.add(invokeId);
    }

    @Override
    public void refuse(Reason refuseReason) throws MAPException {
```

```
        this.refuseReason = refuseReason;
    }

    @Override
    public void release() {
        // TODO Auto-generated method stub

    }

    @Override
    public void resetInvokeTimer(Long arg0) throws MAPException {
        throw new MAPException(new OperationNotSupportedException());
    }

    @Override
    public void send() throws MAPException {
        throw new MAPException(new OperationNotSupportedException());
    }

    @Override
    public void sendDelayed() throws MAPException {
        throw new MAPException(new OperationNotSupportedException());
    }

    @Override
    public void sendErrorComponent(Long invokeId, MAPErrorMessage mapErrorMessage)
throws MAPException {
        this.errorComponents.put(invokeId, mapErrorMessage);
    }

    @Override
    public void sendInvokeComponent(Invoke arg0) throws MAPException {
        throw new MAPException(new OperationNotSupportedException());
    }

    @Override
    public void sendRejectComponent(Long arg0, Problem arg1) throws MAPException {
        // TODO Auto-generated method stub

    }

    @Override
    public void sendReturnResultComponent(ReturnResult arg0) throws MAPException {
        // TODO Auto-generated method stub

    }

    @Override
    public void sendReturnResultLastComponent(ReturnResultLast arg0) throws
MAPException {
        throw new MAPException(new OperationNotSupportedException());
```

```
    }

    @Override
    public void setExtentionContainer(MAPExtensionContainer arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void setLocalAddress(SccpAddress origAddress) {
        this.localAddress = origAddress;
    }

    @Override
    public void setRemoteAddress(SccpAddress destAddress) {
        this.remoteAddress = destAddress;
    }

    @Override
    public void setReturnMessageOnError(boolean returnMessageOnError) {
        this.returnMessageOnError = returnMessageOnError;

    }

    @Override
    public void setUserObject(Object obj) {
        this.userObject = obj.toString();
    }

    /**
     * Non MAPDialog methods
     */

    public void addMAPMessage(MAPMessage mapMessage) {
        this.mapMessages.add(mapMessage);
    }

    public boolean removeMAPMessage(MAPMessage mapMessage) {
        return this.mapMessages.remove(mapMessage);
    }

    public FastList<MAPMessage> getMAPMessages() {
        return this.mapMessages;
    }

    public FastList<Long> getProcessInvokeWithoutAnswerIds() {
        return this.processInvokeWithoutAnswerIds;
    }

    public ErrorComponentMap<Long, MAPErrorMessage> getErrorComponents() {
```

```
        return errorComponents;
    }

    public MAPUserAbortChoice getMAPUserAbortChoice() {
        return this.mapUserAbortChoice;
    }

    public MAPAbortProviderReason getMapAbortProviderReason() {
        return mapAbortProviderReason;
    }

    public void setMapAbortProviderReason(MAPAbortProviderReason
mapAbortProviderReason) {
        this.mapAbortProviderReason = mapAbortProviderReason;
    }

    public MAPRefuseReason getMapRefuseReason() {
        return mapRefuseReason;
    }

    public void setMapRefuseReason(MAPRefuseReason mapRefuseReason) {
        this.mapRefuseReason = mapRefuseReason;
    }

    public Boolean getDialogTimedOut() {
        return dialogTimedOut;
    }

    public void setDialogTimedOut(Boolean dialogTimedOut) {
        this.dialogTimedOut = dialogTimedOut;
    }

    public Boolean getPrearrangedEnd() {
        return this.prearrangedEnd;
    }

    public void setTCAPMessageType(MessageType messageType) {
        this.messageType = messageType;
    }

    public boolean isRedirectRequest() {
        return redirectRequest;
    }

    public void setRedirectRequest(boolean redirectRequest) {
        this.redirectRequest = redirectRequest;
    }

    public void reset() {
        this.mapMessages.clear();
        this.processInvokeWithoutAnswerIds.clear();
```

```
        this.errorComponents.clear();
    }

}
```