HTTP Transfer Mechanism

Table of Contents

Restcomm USSD GATEWAY supports implementation of HTTP 1.1 standards and acts as a HTTP Client invoking the HTTP Application deployed on the third-party Application Server. The `HTTP Request/Response` carries XML payload with USSD specific information.

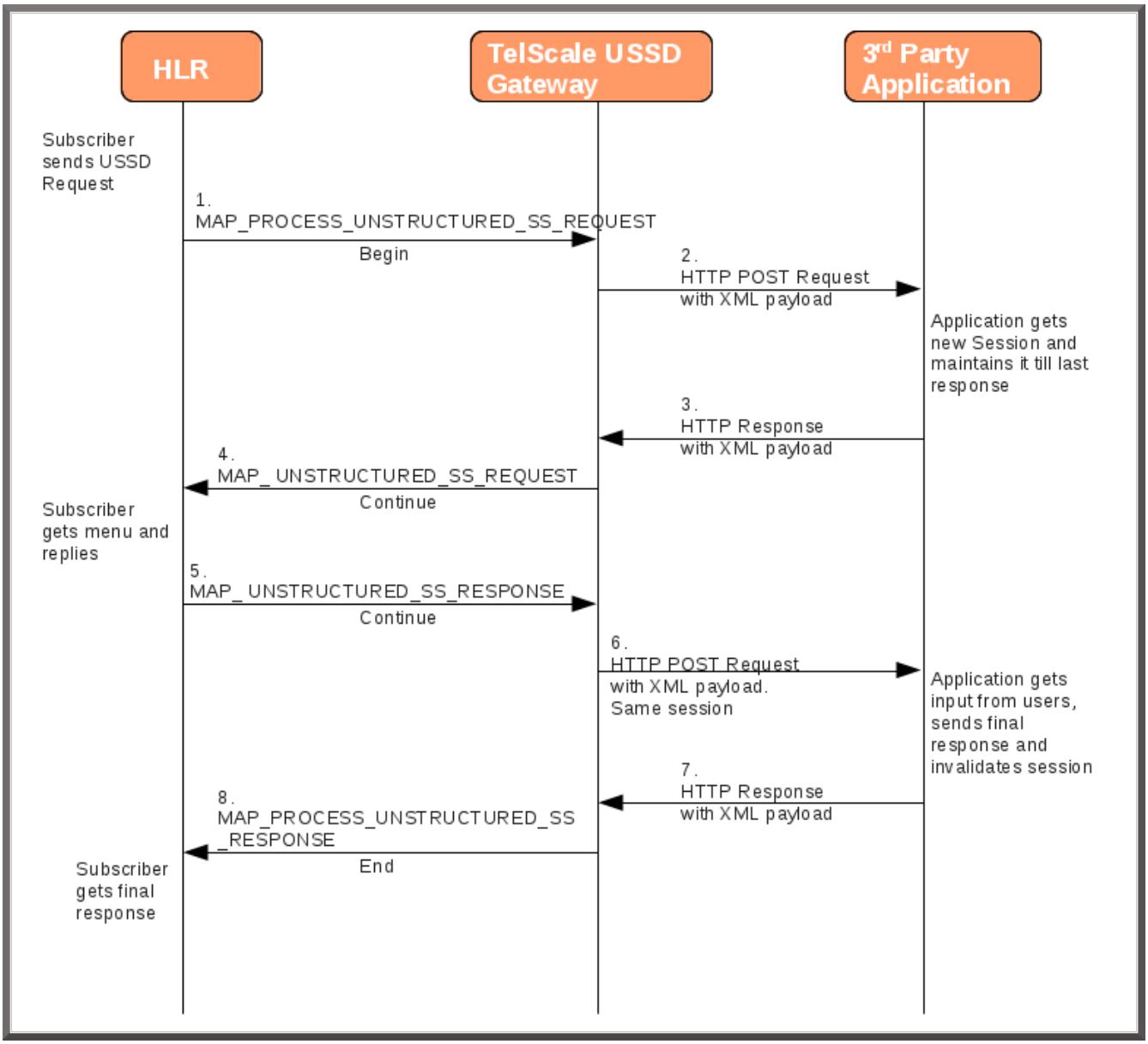
The HTTP callback mechanism allows the third-party Application to be agnostic to Operating System, Programming Language and Framework. The third-party Application could be one of the following technologies on any Operating System:

- 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

Chapter 1. HTTP Message Structure - USSD Pull

The diagram below depicts an example message sequence for interacting with the Restcomm USSD GATEWAY HTTP API.



1. When the subscriber initiates a USSD Request, the HLR sends a `MAP_PROCESS_UNSTRUCTURED_SS_REQUEST` to the USSD Gateway.
2. Upon receipt of the `MAP_PROCESS_UNSTRUCTURED_SS_REQUEST` of type `Begin`, USSD GATEWAY invokes the third-party Application through a `HTTP POST Request`, carrying an XML Payload with USSD specific information. The XML Structure of this payload is explained in the sections below.
3. The third-party Application will now get a new Session and maintain it till the last response. The Application will send a `HTTP Response`, carrying an XML Payload, to the USSD GATEWAY .
4. The USSD GATEWAY will send a `MAP_UNSTRUCTURED_SS_REQUEST` of type `Continue` to the HLR.
5. The Subscriber will receive a menu to choose from and reply. When the subscriber replies, the

HLR will send a `MAP_UNSTRUCTURED_SS_RESPONSE` of type `Continue` to the USSD GATEWAY .

6. The USSD GATEWAY will send a `HTTP POST Request`, carrying an XML Payload, to the third-party Application in the same session.
7. Based on the input from the subscriber, the third-party Application will send a final response (`HTTP Response`, carrying an XML Payload) and invalidate the session.
8. The USSD GATEWAY will send a `MAP_PROCESS_UNSTRUCTURED_SS_RESPONSE` of type `End` to the HLR. Consequently, the subscriber will receive a final response.

Chapter 2. HTTP Message Structure - USSD Push

For USSD PUSH, the third party application should send a HTTP POST request to the Restcomm USSD GATEWAY . The URL for sending this is http://<bind address of JBoss>:8080/mobicents. The diagram below depicts an example message sequence (USSD Push) for interacting with the Restcomm USSD GATEWAY HTTP API.

Chapter 3. HTTP Payload

3.1. XML Structure

The `HTTP Request/Response` carries XML Payload with USSD specific information as defined below:

```
<dialog type="Begin" appCntx="networkUnstructuredSsContext_version2"
    networkId="0" localId="1" remoteId="1" mapMessagesSize="1"
    returnMessageOnError="false">
    ....
    ....
    ....
</dialog>
```

3.2. Dialog Attributes

The XML element `<dialog>` contains various attributes to represent the state and the parameters of the Dialog. Some of them are mandatory while others are optional. This section explains in detail all possible attributes and what they represent. The list of mandatory attributes that a USSD application must supply will be described for an each concrete response.

Parameters

type

Represents the state of the Dialog. This parameter is for information only and is not used by the USSD GATEWAY to take any decision on the fate of the underlying MAP Dialog. The possible values for 'type' are: `Unidirectional, Begin, End, Continue, Abort, Unknown.`You do not need to supply this parameter in payload from HTTP App.

appCntx

Represents the MAP Application Context for USSD GW request. Now it always "networkUnstructuredSsContext_version2". You do not need to supply this parameter in payload from HTTP App. For outgoung USSD operations it is always networkUnstructuredSsContext version 2. For outgoing SRI request an initial MAP operation version is defined by the 'maxmapv' parameter configured in [_setting_ussd_maxmapv].

networkId

Represents a networkId value. USSD Gateway can be connected to multiple operators/network at same time. For each operator unique networkId is assigned. In the initial PUSH (unstructuredSS-Request or unstructuredSS-Notify) message HTTP app can specify networkId parameter for forsing of usage of desired network operator. If this value is missed then the default value ("0") will be used.

localId

Represents the local TCAP transaction ID. You do not need to supply this parameter in payload from HTTP App because localId values for all outgoing MAP dialogs is assigned by TCAP stack

itself. All subsequent messages from the USSD Gateway will have the actual value.

remoteId

Represents the remote TCAP transaction ID. You do not need to supply this parameter in payload from HTTP App.

mapMessagesSize

Represents the actual number of MAP messages carried in the <dialog>. This value can be 0 if no MAP messages are carried or 1 when a USSD message is present. This parameter is mandatory in payload from HTTP App and is usually "1" when you are sending a message or "0" for an empty TC-END response in PUSH case.

returnMessageOnError

For USSD GW originated XMP payload this parameter corresponds to SCCP message returnMessageOnError parameter. You can supply this parameter in payload from HTTP App. This will affect to outgoing SCCP messages. (SCCP will return the notification is the message has non been delivered to the peer). Mostly we do not need to specify this parameter. By default this is true.

prearrangedEnd

This parameter can only be present in payload from HTTP App. If this parameter is present, it means the underlying TCAP Dialog will be closed. The value can be true or false. If it is false, the messages will be sent to peer and the TCAP dialog ended. If it is true, all the messages in the Dialog are dropped and the Dialog is closed without informing peer. If this parameter is not present, all the messages in the Dialog are sent to peer as TCAP Continue. If you do not want to close dialog this step do not include this parameter. If you want to close dialog this step include this parameter with value "false".

mapAbortProviderReason

If this parameter is present, it means the underlying Dialog is Provider Aborted. The example below will describe in detail:

```
<dialog type="Unknown" localId="12" remoteId="13" mapMessagesSize="0"
mapAbortProviderReason="SupportingDialogueTransactionReleased"
returnMessageOnError="false">
    <errComponents/>
</dialog>
```

The possible values for `mapAbortProviderReason` are `ProviderMalfunction`, `SupportingDialogueTransactionReleased`, `ResourceLimitation`, `MaintenanceActivity`, `VersionIncompatibility`, `AbnormalMAPDialogueLocal`, `AbnormalMAPDialogueFromPeer` and `InvalidPDU`. You do not need to supply this parameter in payload from HTTP App.

mapRefuseReason

If this parameter is present, it means the underlying Dialog is refused by peer. The example below will describe in detail:

```
<dialog type="Unknown" localId="12" remoteId="13" mapMessagesSize="0"
mapRefuseReason="NoReasonGiven" returnMessageOnError="false">
    <errComponents/>
</dialog>
```

The possible values for `mapRefuseReason` are `ApplicationContextNotSupported`, `InvalidDestinationReference`, `InvalidOriginatingReference`, `NoReasonGiven`, `RemoteNodeNotReachable` and `PotentialVersionIncompatibility`. You do not need to supply this parameter in payload from HTTP App.

mapUserAbortChoice

If this parameter is present, it means peer user has aborted Dialog. The example below will describe in detail:

```
<dialog type="Unknown" localId="12" remoteId="13" mapMessagesSize="0"
mapUserAbortChoice="isUserSpecificReason" returnMessageOnError="false">
    <errComponents/>
</dialog>
```

The possible values for `mapUserAbortChoice` are `isProcedureCancellationReason_handoverCancellation`, `isProcedureCancellationReason_radioChannelRelease`, `isProcedureCancellationReason_networkPathRelease`, `isProcedureCancellationReason_callRelease`, `isProcedureCancellationReason_associatedProcedureFailure`, `isProcedureCancellationReason_tandemDialogueRelease`, `isProcedureCancellationReason_remoteOperationsFailure`, `isResourceUnavailableReason_shortTermResourceLimitation`, `isResourceUnavailableReason_longTermResourceLimitation`, `isUserResourceLimitation` and `isUserSpecificReason`. Even the HTTP App can Abort a specific Dialog by setting this value and sending it to the USSD Gateway.

dialogTimedOut

If this parameter is present, it means the underlying TCAP Dialog has timedout. The deafult value of TCAP Dialog timeout should always be greater than USSD Timeout value set in [_set_dialogtimeout].

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog type="Unknown" localId="12" remoteId="13" mapMessagesSize="0"
dialogTimedOut="true"
returnMessageOnError="false">
    <errComponents/>
</dialog>
```

You do not need to supply this parameter in payload from HTTP App.

emptyDialogHandshake

This parameter can only be present in payload from HTTP App. This is used only when USSD

gateway is initiating Dialog (Push case). This parameter indicates that USSD Gateway should firstly send empty dialog (without USSD Payload) and only once dialog is accepted by peer, USSD message should be sent.

```
<dialog mapMessagesSize="1" emptyDialogHandshake="true">
...
...
</dialog>
```

customInvokeTimeOut

This parameter can only be present in payload from HTTP App. Each MAP operation has its own default invoke timeout, for example for the unstructuredSS-Request MAP operation default invoke timeout is 10 min. HTTP App can set custom invoke timeout for each USSD message it sends to other end by using of this parameter (value is in milliseconds).

invokeTimedOut

This parameter indicates the invoke sent by USSD Gw has timed out. This generally means user has taken longer than expected to respond to USSD message. HTTP App can set custom invoke timeout for each ussd message by setting customInvokeTimeOut explained above. Once invoke timesout, USSD gateway will automatically abort the Dialog and send corresponding message to HTTP App. You do not need to supply this parameter in payload from HTTP App.

userObject

Application can set some user specific String value that the USSD gateway will always send back in corresponding messages exchanged:

```
<dialog type="Continue" appCntx="networkUnstructuredSsContext_version2"
localId="12" remoteId="13" mapMessagesSize="1" returnMessageOnError="false"
userObject="123456789">
....
....
</dialog>
```

3.3. Child Elements

The element <dialog> may contain any of these child elements but the order has to be respected. The possible child elements and the order to be followed, if present, must be as in the below list:

- SCCP Address
- AddressString
- Error Components
- processUnstructuredSSRequest_Request
- processUnstructuredSSRequest_Response
- unstructuredSSRequest_Request

- unstructuredSSRequest_Response
- unstructuredSSNotify_Request
- unstructuredSSNotify_Response

3.3.1. SCCP Address

Dialog carries SCCP information like <localAddress>, which is the SCCP Address of USSD gateway and <remoteAddress>, which is the SCCP address from where the Dialog is initiated, in case if this Dialog is received from other side (USSD Pull) or SCCP Address of remote side, incase of USSD Gateway acting as Proxy.

Both the elements are optional and if they are not passed, the USSD Gateway will get the values from the Global Title configured from [_setting_ussd_gt] for <localAddress>. If the USSD Gateway is acting as proxy, it mandatory to set the <remoteAddress>, else this parameter is ignored.

Attributes of SCCP (localAddress and remoteAddress) address are:

- pc: Mandatory parameter. Represents the point code.
- ssn: Mandatory parameter. Represents the Sub System Number.

Child elements of SCCP Address are <ai> and <gt>, where <ai> is Address indicator and suggests if routing is based on PC + SSN or GT. If it is based on GT, include <gt> element.

Please refer to the examples below:

```
a) Routing based on PC + SSN

    <localAddress pc="1" ssn="8">
        <ai value="67"/>
    </localAddress>

b) Routing based on GT
    <localAddress pc="0" ssn="146">
        <ai value="18"/>
        <gt type="GlobalTitle0100" tt="0" es="2" np="1" nai="4" digits="9960639902"/>
    </localAddress>
```

Different Gloabl Titles are `GlobalTitle0001`, `GlobalTitle0010`, `GlobalTitle0011` and `GlobalTitle0100`. For more details about SCCP, please refer to the jSS7 Admin Guide included in the documentation.

3.3.2. AddressString

Dialog carries the AddressString information like <destinationReference> and <originationReference>. The attributes of AddressString are defined below:

```
"nai" : nature of address indicator. The values are
    0 : unknown
    1 : international_number
    2 : national_significant_number
    3 : network_specific_number
    4 : subscriber_number
    5 : reserved
    6 : abbreviated_number
    7 : reserved_for_extension

"npi" : Numbering plan. The values are
    0 : unknown
    1 : ISDN
    2 : spare_2
    3 : data
    4 : telex
    5 : spare_5
    6 : land_mobile
    7 : spare_7
    8 : national
    9 : private_plan
    15 : reserved

"number" : The actual number

The XML example is
<destinationReference number="204208300008002" nai="international_number"
npi="land_mobile"/>
<originationReference number="204208300008002" nai="international_number" npi="ISDN"/>
```

3.3.3. Error Components

If peer reports `ErrorComponents`, same is forwarded to the Application through the child element `<errComponents/>`. The example of payload is:

```
<dialog type="End" networkId="0" localId="0" remoteId="0" mapMessagesSize="0"
sriPart="true" emptyDialogHandshake="true" returnMessageOnError="false">
    <errComponents>
        <invokeId value="1"/>
        <errorComponent type="MAPErrorMessageAbsentSubscriberSM" errorCode="6">
            <absentSubscriberDiagnosticSM value="IMSIDetached"/>
        </errorComponent>
    </errComponents>
</dialog>
```

3.3.4. processUnstructuredSSRequest_Request

This message is always sent by the USSD Gateway to the Application as a HTTP POST request or

from the Application to the USSD Gateway if it is acting as Proxy. The Application should always send back `processUnstructuredSSResponse` indicating that it is the last message of this dialog or can also send `unstructuredSSRequest` indicating that the Application is expecting more response from the user (menu structure).

The Attributes of `processUnstructuredSSRequest_Request` are defined below:

- "invokeId" : All message types have the mandatory `invokeId` attribute helping to relate the response to request. For example, `processUnstructuredSSResponse` will have the same `invokeId` as carried by `processUnstructuredSSRequest`. Hence if the Application has a multi-level menu, it should store the `invokeId`, received in the `processUnstructuredSSRequest` in a HTTP Session, for later use. Also for every new request in the same dialog, `invokeId` should be incremented by 1. For example when the Application sends `unstructuredSSRequest` to a received `processUnstructuredSSRequest` with `invokeId` equal to zero, it should set the `invokeId` equal to 1 in `unstructuredSSRequest`.
- "dataCodingScheme" : The Attribute `dataCodingScheme` is mandatory and represents the actual USSD Message. `dataCodingScheme` is the encoding parameter of the USSD Message.
- "string" : The Attribute `string` is mandatory and represents the USSD String length. In GSM 0902 160, octets are stated as the maximum length for the USSD String. However due to underlying signalling layers the maximum length of the USSD string depends on the message and can be less than 160.

The XML structure is defined below:

```
<processUnstructuredSSRequest_Request invokeId="0" dataCodingScheme="15"
string="*234#">
    <msisdn nai="international_number" npi="ISDN" number="79273605819"/>
    <alertingPattern size="1">
        <value value="6"/>
    </alertingPattern>
</processUnstructuredSSRequest_Request>
```

Child elements of `processUnstructuredSSRequest_Request` are <msisdn> and <alertingPattern>. The child element <msisdn> is optional and included only if the actual MAP message received by the USSD Gateway carries this value. This is MSISDN of the user who originated this request.

If <msisdn> is not included in processUnstructuredSSRequest, originationReference will be included in the dialog and this will be the MSISDN of the user originating the request.

<alertingPattern> is also an optional attribute.

3.3.5. processUnstructuredSSRequest_Response

This message is always sent by the Application to the USSD GateWay as a response to the received `processUnstructuredSSRequest` or `unstructuredSSResponse`. If the USSD Gateway is acting as Proxy, this message is sent from the USSD Gateway to the Application. This should always be the last message

in the dialog.

The XML structure is defined below:

```
<processUnstructuredSSRequest_Response invokeId="0" dataCodingScheme="15"
string="Thank You!"/>
```

3.3.6. unstructuredSSRequest_Request

This message is sent by the Application to the USSD GateWay in response to the received `processUnstructuredSSRequest` or `unstructuredSSResponse` in case of USSD Pull. In case of USSD Push, the Application can send this message to initiate a tree based menu push. This indicates that the Application is expecting some response from the user. If the USSD Gateway is acting as Proxy, this message is sent by the USSD Gateway to the Application.

The XML structure is defined below:

```
<unstructuredSSRequest_Request invokeId="0" dataCodingScheme="15" string="USSD String
: Hello World&#10; 1. Balance&#10; 2. Texts Remaining"/>
```

3.3.7. unstructuredSSRequest_Response

This message is sent by the USSD GateWay to the Application in HTTP POST request. This is a response to `unstructuredSSRequest` sent by the Application earlier. If the USSD Gateway acts as proxy, this message is sent by the Application to the USSD Gateway in response to `unstructuredSSRequest_Request` received by the Application.

The XML structure is defined below:

```
<unstructuredSSRequest_Response invokeId="0" dataCodingScheme="15" string="1"/>
```

3.3.8. unstructuredSSNotify_Request

This message is sent by the Application to the USSD GateWay to initiate USSD Push. This is just to notify the user and no input is expected back from the user. If the USSD Gateway is acting as proxy, this message is sent by USSD Gateway to the Application.

The XML structure is defined below:

3.3.9. unstructuredSSNotify_Response

This message is sent by the USSD Gateway to the Application in response to `unstructuredSSNotify_Request` sent by the Application. If the USSD Gateway is acting as proxy, the Application will send this message to the USSD Gateway.

The XML structure is defined below:

```
<unstructuredSSRequest_Response invokeId="0"/>
```

3.3.10. Using non-Latin menu

If you wish to send non-Latin text (example, Cyrillic, Arabic, etc) then you must set the value of the attribute `dataCodingScheme` to "72" (unlike the case of latin/digits where this is set to 15). You must ensure that any message text that contains non-Latin symbols must be UTF-8 encoded. The entire length of the message (in characters) must be less because this uses 2-byte encoding per character.

3.4. Typical usage of USSD application originated XML payload.

For USSD PULL case, a USSD application can usually issue following messages:

- PROCESS_UNSTRUCTURED_SS_RESPONSE
- UNSTRUCTURED_SS_REQUEST

For USSD PUSH case, a USSD application can usually issue following messages:

- UNSTRUCTURED_NOTIFY_REQUEST
- UNSTRUCTURED_SS_REQUEST
- RELEASE COMPLETE - finishing of a dialog

We will discuss few payload examples in this section.

3.4.1. PULL case - PROCESS_UNSTRUCTURED_SS_RESPONSE

Payload example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog mapMessagesSize="1" prearrangedEnd="false">
  <processUnstructuredSSRequest_Response invokeId="1" dataCodingScheme="15"
string="Your balance is 1 USD"/>
</dialog>
```

This message must finish PULL Dialog so we set "prearrangedEnd="false"". For USSD GW a count of internal messages is always 1, but this field is mandatory and we have to include it into a payload: mapMessagesSize="1". You have to also set "invokeId="1"" record which must be equal invokeId from processUnstructuredSSRequest_Request. "dataCodingScheme" value is usually "15" (GSM7 encoding) or "72" (USC2 encoding). "string" is USSD string value. Pay attention that as for GSM specification USSD string length has maximum length 182 characters for GSM7 coding and 80 characters for USC2 encoding.

3.4.2. PULL case - UNSTRUCTURED_SS_REQUEST

Payload example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog mapMessagesSize="1">
  <unstructuredSSRequest_Request dataCodingScheme="15" string="Press 1 for paying or
press 2 for aborting"/>
</dialog>
```

This message will continue PULL Dialog. "userObject" attribute is optional. You can use it for identification puposes. This value will be return to USSD application with the all next xml payloads.

3.4.3. PUSH case - UNSTRUCTURED_NOTIFY_REQUEST

Payload example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog mapMessagesSize="1">
  <unstructuredSSNotify_Request dataCodingScheme="15" string="Your new balance is
34.38 AFN and expires on 30.07.2012. Cost of last event was 0.50 AFN.">
    <msisdn nai="international_number" npi="ISDN" number="11111111111111"/>
  </unstructuredSSNotify_Request>
</dialog>
```

You are initiating a PUSH dialog. We need to include "msisdn" parameter which identifies a destination subscriber phone number (nai is "Nature of address", npi means "Numeric plan"). "msisdn" parameter must be included only in the first unstructuredSSNotify_Request or unstructuredSSRequest_Request of the PUSH dialog.

3.4.4. PUSH case - UNSTRUCTURED_SS_REQUEST

Payload example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog mapMessagesSize="1">
  <unstructuredSSRequest_Request dataCodingScheme="15" string="Press 1 for paying or
press 2 for aborting">
    <msisdn nai="international_number" npi="ISDN" number="5444444444"/>
  </unstructuredSSRequest_Request>
</dialog>
```

3.4.5. PUSH case - RELEASE COMPLETE - finishing dialog

Payload example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog mapMessagesSize="0" prearrangedEnd="false">
</dialog>
```

PUSH dialog can include one or more UNSTRUCTURED_NOTIFY_REQUEST or UNSTRUCTURED_SS_REQUEST. After this USSD application must finish the dialog by this payload. PUSH dialog always uses several HTTP request to create, continue and terminatie it. For successfull processing it is needed to use a http cookie JSESSIONID. First http response contains a session cookie like "Set-Cookie: JSESSIONID=1379BF8AF4DB8CACF444AEA6375AD85E; Path=/mobicents". Every next request must contain a tag like: "Cookie: JSESSIONID=1379BF8AF4DB8CACF444AEA6375AD85E" Every request (UNSTRUCTURED_SS_REQUEST and UNSTRUCTURED_NOTIFY_REQUEST) can contain extra parameter "customInvokeTimeout" in the dialog section to set up a custom invoke timeout (in milliseconds). For example (for 10 minutes invoke timeout):

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialog mapMessagesSize="1" customInvokeTimeout="600000">
  <unstructuredSSRequest_Request dataCodingScheme="15" string="Press 1 for paying or
press 2 for aborting">
    <msisdn nai="international_number" npi="ISDN" number="5444444444"/>
  </unstructuredSSRequest_Request>
</dialog>
```