JSR-180 Maintenance Release Change Log

13/10/2004

Please send feedback to: `jsr-180-comments@jcp.org`

PROPOSED changes

	Change	Reasoning
1	Add Message Sequence Chart (MSC) and/or example code snippet of the method usage: `SipClientConnection.setCredentials()`. page 23	It should be more specific how the SIP authentication below the API works. Now the `setCredentials()` method explanation is a bit vague.
2	Should say clearly that the `SipClientConnection` returned from the `SipDialog.getNewClientConnection()` is in *Initialized* state. page 35	The state is not specified explicitly in the text.

3	Correct `SipDialog` SUBSCRIBE code example. The opened `SipClientConnection` is not in shared mode. Should set both `From` and `Contact` header for the initial SUBSCRIBE request.
	
	Also correct the line
	`scc.setHeader("Accept", "application/xpidf+xml");`
	to be
	`scc.setHeader("Accept", "application/pidf+xml");`
	
	page 34
		Confusion when the `From` and `Contact` header should be set by the application. `From` and `Contact` headers are never set in the code example.

4	The `SipDialog` state diagram indicates that the `SipDialog` instance is already created when INVITE/SUBSCRIBE is sent. That is not the case; it is not necessary (for API implementation) to create the `SipDialog` until the provisional 101-199 response is received. The state diagram is also wrong for the SUBSCRIBE case because there you do not get provisional response at all. It is directly 2xx or NOTIFY, that creates the dialog. Proposed to add a new state *Initialized* which in practise is never visible to the user, but acts as a common starting point for both INVITE/SUBSCRIBE cases. The `SipDialog` can be fetched earliest in the *Early* state. In order to help reading draw separate `SipDialog` state diagrams for both client and server sides. page 33	The `SipDialog` state diagram is unambiguous and false for SUBSCRIBE method.

5	`SipClientConnection.receive()` cannot be called after sending ACK. This prevents receiving 200 OKs from multiple end-points (forking case) and multiple re-sent 200 OKs from the same end-point. Correct state diagram and the rules for receiving multiple 2xx responses. Enable calling `receive()` in *Completed* state. Furthermore, it should be clarified how responses from multiple endpoints are treated in the `SipClientConnection`. Essentially `SipClientConnection` associates with latest response fetched with `receive()` method. Also the dialog will be always associated to the latest response received. page 16 & 22	Unclear how the multiple 200 OK responses are handled with the API.
6	Error response to INVITE: When error response for INVITE is received the client transaction sends automatically ACK. Now on the server side this ACK is handled in the same server transaction that received original INVITE, but the ACK is not given up to the TU (as a new `SipServerConnection`). 200 OK response to INVITE: Should be able to resend 200 OK if the ACK has not been received. Correct `SipServerConnection` state diagram to enable calling `send()` for 2xx responses in *Completed* state. Generally if sending response fails `IOException` is thrown. These should be clarified in the `SipServerConnection` section. page 24	Clarification why ACK is (not) visible to the application in error response case. In the case where 200 OK has been sent for INVITE, but the ACK has not been received it should be possible to resend the 200 OK

7	Rewrite server connection initialization rules. `SipConnectionNotifier` can be opened with following URI:
	
	`sip:[nnnn][;type="application/x-game"]`
	
	`sip: or sips: - protocol scheme without address to indicate server mode`
	
	`nnnn - listening port number (optional)`
	
	`type - URI parameter specifying application identifier (optional)`
	
	- Port number `nnnn` specifies the listening port. The `IOException` is thrown if the port number is already in use or it cannot be opened for other reason. If the port number is given it always indicates dedicated port number for the application (not shared SIP identity mode).
	
	- If only "`sip:`" scheme is given the system is selecting the port number, otherwise the case is similar as if the number was given by the application.
	
	- If application identifier is given with parameter 'type' the system listening port and the SIP identity is shared, with other applications (shared SIP identity mode).
	
	page 9 & 10
		Clarify `SipConnectionNotifier` opening method to have more exact rules. It is assumed that applications using JSR180 would mostly use the shared identity case (where port number is omitted).

8	More specific definition how incoming request messages are dispatched to applications when using shared mode server connection. page 9,10, 11 (SIP Identity) page 58 (Appendix A)	Clarify how the SIP dispatching should be done in the JSR180 implementations.
9	Correct interface `SipServerConnectionListener`, the parameter `ssc` in `notifyRequest()` should be named `scn`. page 32	This is just to avoid confusion. The type of the `notifyRequest()` parameter is `SipConnectionNotifier` not `SipServerConnection`.
10	Should state clearly that the `SipConnection.send()` method is asynchronous. Any kind of immediate failure should throw `IOException`. The `InterruptedIOException` should be removed from `SipConnection.send()`. page 12	Now for example if the network is not available calling `send()` could throw `IOException` immediately. The description of `InterruptedIOException` is wrong for `send()`, since it does not timeout. Furthermore, this exception does not carry any extra information to the `IOException`. Also this exception is used with `InputStream/OutputStream` based classes.
11	Create PRACK from the `SipDialog`. At the moment the `SipDialog` does not allow that in the state diagram specification. It should be generally possible to call `getNewClientConnection()` in *Early* state page 33	Support for RFC3262, Reliability of Provisional Responses in the Session Initiation Protocol (SIP)

12	Remove the `InterruptedIOException` from `SipConnectionNotifier.acceptAndOpen()`. page 29-30	The `InterruptedIOException` is used with the `InputStream` and `OutputStream` based classes and does not give any additional information here. If the `SipConnectionNotifier` is closed for some reason the `IOException` should be thrown.
13	Giving `SipConnectionNotifier` as a parameter for `SipClientConnection.initRequest(…)` 1) In the shared mode, when creating a new request, the `From` and `Contact` headers are set by the system (when the request is initialized). Furthermore, the SIP registration is done by the system automatically (e.g. when terminal is started). 2) In not-shared mode, the application is responsible for setting the `From` and `Contact` header (change to current specification). Furthermore, the application has to do the SIP registration. page 19-20	Clarify the rules for `SipClientConnection.initRequest(…)`, when `SipConnectionNotifier` is given as a parameter. The `SipConnectionNotifier` can be in either shared or not shared SIP identity mode. See also the item 7.

ACCEPTED changes

None

DEFERRED changes

None