Review Record
Document: FSD for Number Normalization and SIP URI Aliasing version 1.0.
Review Date: 2008-09-18

Reviewer	Sreeram Duvur
Comment	Can you document the syntax for the mapping rules more formally? I am thinking of the end user documentation writer who could simply borrow from this spec.
Response/Resolution	I will clarify the syntax, but it is quite simple. The property file declares the mappings in the order they shall be applied. Each line in the property file specifies the mapping: <regular expression>=<fully qualified class name of handler plug-in class> The regular expression will be according to the documentation of the class of java.util.regex.Pattern, with the appropriate escaping to be handled correctly as a resource bundle. For example would the regexp for matching strings starting with "+46" be \\+46.*. The first '\' is needed to escape the second '\' so that the resource reader does not interpret it as a line continuation. The second '\' is used to escape the '+' so that the regexp matcher does not interpret the '+' as a quantifier.

Reviewer	Sreeram Duvur
Comment	If you want mappings per application, why are they not in the application archive itself? Inside the sar, war, ear?
Response/Resolution	I do not mean that mapping rules are specified per application, mapping rules are configured for the application server as a whole. The plugins are most probably be provided by the application, but they cannot be included in the application archive when deploying multiple applications. The reason for this is that for a given phone-context or SIP URI alias pattern, each application might provide its own mapping rule implementation. However, at the point in the request processing where the mapping rules will be used (in the CLB) it is not known which application(s) are going to handle the request and the application server would not know which variant of the same mapping rule to use. Consequently, for a given application combination each such clashing mapping rule plug-in class must be combined into one single plug-in class. To make this possible the plug-ins and their mappings must be provided at application server level (rather than on application level) so that it is

	possible to create the appropriate combination for a given application combination. An example would be the implementation of the swedish number plan. Each application would need to have an implementation (to be able to work on its own in a single-application deployment), but when multiple applications are deployed they would all use the same plug-in for the swedish number plan. The worst situation that could occur is that they have different implementations and application-specific features must be supported. In that case a new plug-in must be implemented combining the different features. This could of course be tricky since it requires knowledge of the different implementations (in the worst case the rules are exclusive and might not be possible to combine, but such situation cannot be solved in another way than to make a compromise). However, these situations should be quite rare, but at least it is possible solve then without the need of rewriting the applications.

Reviewer	Binod P.G
Comment	Given the actual mapping is done by the plugin, it is possible (or quite likely in case of aliases) that the plugin will do a DB access to complete the mapping, right? Are there any guideline to avoid performance impact due to the DB access? DB access may be required for all requests (Any string can be alias for another string). That would affect the front end performance a lot. As you said, it may be unavoidable
Response/Resolution	This is a situation where the container must do a call into application code. The application must try to write this code as efficient as possible to minimize the performance impact. This must be emphasized in the CPI. It can be limited somewhat by the regular expressions mapping it, so that only certain alias patterns are handled. But it is correct that if one has an alias pattern like "*" (i.e. all URI:s can be aliased in any way), then all requests must be checked. But I guess it might be required that the application do some caching in certain situations. The container shall not try to do caching of the results from the handlers (since we do not the dynamics of the application in this area, e.g. the handler might come up with a different result at different times on the day).

Reviewer	Binod P.G
Comment	Is there any reason, why we choose properties files for specifying the mapping rules. Given that we already have configuration files for CLB and DCR, yet another file may not be good. Cant we just put this into domain.xml?
Response/Resolution	I did this because it would be easy to pack the handler class files and the mappings into the same jar-file, but we might as well put the mapping rules into the domain.xml.

	Decision: the mappings are specified as properties below the sip-service element in the domain.xml

Reviewer	Binod P.G
Comment	Assuming that the plugins will need to implement a new java interface, please add it under sailfin/common/src/main/java/org/glassfish/comms/api
Response/Resolution	Yes.

Reviewer	Lars Andersson
Comment	Must the server be restarted when adding a plugin class?
Response/Resolution	Yes, the server must be restarted. However, it is possible to update the mappings without restarting the server. A possible future enhancement could be to support installation of new plugins without restart.

Reviewer	Andreas Burmester
Comment	It would be preferable if the plugin classes (jar-file) could be deployed using an admin command rather than copying a jar to a directory.
Response/Resolution	OK. This will also be a future enhancement.