3.7 operations searchForMembers, searchForMembersGraphic

M 123 `searchForMembers` ($MUST_{123}$) realizes the core of the usecase 2.4. It takes many parameters describing the search filters and returns an array of `Memberinfo` objects that represent the members thus found as described in Section 3.2.

Unused filters are indicated by nil for string, false for boolean, 0 for int where 0 is a null-filter, 999999 for int where infinity is a null-filter. tttTypes is an array of TTT type strings (such as "ESTP" etc.) that indicates the set of types of interest.

`searchForMembersGraphic` ($MUST_{124}$) performs the same search, but instead of returning `Memberinfo` objects, it returns the byte sequence of a GIF or PNG image file as described in Section 2.5.3. The image is `xSize` pixels wide, `ySize` pixels high and indicates the values of `xVariable` along the horizontal axis and `yVariable` along the vertical axis; the variables are given by their names as described in Section 2.5.3. M 124

3.8 operations getMemberlist, getMemberlistGraphic

`getMemberlist` ($MUST_{125}$) realizes the core of the usecase 2.5. It retrieves the `Memberinfo` of all members that have the given `rcdStatus` from the point of view of member `username`. M 125

`getMemberlistGraphic` ($MUST_{126}$) is to `getMemberlist` what `searchForMembersGraphic` is to `searchForMembers`. M 126

For both operations, if the current user is not allowed to get the list (because `username` is not the current user and `rcdStatus` is not "in_contact" or because `rcdStatus` is "no_contact"), nil is returned.

3.9 operation sendRcd

`sendRcd` ($MUST_{127}$) realizes an aspect of usecases 2.5 and 2.6: it sends an RCD from the current user to the members indicated in the `username` array or answers an RCD received from these members. Sending and positive answering is exactly the same: `positive` is true. Negative answering happens when `positive` is false. Negative answering to a member that had not send an RCD has no consequence whatsoever. Sending an RCD to a member with in_contact status has no consequence whatsoever. M 127

4 Non-functional requirements

4.1 User interface

M 128 The user interface $MUST_{128}$ conform to the above-mentioned requirements (as far as they are realized at all) in a sensible way with respect to the arrangement and markup of its elements and the labels, prompts and explanations that guide the user. Within those limits, the organization and design of the interface is left to the professional judgement of the participants.

S 129 The userinterface $SHOULD_{129}$ provide sufficient explanation of all uncommon concepts to guide a user who does not have prior knowledge about these topics. Make use of external links where needed. Do not include copyrighted material without permission.

m 130 Elaborate graphical design (MAY_{130}) is not important, but CSS $SHOULD_{131}$ be used throughout to
S 131 simplify future improvements.

m 132 It would be nice if the user interface works even when Javascript is turned off in a user's browser (MAY_{132}).

4.2 Browser compatibility

M 133 • The portal $MUST_{133}$ work fully with Firefox 1.5 and higher.

M 134 • The portal $MUST_{134}$ work fully with Internet Explorer 6 and higher.

S 135 • The portal $SHOULD_{135}$ work fully with Safari 2 and higher.

S 136 • The portal $SHOULD_{136}$ work fully with Opera 8 and higher.

m 137 • The portal MAY_{137} work fully with other browsers such as Konqueror, Opera Mini, Lynx etc.

S 138 • If the portal relies on Javascript in the browser and Javascript is unavailable, it $SHOULD_{138}$ either produce a clear message saying that the portal is not functional (and why) or fall back to reduced (but still useful) functionality.

4.3 Scalability

M 139 The system $MUST_{139}$ scale without problems to 1,000,000 registered members.

M 140 For a realistic mix of requests, it $MUST_{140}$ have response times below three seconds (for at least 90% of the requests) up to at least 200 concurrently active users on a server with 1 CPU, 1 disk and 1 GB of RAM.

S 141 It $SHOULD_{141}$ have such response times up to at least 1000 concurrent users.