

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

Marlin - Proximity Specification

Version 1.0.1
Final

Source
Date

Marlin Developer Community
January 20, 2012

32 **Notice**

33 THIS DOCUMENT IS PROVIDED "AS IS" WITH NO REPRESENTATION
34 OR WARRANTY, EXPRESS OR IMPLIED, CONCERNING THE
35 COMPLETENESS, ACCURACY, OR APPLICABILITY OF ANY INFORMATION
36 CONTAINED IN THIS DOCUMENT. THE MARLIN DEVELOPER COMMUNITY
37 ("MDC") ON BEHALF OF ITSELF AND ITS PARTICIPANTS (COLLECTIVELY,
38 THE "PARTIES") DISCLAIM ALL LIABILITY OF ANY KIND WHATSOEVER,
39 EXPRESS OR IMPLIED, ARISING OR RESULTING FROM THE RELIANCE OR
40 USE BY ANY PARTY OF THIS DOCUMENT OR ANY INFORMATION
41 CONTAINED HEREIN. THE PARTIES COLLECTIVELY AND INDIVIDUALLY
42 MAKE NO REPRESENTATIONS CONCERNING THE APPLICABILITY OF ANY
43 PATENT, COPYRIGHT (OTHER THAN THE COPYRIGHT TO THE
44 DOCUMENT DESCRIBED BELOW) OR OTHER PROPRIETARY RIGHT OF
45 THIS DOCUMENT OR ITS USE, AND THE RECEIPT OR ANY USE OF THIS
46 DOCUMENT OR ITS CONTENTS DOES NOT IN ANY WAY CREATE BY
47 IMPLICATION, ESTOPPEL OR OTHERWISE, ANY LICENSE OR RIGHT TO
48 OR UNDER ANY PATENT, COPYRIGHT, TRADEMARK OR TRADE SECRET
49 RIGHTS WHICH ARE OR MAY BE ASSOCIATED WITH THE IDEAS,
50 TECHNIQUES, CONCEPTS OR EXPRESSIONS CONTAINED HEREIN.

51 Use of this document is subject to the agreement executed between you and the
52 Parties, if any.

53 Any copyright notices shall not be removed, varied, or denigrated in any manner.

54 Copyright © 2003 - 2012 by MDC, 415-112 North Mary Avenue #383 Sunnyvale, CA
55 94085, USA. All rights reserved. Third-party brands and names are the property of
56 their respective owners.

57 **Intellectual Property**

58 A commercial implementation of this specification requires a license from the Marlin
59 Trust Management Organization.

60 **Contact Information**

61 Feedback on this specification should be addressed to: editor@marlin-community.com

62 Contact information for the Marlin Trust Management Organization can be found at:
63 <http://www.marlin-trust.com/>

64

65 **Contents**

66

67 1 Introduction 4

68 1.1 Terminology and Conventions..... 4

69 1.2 Namespaces and Identifiers 4

70 1.2.1 Namespaces and Notation..... 4

71 1.3 Abbreviations 4

72 1.4 References 5

73 1.4.1 Normative References 5

74 2 Mechanism for Proximity Check 6

75 2.1 Proximity Check Protocol over UDP 6

76 2.1.1 Overview 6

77 2.1.2 Preconditions..... 6

78 2.1.3 Generation of the set R of Q pairs from a Seed S 6

79 2.1.4 Protocol Steps 6

80 2.1.4.1 Target Setup 6

81 2.1.4.2 RTT Measurement Loop 7

82 2.1.5 Sequence Diagram..... 8

83 2.1.6 Timing Parameters 8

84 2.1.7 Security Considerations 8

85 2.1.8 NEMO Security Policies 9

86 2.1.9 Message Encodings 9

87 2.1.9.1 Setup..... 9

88 2.1.9.2 RTT Measurement Loop 9

89 3 Octopus Bindings 10

90 3.1 Constraints..... 10

91 3.2 Control Context..... 10

92 Appendix: XML Schema and WSDL File Names 11

93

94

95 1 Introduction

96 This specification describes the protocol and Octopus binding mechanism used for
97 proximity check.

98 1.1 Terminology and Conventions

99 The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”,
100 “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this
101 specification are to be interpreted as described in IETF RFC 2119 [RFC2119].
102

103 These capitalized key words are used to unambiguously specify requirements and
104 behavior that affect the interoperability and security of implementations. When these
105 key words are not capitalized they are meant in their natural-language sense.
106

107 All elements of this specification are considered Normative unless specifically
108 marked Informative. All Normative Elements are Mandatory to implement, except
109 where such an element is specifically marked OPTIONAL. Finally, where Normative
110 elements are described as OPTIONAL, they MAY be omitted from an implementation,
111 but when implemented, they MUST be implemented as described.

112 1.2 Namespaces and Identifiers

113 This specification defines schemas conforming to XML Schemas [Schema] and
114 normative text to describe the syntax and semantics of XML-encoded objects and
115 protocol messages. In cases of disagreement between the schema documents and
116 the schema listings in this specification, the schema documents take precedence.
117 Note that in some cases the normative text of this specification imposes constraints
118 beyond those indicated by the schema documents.

119 1.2.1 Namespaces and Notation

120 The following table summarizes the normative schemas defined by this specification,
121 and their XML namespace [XMLns] URIs. These URIs MUST be used by
122 implementations of this specification:
123

Prefix	XML Namespace	Schema File Name	Description
--------	---------------	------------------	-------------

124
125 In addition to the schemas defined by this specification, we leverage existing
126 schemas to achieve our design goals. The following table summarizes the external
127 schemas used in this specification:
128

Prefix	XML Namespace	Description
xsd:	http://www.w3.org/2001/XMLSchema	[Schema]

129

130 1.3 Abbreviations

HTTP	Hypertext Transfer Protocol
NEMO	Networked Environment for Media Orchestration
RTT	Round-Trip Time
SOAP	Simple Object Access Protocol

UDP	User Datagram Protocol
WSDL	Web Services Description Language
XML	Extensible Markup Language

132

133 **1.4 References**

134 **1.4.1 Normative References**

135

[8pus]	Octopus DRM Technology Platform Specifications, Version 1.0
[MCS]	Marlin – Core System Specification, version 1.3 and its latest errata
[SHA1]	FIPS PUB 180-1. <i>Secure Hash Standard</i> . U.S. Department of Commerce/National Institute of Standards and Technology. http://www.itl.nist.gov/fipspubs/fip180-1.htm
[RFC2119]	S. Bradner, <i>Key words for use in RFCs to Indicate Requirement Levels</i> , IETF RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt .

136 **2 Mechanism for Proximity Check**

137 This section defines the mechanism which SHALL be used to check proximity.

- 138 • When two implementations are connected through IP, the proximity check SHALL
139 be done by using the Proximity Check Protocol over UDP defined in §2.1.
- 140 • When two implementations are connected through USB, the proximity check
141 SHALL always be considered valid without measurement.
- 142 • In all other cases, the proximity check SHALL be considered failure.

143 **2.1 Proximity Check Protocol over UDP**

144 **2.1.1 Overview**

145 This protocol allows an anchor to check the proximity of a target.

146 This protocol is asymmetric as the anchor generates a secret seed and is the only
147 one that requires a secure timer. Moreover, the target does not need to trust the
148 anchor. It is also cryptographically efficient requiring only two public key operations.

149 **2.1.2 Preconditions**

150 The target is a NEMO client node and has, as such, a set of NEMO keys and
151 credentials. The anchor is not required to be a NEMO node, as no anchor NEMO
152 credentials are used in this protocol.

153 **2.1.3 Generation of the set R of Q pairs from a Seed S**

154 The set R is obtained from randomly generated seed using the following method:

$$155 R_i = H^{2^{Q-i}}(S)$$

156

157 H(M) is the digest value of the hash function H over the message M.

$$158 H^n(M) = H(H^{n-1}(M)) \text{ for } n \geq 1 \text{ and } H^0(M) = M$$

159

160 The algorithm used for the hash function H() SHALL be [SHA1].

161 **2.1.4 Protocol Steps**

162 **2.1.4.1 Target Setup**

163 The target chooses a UDP port number TargetPort that it is ready to receive
164 ChallengeRequest UDP datagrams on. The target chooses 32-bit number,
165 TargetSessionId, that it can use to differentiate between several concurrent protocol
166 sessions if necessary. The target also chooses the target timing parameters
167 SetupDelay and LoopDelay.

168 The target sends a TargetSetupRequest message to the anchor. The payload of this
169 message contains TargetSessionId, TargetPort, SetupDelay and LoopDelay.

170

171 Upon receiving and validating the TargetSetupRequest, the anchor chooses a
172 random seed S, a maximum loop count Q, and generates a set R of Q pairs of
173 random numbers as specified in §2.1.32.1.2. The anchor chooses a UDP port
174 number AnchorPort that it is ready to receive ChallengeResponse UDP datagrams
175 on. Q MUST NOT exceed 254. The anchor chooses 32-bit number, AnchorSessionId,
176 that it can use to differentiate between several concurrent protocol sessions if

177 necessary. The anchor also chooses the anchor timing
178 parameterTerminationTimeout.
179 The anchor then sends a TargetSetupResponse reply back to the target. The
180 payload of the reply contains AnchorSessionId, Q, S, AnchorPort and
181 TerminationTimeout.
182
183 Upon receiving this reply, the target computes R from Q and S, as specified in
184 §~~2.1.32-1.2.~~
185 The target is now ready to receive ChallengeRequest UDP datagrams on TargetPort.

186 **2.1.4.2 RTT Measurement Loop**

187 The Anchor MAY perform up to Q RTT measurement loops for each protocol session.
188 Each loop consists of the following steps. The loop counter i is initialized at 0 and is
189 incremented by 1 for each iteration through the loop.

- 190 a) The anchor measures $T = \text{now}$
- 191 b) The anchor sends to the target a ChallengeRequest UDP datagram on port
192 Target-IP:TargetPort, where Target-IP is the IP address of the TCP endpoint
193 used by the target to send the TargetSetupRequest message. The
194 ChallengeRequest payload contains TargetSessionId (established during the
195 setup step), i, and R_{2^i}
- 196 c) The target receives the ChallengeRequest UDP datagram. If the value of R_{2^i} is
197 correct, it responds with ChallengeResponse UDP datagram sent to Anchor-
198 IP:AnchorPort, where Anchor-IP is the IP address of the TCP endpoint to which
199 the TargetSetupRequest was sent. The ChallengeResponse payload contains
200 AnchorSessionId (established during the setup step), i, and $R_{2^{i+1}}$.
- 201 d) The anchor receives the ChallengeResponse UDP datagram. The anchor
202 measures $\text{RTT} = \text{now} - T$. If the value of $R_{2^{i+1}}$ is correct, the value of RTT is
203 accepted as a valid measurement. The anchor may keep the lowest valid
204 measured RTT along with the date and time of the measurement if it does not
205 terminate the loop before it has exhausted the Q possible iteration.

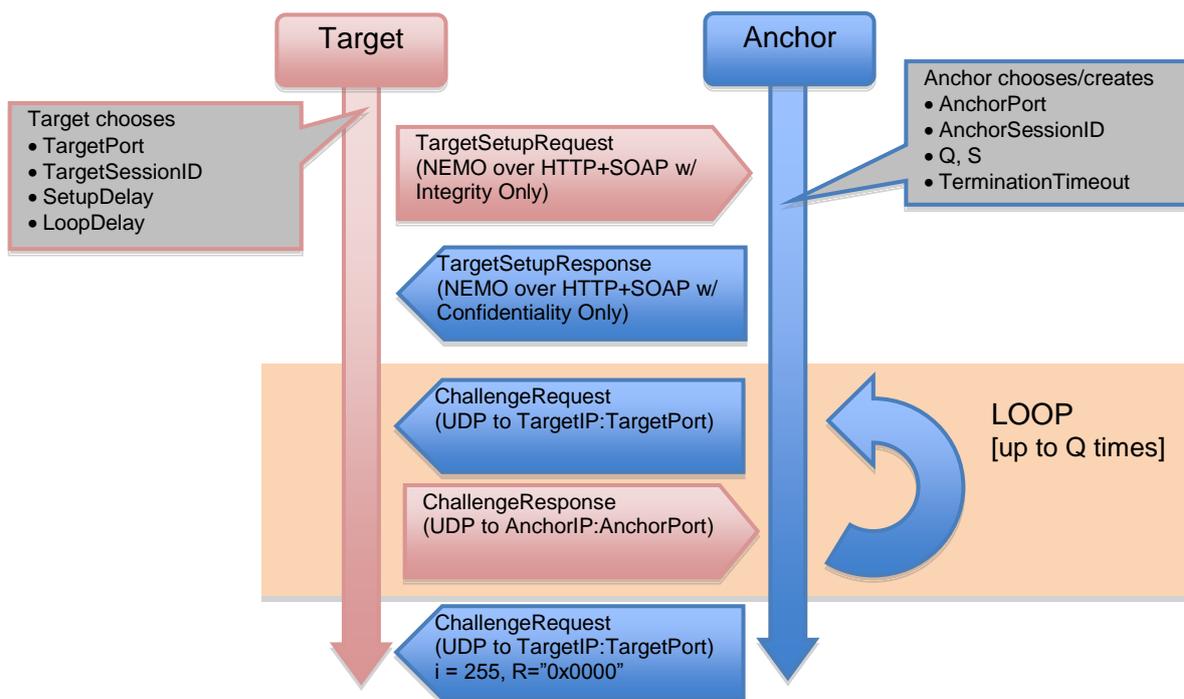
206
207 The anchor MAY terminate the loop at any point before it has exhausted the Q
208 possible iterations (for example if it determines that the lowest measured RTT value
209 is below a certain threshold, or if it receives no response from the target for a long
210 time). When the anchor terminates the loop, it MUST send at least one special
211 "Termination" ChallengeRequest UDP datagram. A "Termination" ChallengeRequest
212 UDP datagram is one where the value of "i" is equal to 255 and the bytes for the ' R_{2^i} '
213 field are all set to 0. The anchor MAY send more than one "Termination"
214 ChallengeRequest UDP datagram for redundancy (delivery of UDP datagrams is not
215 guaranteed).

216
217 Since the datagrams exchanged during this loop are exchanged over UDP, it is
218 possible that either the target and/or the anchor receive on the same UDP port some
219 datagrams belonging to different sessions. It is therefore important that when a
220 session is started, both the target and the anchor use the TargetSessionId and
221 AnchorSessionId, respectively, to decide which session a received datagram is a part
222 of. All datagrams processed during the RTT measurement loop MUST match the
223 TargetSessionId or AnchorSessionId that were established during the setup phase;
224 all other datagrams MUST NOT be considered part of the session (but they may still
225 be part of a different session happening at the same time).

226
227

2.1.5 Sequence Diagram

228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250



2.1.6 Timing Parameters

251
252 The target timing parameters sent in the TargetSetupRequest message, SetupDelay
253 and LoopDelay, are both expressed in milliseconds.
254 The anchor MUST wait at least SetupDelay between the transmission of the
255 TargetSetupResponse reply and the transmission of the first ChallengeRequest
256 datagram.
257 The anchor MUST wait at least LoopDelay between two consecutive
258 ChallengeRequest messages while in the RTT measurement loop.
259
260 The anchor timing parameter sent in the TargetSetupResponse message,
261 TerminationTimeout, is expressed in milliseconds. After the setup, the target
262 SHOULD keep listening for UDP datagrams on TargetPort for at least
263 TerminationTimeout after the last received message from the anchor (either the
264 TargetSetupResponse or any ChallengeRequest) unless the protocol can be
265 determined to have terminated (either a "Termination" ChallengeRequest UDP
266 datagram has been received, or the last possible ChallengeRequest, with the loop
267 counter i equal to Q-1, has been received). If no message has been received for
268 more than that amount of time, the protocol is implicitly terminated.
269
270 A valid under-threshold RTT measurement MUST be 7 milliseconds or less, unless it
271 is overridden by another specification.

2.1.7 Security Considerations

272
273 When engaging in this protocol, care must be taken to follow the following basic
274 requirements.

275 The anchor MUST choose the seed S with a non-guessable secure random or
276 pseudo-random number generator such that the chances of using the same value S
277 in two separate protocol sessions is infinitesimal.

278 The RTT measurement loop MUST NOT be repeated with the same value of i during
279 a protocol session.

280 The protocol MUST be aborted if any unexpected message is received by either
281 party, including:

- 282 • If the target receives an incorrect value for R_{2^i} in step c.
- 283 • If Q is larger than the maximum allowed value.
- 284 • If i is repeated in the loop
- 285 • If i exceeds Q

286 **2.1.8 NEMO Security Policies**

287 The TargetSetupRequest request MUST follow the 'Integrity Only' policy, as defined
288 in [MCS] §5.2.3.3.

289 The TargetSetupResponse reply MUST follow the 'Confidentiality Only' policy as
290 defined in [MCS] §5.2.4.4.

291
292 The identifier for this protocol's security policy is

urn:marlin:proximityoverudp:1-0:nemo:services:proximity-check:policy:1
--

293 **2.1.9 Message Encodings**

294 The XML schema for this protocol is defined in the XML Namespace
295 urn:marlin:proximityoverudp:1-0:nemo:services:schemas

296
297 A copy of the XML schema and WSDL is in Appendix A.1 and A.2, respectively.
298

299 **2.1.9.1 Setup**

300 The TargetChallengeRequest and TargetChallengeResponse messages are defined
301 in the XML schema

302 **2.1.9.2 RTT Measurement Loop**

303 **2.1.9.2.1 ChallengeRequest**

304 The payload for the ChallengeRequest is the following byte sequence.

Byte	0	1-4	5-24
Description	i	TargetSessionId, encoded as a 32-bit integer in big-endian byte order	R_{2^i}

305

306 **2.1.9.2.2 ChallengeResponse**

307 The payload for the ChallengeResponse is the following byte sequence.

Byte	0	1-4	5-24
Description	i	AnchorSessionId, encoded as a 32-bit integer in big-endian byte order	$R_{2^{i+1}}$

308

309 **3 Octopus Bindings**

310 **3.1 Constraints**

311 An Octopus control can signal that it requires a proximity measurement to be done by
312 carrying a ProximityRequired constraint in an ESB.

313 The ProximityRequired constraint is in the spatial constrains category, indicated by
314 the local flag SPATIAL_CONSTRAINT as specified in §3.3.2.1 of [8pus].

315 The constraint entry in the ESB has the following format:

316

Name	Type	Description
ProximityRequired	Integer	Expected freshness of the proximity measurement in seconds, or 0 if there is no fixed expected freshness. The expected freshness is the amount of time elapsed, at the time of measurement, since the last valid under-threshold proximity measurement of the peer target.

317

318 **3.2 Control Context**

319 When a running control signals that it requires a proximity measurement by carrying
320 a ProximityRequired constraint described in §3.1, in a NEMO protocol session such
321 as the License Transfer protocol defined in [MCS], the host application SHALL reveal
322 the date of the last valid proximity check between the host and the session's peer
323 NEMO node in the context of that running control. The Host Object path for this value
324 is Sink/Proximity/LastProbe, as specified in §11 of [MCS]. The value is of type
325 Integer, representing the number of minutes elapsed since January 1, 1970 00:00:00
326 (UTC). The most significant bit MUST be 0. It contains the date of the last valid
327 proximity check of the target that is the NEMO peer of the ongoing NEMO protocol
328 session.

329

Appendix: XML Schema and WSDL File Names

330

- **A.1: proximity-check.xsd**
 proximityoverudp.xsd

331

332

- **A.2: proximity-check.wsdl**

333