NFC Device Interoperability Scenarios

Release 0.4

Stephen Tiedemann

March 13, 2014

Contents

This documentation proposes interoperability test scenarios for NFC Devices such as smartphones, tablets, laptops and other comparably capable computing devices. NFC Devices that would typically be characterized as embedded or functionally restricted devices are not subject of the proposal.

All interoperability test scenarios are intended to be run by a *Device In Testmode* against a *Device Under Test* and for any given pairing of *Device A* and *Device B* to be executed once with *Device A* as the *Device In Testmode* and *Device B* as the *Device Under Test*, and once with *Device A* as the *Device Under Test* and *Device B* as the *Device In Testmode*.

CHAPTER 1

User Experience

The user experience test scenarios focus on functionality an end user may expect to work across different NFC devices in a similar, consistent manner. Technology or protocol specific details are thus completely ignored; instead assumptions are made on context sensitive behavior and perceivable end results.

1.1 Share a web page address

An NFC device with the ability to navigate and render web pages shall be able to receive and open a web page address that is transferred using near field communication. It shall also be able to send the address of a web page that is currently visited and displayed on the device screen.

A web page address shall be exchanged as an NDEF message with the NFC Forum Well Known Type "urn:nfc:wkt:U" encoded as defined in the NFC Forum URI RTD specification.

1.1.1 Test data

- http://www.w3.org
- https://www.ssllabs.com
- http://nfc-forum.org
- https://github.com/

1.1.2 Send a web page address

1. With a web browser open the page at http://www.w3.org
2. Touch the other NFC device and verify the same page is opened.
3. With a web browser open the page at https://www.ssllabs.com
4. Touch the other NFC device and verify the same page is opened.
5. With a web browser open the page at http://nfc-forum.org
6. Touch the other NFC device and verify the same page is opened.
7. With a web browser open the page at https://github.com/
8. Touch the other NFC device and verify the same page is opened.

1.1.3 Receive a web page address

1. Use a web browser on the test device to open the page at http://www.w3.org
2. Touch the test device and verify the same page is opened on the device under test.
3. Use a web browser on the test device to open the page at https://www.ssllabs.com
4. Touch the test device and verify the same page is opened on the device under test.
5. Use a web browser on the test device to open the page at http://nfc-forum.org
6. Touch the test device and verify the same page is opened on the device under test.
7. Use a web browser on the test device to open the page at https://github.com/
8. Touch the test device and verify the same page is opened on the device under test.

1.2 Exchange contact information

An NFC device with the ability to handle contact information shall be able to receive and persistently add contact information when touched to another NFC device or tag that presents such information. It shall also be able to send contact information to another NFC device if the user intends to do so. The user's intention to share contact information shall be assumed if a contact is displayed. The user's consent to send contact information should be implied by the fact that (a) contact information is displayed and (b) the device has established near field communication with another device. The user's consent to receive contact information should be implied by the fact that near field communication was established. Prior to both sending and receiving contact information a confirmation dialog may be displayed.

Personal contact information shall be send to a SNEP Default Server as an NDEF message with media type "text/vcard". The NDEF message should contain no more than one record with media type "text/vcard". If the message contains more than one media type "text/vcard" record only the first shall be processed.

The payload of an NDEF record with media type "text/vcard" shall contain a vCard data object that complies with **RFC 6350**.

1.2.1 Example vCard for test

```
BEGIN:VCARD
VERSION:4.0
N:Gump;Forrest;;;
FN:Forrest Gump
ORG:Bubba Gump Shrimp Co.
TITLE:Shrimp Man
PHOTO;MEDIATYPE=image/gif:http://www.example.com/dir_photos/my_photo.gif
TEL;TYPE=work,voice;VALUE=uri:tel:+1-111-555-1212
TEL;TYPE=home,voice;VALUE=uri:tel:+1-404-555-1212
ADR;TYPE=work;LABEL="100 Waters Edge\nBaytown, LA 30314\nUnited States of America"
  :;;100 Waters Edge;Baytown;LA;30314;United States of America
ADR;TYPE=home;LABEL="42 Plantation St.\nBaytown, LA 30314\nUnited States of America"
  :;;42 Plantation St.;Baytown;LA;30314;United States of America
EMAIL:forrestgump@example.com
END:VCARD
```

1.2.2 Read vCard from tag

1. Write the example vCard to an NFC tag.
2. Read from the vCard from the NFC tag. All properties except PHOTO must be imported successfully.

1.2.3 Send vCard to device

1. Open the contact manager application and display a single entry.
2. Perform any required user interface operations to start sharing the contact entry.
3. Touch the other device to send the contact information.
4. Verify that the other device has received the vCard.

1.2.4 Receive vCard from device

1. Close all applications on the device under test.
2. Prepare the other device to send contact information when near field communication is established.
3. Touch both devices to establish near field communication.
4. Verify that a vCard is received and properly saved (optionally after confirmation).

1.2.5 Mutually exchange vCard

1. Prepare both devices to share contact information when near field communication is established.
2. Touch both devices to establish near field communication.
3. Verify that both devices received and properly saved a vCard.

1.3 Share an image or photo

An NFC device with the ability to capture photographs or otherwise hold and display pictures shall be able to share them with another device if both devices feature a common alternative high-speed carrier technology. The default context for initiating a share action shall be the visual rendering of an image on the source device. Sharing is then initiated when near field communication with another device is established and common alternative carrier discovered.

1.3.1 Send an image

1. Choose an image on the *Device Under Test* and make it render on the screen or otherwise selected for sharing when near field commication will be established.
2. Establish near field communication between the *Device Under Test* and *Device In Testmode* and verify that the *Device Under Test* starts sending the image.

1.3.2 Receive an image

1. Choose an image on the *Device In Testmode* and make selected for sharing when near field commication will be established.
2. Establish near field communication between the *Device Under Test* and *Device In Testmode* and verify that the image is received by the *Device Under Test* immediately or after a single confirmation. If a Bluetooth alternative carrier is used for the data transfer the user shall not be requested to confirm device pairing.

1.4 Handling of unknown data

From time to time a device may receive information types it does not understand and can therefore not process. On a fixed function device this will be known by design while on an application hosting device it will typically mean that no appropriate data handler was found. It is desirable that a user is appropriately informed about this situation and not left with the impression that NFC is not working.

1.4.1 Receive unknown data from device

1. Prepare the *Device In Testmode* to send a single record NDEF message with the media type "application/octet-stream" and some arbitrary amount of payload data bytes when near field communication is established.
2. Establish near field communication between the *Device In Testmode* and the *Device Under Test* and verify that the user is notified of unknown content being received.
3. Prepare the *Device In Testmode* to send a single record NDEF message with the NDEF record type "Unknown" (Type Name Format value 0x05) and some arbitrary amount of payload data bytes when near field communication is established.
4. Establish near field communication between the *Device In Testmode* and the *Device Under Test* and verify that the user is notified of unknown content being received.

1.4.2 Read unknown data from tag

1. Prepare a tag to contain a single record NDEF message with the media type "application/octet-stream" and some arbitrary amount of payload data bytes.
2. Use the *Device Under Test* to the tag and verify that the user is notified of unknown content being received.
3. Prepare a tag to contain a single record NDEF message with the NDEF record type "Unknown" (Type Name Format value 0x05) and some arbitrary amount of payload data bytes.
4. Use the *Device Under Test* to read the tag and verify that the user is notified of unknown content being received.

CHAPTER 2

Peer Communication

Near field communication between peer devices is facilitated by the Logical Link Control Protocol (LLCP). LLCP provides for independent connection-less and connection-mode transport channels that allow concurrent transfer of service data units.

Peer communication test scenarios attempt to ensure that components on top of LLCP can communicate with peer components over the near field communication link at the time established and benefit from the service guarantees designed into the protocol.

In order to execute peer communication test scenarios a *Device In Testmode* must provide a mode that allows triggering and execution of selected scenarios in the manner and with the technical steps and details defined.

For a *Device Under Test* to participate in peer communication test scenarios the *Echo Test Application* is required to provide specific test services on top of LLCP.

2.1 Echo Test Application

The *Echo Test Application* is an application running on the *Device Under Test* that uses functionalities of the local LLCP implementation and API.

2.1.1 Connection-less Echo Test Application

The *Connection-less Echo Test Application* provides an echo service that receives service data units on a *logical data link* initiated by the *Device In Testmode* and returns the service data units on a *logical data link* initated by the *Device Under Test*. The behavior is defined by the following two procedures run concurrently.

- Receive service data units at the local service access point `urn:nfc:sn:dta-cl-echo-in` and store them as atomic entities into a first-in-first-out buffer that is able to hold *N* service data unit entities. Start a *delay timer* if the buffer was empty before entering a service data unit. If the buffer capacity is exhausted when a new service data unit is available then wait until it can be successfully stored before accepting a next service data unit from the LLC.
- When the delay timer expires retrieve all service data units from the buffer and send them, in the order retrieved, to the remote service access point `urn:nfc:sn:dta-cl-echo-out`.

2.1.2 Connection-mode Echo Test Application

The *Connection-mode Echo Test Application* provides an echo service that receives service data units on a *data link connection* established by the *Device In Testmode* and returns the service data units on a *data link connection* established by the *Device Under Test*. The behavior is defined by the following two procedures run concurrently.

- Receive service data units at the local service access point `urn:nfc:sn:dta-co-echo-in` and store them as atomic entities into a first-in-first-out buffer that is able to hold *N* service data unit entities. Start a *delay timer* if the buffer was empty before entering a service data unit. If the buffer capacity is exhausted when a new service data unit is available then wait until it can be successfully stored before accepting a next service data unit from the LLC.
- When the delay timer expires retrieve all service data units from the buffer and send them, in the order retrieved, to the remote service access point `urn:nfc:sn:dta-co-echo-out`.

2.2 Link Management

The LLCP Link Management component is responsible for link activation, and deactivation, keep the appearance of a symmetric communication link, as well as frame aggregation and disaggregation.

2.2.1 Link Activation

The link activation is started when the local Medium Access Layer (MAC) notifies the LLC that a peer device capable of executing LLCP has come into communication range. Presently, the only MAC layer defined is the NFC Data Exchange Protocol (DEP). In NFC-DEP, a device capable of executing LLCP is discovered if the magic octet sequence `46666Dh` is received during MAC activation.

The LLCP specification defines that a Parameter Exchange (PAX) PDU is send and received during activation. However, if the MAC layer is NFC-DEP, then the LLC Parameters that would go into the PAX PDU are transmitted as part of the NFC-DEP activation procedure and the PAX PDU is actually forbidden.

LLC Parameters received are *commitments* of capabilities of the sending party, not subjected to negotiation. The most important, and strictly required, parameter is the *Version Number* to indicate the *major* and *minor* protocol version supported. Both sides then independently determine the protocol version to run by using the lower minor version if the major versions are identical. If the major versions differ then the more advanced implementation may decide if it can fall back to the <major>.<minor> version of the peer device.

Interoperability Test Requirement

The interoperability test scenarios require that NFC devices implement at least LLCP Version 1.1.

The *Maximum Information Unit Extension (MIUX)* parameter indicates the maximum number of information octets that the implementation is able to receive within a single LLC PDU. The guaranteed MIU is 128 octets and the MIUX parameter only encodes the number of additional octets that are acceptable. The sum of 128 and MIUX is the *Link MIU*. It is highly recommended that implementations provide a *Link MIU* of more than 128 octets (thus send an MIUX parameter). If an implementation can afford the memory, the largest possible Link MIU of 2176 octets should be used. If a device is short on memory it should at least use a Link MIU of 248 octets, to allow full utilization of an NFC-DEP information packet.

Interoperability Test Requirement

The interoperability test scenarios require that NFC devices have a Link MIU of 248 octets or more.

The *Well-Known Service List (WKS)* parameter informs the peer device of the well-known service access points that are active on the device and for which LLC PDUs will be accepted. It does, however, not imply that other well-known services would not become available after link activation, for example on a platform with on-demand service activation. The main purpose of the WKS parameter is to reduce the amount of service discovery or blind communication attempts.

Interoperability Test Requirement

The interoperability test scenarios require that NFC devices send a WKS parameter during link activation.

The *Link Timeout (LTO)* parameter announces the maximum time the LLC may ever need from receiving to returning an LLC PDU. Said differently, a local LLC can safely assume that if, after sending an LLC PDU, the remote device's Link Timeout expires before an LLC PDU is received, that communication will no longer be possible. If an LTO parameter is not received during link activation, a default value of 100 milliseconds is applied. The larget possible Link Timeout value is 2550 milliseconds which is, as a time to let the user know the end of communication, not an ideal upper bound.

Interoperability Test Requirement

The interoperability test scenarios require that if an LTO parameter then the resulting remote Link Timeout value does not exceed 1000 milliseconds.

The *Option (OPT)* parameter communicates the link service class supported by the sending LLC. The link service class indicates the supported transport types: connection-less, connection-oriented, or both. If the OPT parameter is not received during link activation (or the link service class set to zero), the local LLC may behave as if both connection-less and connection-oriented transport type are supported.

Interoperability Test Requirement

The interoperability test scenarios require that NFC devices support both connection-less and connection-oriented transport type and send an an OPT parameter during link activation.

1. Enable near field communication between the *Device In Testmode* and the *Device Under Test* and receive the LLC Parameters.
2. Verify that the VERSION parameter is received and announces a major version of 1 and a minor version of 1 or higher.
3. Verify that the MIUX parameter is received with an MIU extension value of 120 or more octets (resulting in a remote Link MIU of 248 or more octets).
4. Verify that the WKS parameter is received and announces presence of well-known services at service access point addresses 0, 1, and 4.
5. Verify that if the LTO parameter is received it's value does not exceed 100 (resulting in a remote Link Timeout value of no more than 1000 milliseconds).
6. Verify that the OPT parameter is received and indicates support for both connection-less and connection-oriented transport type communication.

2.2.2 Link Deactivation

The most usual way of Peer-To-Peer link termination between two NFC devices is a communication timeout due to both devices moved out of near field communication range. Nevertheless, a device may for whatever reason wish to terminate the link while communication would still be possible. The LLCP specification call this *intentional link deactivation* and allows a local link management component to send a Disconnect (DISC) PDU to the remote link management component. No further PDUs are then to be exchanged between the two LLCs (buffered transmissions may still be send by a MAC layer but not propagate to the LLC). Note that unlike termination of a *data link connection* the link management component receiving a DISC PDU will not return a Disconnected Mode (DM) PDU.

1. Perform *Link Activation*
2. Send a Disconnect (DISC) PDU with source service access point address 0 to the remote link management component at the destination service access point address 0.
3. Verify that the *Device Under Test* does not send any further LLC PDU.

2.2.3 Link Symmetry

The LLC layer allows service users to run symmetrical communication on top a master-slave communication style MAC layer such as NFC-DEP. To applications or protocols on top of LLCP this means that service data units can be send or received at any point in time, independent of the time when the other device would eventually ask for or answer a transmission.

To achieve symmetrical communication both link management components observe the flow of outbound PDUs and send, if no other PDU is available, a Symmetry (SYMM) PDU as a substitute. The time until a SYMM PDU is sent as a substitute is critical for performance and the appearance of symmetrical communication. Generally it should be as short as possible, but if an implementation expects other PDUs to become available within a short amount of time it may well increase performance if that PDU is sent a few milliseconds later instead of delaying it until a next PDU is received from the remote LLC.

Interoperability Test Requirement

The interoperability test scenarios require that NFC devices send a SYMM PDU no later than 10 milliseconds after a PDU was received and no other PDU became availble for sending.

Sometimes a concern exists that if only SYMM PDUs are exchanged with short delays it does negatively affect power consumption for no useful information exchange (apart from the fact that two devices are still in proximity which could as well regarded useful information). Without debating that concern, a viable way to reduce the exchange of only SYMM PDUs is to observe when a specific number of SYMM PDUs has been the only exchange between the two LLCs, and then increase the time between receiving and returning a SYMM PDU. Any other PDU sent or received would then restore the original conditions.

Interoperability Test Requirement

The interoperability test scenarios require that NFC devices do not increase the time between receiving and sending a SYMM PDU before at least a consecutive sequence of 10 SYMM PDUs has been received and send (5 per direction).

---

1. Perform *Link Activation*

2. Verify for at least 5 seconds that Symmetry (SYMM) or other PDUs are received within the time limits of the remote *Link Timeout*.
3. Verify that the average time between an outbound and the next inbound PDU does not exceed 10 milliseconds until a sequence of 10 consecutive SYMM PDUs are sent and received (5 per direction).
4. Perform *Link Deactivation*

2.2.4 Aggregation

Frame aggregation allows an LLC to send more than one PDU in a single transmission using Aggregated Frame (AGF) PDUs. As LLCP allows multiple conversations at the same time this does almost always significantly increase data throughput and decrease transaction delays for all communications running across the LLCP Link. It is thus highly recommended that NFC Devices implement and use frame aggregation whenever possible.

Interoperability Test Requirement

The interoperability test scenarios require that NFC devices implement and use frame aggregation.

Disaggregating AGF PDUs is mandatory for any LLCP implementation. When disaggregating, embedded PDUs are to be processed in the order they appear within the AGF PDU and treated as if they were received individually in that order.

1. Perform *Link Activation*
2. Send two CONNECT PDUs with different source service access point addresses and the destination service access point address `0` aggregated into a single AGF PDU. Both CONNECT PDUs shall not contain a Service Name (SN) parameter, so they are not treated as a request to resolve and connect by service name.
3. Verify that the *Device Under Test* returns a Disconnected Mode (DM) PDU to each of the service access points that sent a CONNECT PDU aggregated within a single AGF PDU.
4. Perform *Link Deactivation*

2.3 Service Discovery

Service discovery provides an application component a way to learn if a corresponding application component is available on the peer device, and under which service access point address it is expecting to receive service data units. No distinction between connection-less and connection-mode transport is made with regard to service discovery.

The primary method for service discovery is to send a Service Name Lookup (SNL) PDU with one or multiple Service Discovery Request (SDREQ) parameters to the remote service discovery component at service access point 1. The remote service discovery component will then return an SNL PDU with a number of Service Discovery Response (SDRES) parameters each revealing the service access point address of one of the requested service names. For the entire near field communication session these results are cacheable, i.e. service access point addresses will not change after discovery.

An alternate method for service discovery is to send Service Name (SN) parameter within a CONNECT PDU to the remote service discovery component, as a request for both service name lookup and connecting to that service if it exists. This is also called *connect-by-name* and intended to provide a fast

connect method for client connectors that are only interested in a single remote service or assume for good reasons that the service exists.

2.3.1 Service Name Lookup

Verify that a service name is correctly resolved into a service access point address by the remote LLC. The LLCP Link must be activated prior to running this scenario.

1. Establish Peer-To-Peer communication between the *Device Under Test* and the *Device In Test-mode*.
2. Send an SNL PDU with a single SDREQ parameter that encodes the value `urn:nfc:sn:sdp` to the remote service discovery component
3. Verify that the request is answered with an SNL PDU that contains a single SDRES parameter with the SAP value `1` and a TID value that is the same as the value encoded in the previously transmitted SDREQ parameter.
4. Send an SNL PDU with a single SDREQ parameter that encodes the value `urn:nfc:sn:snep` to the remote service discovery component
5. Verify that the request is answered with an SNL PDU that contains a single SDRES parameter with the SAP value `4` and a TID value that is the same as the value encoded in the previously transmitted SDREQ parameter.
6. Send an SNL PDU with a one SDREQ parameter that encodes the value `urn:nfc:sn:sdp` and one SDREQ parameter that encodes the value `urn:nfc:sn:snep` to the remote service discovery component
7. Verify that the request is answered with an SNL PDU that contains two SDRES parameters with TID values matching the previously transmitted TID values, resolving `urn:nfc:sn:sdp` to the service access point value `1` and `urn:nfc:sn:snep` to the service access point value `4`.
8. Terminate Peer-To-Peer communication.

2.3.2 Connect By Name

Verify that a data link connection can be established by specifying a service name. The LLCP Link must be activated prior to running this scenario and the connection-oriented mode echo service must be in the unconnected state.

1. Establish Peer-To-Peer communication between the *Device Under Test* and the *Device In Test-mode*.
2. Send a CONNECT PDU with a Service Name (SN) parameter that encodes the value `urn:nfc:sn:snep` to the service discovery component on the *Device Under Test*.
3. Verify that the *Device Under Test* confirms the connect request with a Connection Complete (CC) PDU.
4. Terminate Peer-To-Peer communication.

2.4 Connection-less Transport

Note: The scenarios in this section require that the *Device Under Test* has the *Connection-mode Echo Test Application* installed and accessible under the service name `urn:nfc:sn:dta-cl-echo-in`.

Connection-less transport provides best effort delivery of service data units between a local and a remote service access point in either direction. An connection-less transport channel is termed a *logical data link* and implied by a pair of local and remote service access point adress values used in a Unnumbered Information (UI) PDU. No setup or termination procedure exisits for a *logical data link*.

Due to the Medium Access (MAC) layer guarantees any outbound LLCP PDU, includinga UI PDU, will arrive at the remote LLC layer. However, a local LLC or remote LLC may drop a UI PDU at any point in time without notification.

2.4.1 Start of Test Sequence

The *start of test* sequence is common to all further test scenarios for connection-less transport mode. Run as a dedicated test scenario, the following steps verify that the *Device Under Test* has the *Connection-less Echo Test Application* installed and accessible, and that the *Connection-less Echo Test Application* attempts to discover the service access point address for returning data packets to the *Device In Testmode*.

1. Perform service discovery to learn the remote service access point address value for the service name `urn:nfc:sn:dta-cl-echo-in`.
2. Send a service data unit with the ASCII string "SOT" to the *Connection-less Echo Test Application* to indicate the start of test.
3. Verify that the *Connection-less Echo Test Application* performs service discovery to learn the service access point address value for the service name `urn:nfc:sn:dta-cl-echo-out` on the *Device In Testmode*.

2.4.2 Guaranteed Information Size

The guaranteed information size for an outbound Unnumbered Information (UI) PDU is 128 octets. Although implementations should support a larger number of information octets, an application designed to work with a variety of peer devices must be able to function even if only the guaranteed information size is available.

1. Perform the *Start of Test Sequence*
2. Send a service data unit with 128 random octets to the *Connection-less Echo Test Application*.
3. Verify that the *Connection-less Echo Test Application* sends the same service data unit to the local service access point bound to `urn:nfc:sn:dta-cl-echo-out`.

2.4.3 Maximum Information Size

The maximum information size of any outbound Unnumbered Information (UI) PDU is determined by the Link MIU value that the remote device transmitted during LLCP Link Activation. The purpose of this scenario is to verify that the *Device Under Test* accepts a UI PDU with a number of information octets equal to the device's Link MIU. Note that in order to run this test the *Device In Testmode* must have a Link MIU that is equal or greater than the Link MIU of the *Device Under Test* because otherwise the *Connection-less Echo Test Application* will not be able to return the service data unit.

1. Perform the *Start of Test Sequence*

2. Send a service data unit with **N** random octets to the *Connection-less Echo Test Application*, with the value of **N** being equal to the Link MIU of the *Device Under Test*.
3. Verify that the *Connection-less Echo Test Application* sends the same service data unit to the local service access point bound to `urn:nfc:sn:dta-cl-echo-out`.

2.4.4 Packet Loss Stimulation

1. Determine the echo buffer capacity **C** and delay **D** of the *Connection-less Echo Test Application* running on the *Device Under Test*.
2. Perform the *Start of Test Sequence*
3. Within **D** seconds send **C + 1** service data units of 128 octets to the *Connection-less Echo Test Application*.
4. Verify that the *Connection-less Echo Test Application* returns the first **C** service data units to the local service access point at `urn:nfc:sn:dta-cl-echo-out`.
5. Send one service data unit of 128 octets to the *Connection-less Echo Test Application*.
6. Verify that the *Connection-less Echo Test Application* returns the service data unit to the local service access point at `urn:nfc:sn:dta-cl-echo-out`.

2.4.5 Link MIU Adherence

The maximum information size of an outbound Unnumbered Information PDU must not exceed the Link MIU of the remote device. This test scenario verifies that the *Device Under Test* observes the Link MIU value of the *Device In Testmode* by sending a service data unit that is one octet larger than the *Device In Testmode* will be able to receive. The test requires that the *Device Under Test* has a Link MIU of at least 129 octets and that the *Device In Testmode* is able to configure its own Link MIU prior to running the test.

1. Determine the Link MIU of the *Device Under Test* as **N** and configure the *Device In Testmode* to use a Link MIU of **N - 1**.
2. Perform the *Start of Test Sequence*
3. Send a service data unit of **N** octets to the *Connection-less Echo Test Application*.
4. Verify that the *Connection-less Echo Test Application* does not return the service data unit to the local service access point at `urn:nfc:sn:dta-cl-echo-out` after the echo buffer delay time.
5. Send a service data unit of **N - 1** octets to the *Connection-less Echo Test Application*.
6. Verify that the *Connection-less Echo Test Application* returns the service data unit to the local service access point at `urn:nfc:sn:dta-cl-echo-out`.

2.5 Connection-mode Transport

Note: The scenarios in this section require that the *Device Under Test* has the *Connection-mode Echo Test Application* installed and accessible under the service name `urn:nfc:sn:dta-co-echo-in`.

Connection-mode transport provides sequenced and guaranteed delivery of service data units between a local and a remote service access point in either direction. An established connection-mode transport channel is termed a *data link connection*. A request for a *data link connection* is made by sending a CONNECT PDU to the remote service access point. A request for a *data link connection* is either confirmed with a Connect Complete (CC) PDU, or rejected with a Disconnected Mode (DM) PDU. Once established, service data units can be moved over a *data link connection* using sequentially numbered Information (I) PDUs.

The flow of I PDUs from sender to receiver is regulated by a sliding window control scheme. A receive window (RW) that is announced during connection setup informs the sender of the number of I PDUs it may send before there comes acknowledgement. Conceptually this is best imagined as a receive buffer to hold up to RW number of I PDUs that fills when I PDUs are received from the remote endpoint and empties when they get dispatched to the upper layer service user. The sender incrementall count of outbound I PDUs is transmitted in the *send sequence number* N(S) field of the I PDU. The receiver populates the *receive sequence number* N(R) field of outbound Information (I) or Receive Ready (RR) or Receive Not Ready (RNR) PDUs with the N(S) of the last I PDU successfully dispatched to the service user, thereby providing acknowledgement and an update of the receive buffer fill status.

To fit into transmissions, the send and receive sequence numbers must be limited using modular arithmetic. LLCP has choosen the modulus 16 so that the send and receive sequence number can be transmitted in a single octet, each consuming 4 bits. Consequently the largest possible receive window size is 15. With the sequence number of the next in-sequence I PDU to be sent denoted by the *send state variable* V(S) and the most recently received N(R) value kept in the *send acknowledgement state variable* V(SA), both initialized with 0 after connection setup, then **V(S) - V(SA) = RW (mod 16)** means that the remote endpoint's receive window is exhausted and an I PDU can not be send. If an I PDU can be send then the *send state variable* V(S) becomes **V(S) + 1 (mod 16)** after transmission.

As *data link connections* are bi-directional acknowledgement, by means of the *receive sequence number*, is ideally transmitted with the I PDUs flowing back and forth. However, if there's no information to send but acknowledgement due, Receive Ready (RR) or Receive Not Ready (RNR) PDUs must be used to convey a new N(R) value to the remote endpoint. For the purpose of sending acknowledgement and flow control there is no difference between sending RR versus RNR. The difference is that sending RNR indicates that the local service user is temporarily unable to consume further data and it's advisable that the remote service user stops generating (or whatever is appropriate). For example, the *Connection-mode Echo Test Application* operates a buffer for data units that regulates how many I PDUs can be dispatched from the receive window. If the receive window would be 3 and the buffer capacity 2, then in total 5 I PDUs may be send by the remote endpoint before all gets stuck. By transmitting the busy signal when the echo buffer becomes full, the remote party can adjust and stop generating.

LLCP requires that the MAC layer is reliable, i.e. data send by a local LLC is guaranteed to arrive at the remote LLC. For *data link connections* this means that a sender does not need to memorize outbound I PDUs for potential retransmission and no mechanism exists that would allow a receiver to request retransmission of an I PDU. What though may happen is that *send sequence numbers* received from the remote endpoint are not monotonically increasing (modulo 16) due to implementation errors in the remote send or the local receive unit. If that happens, the receiver will send a Frame Reject (FRMR) PDU to notify the sender and immediately regard the *data link connection* as terminated.

2.5.1 Connection Establishment

A *data link connection* must be established between a local and a remote service access point before Information PDUs can be exchanged. A connection is requested with a CONNECT PDU to the remote service access point and confirmed with a Connection Complete (CC) PDU to the service access point from which the CONNECT PDU was sent. The address of the remote service access point can be

learned through explicit service discovery sending a Service Name Lookup (SNL) PDU with a service name parameter to the remote service discovery component and learning the address from a response SNL PDU, or by implicit service discovery where the service name is provided as a parameter with a CONNECT PDU to the service discovery component address. The test scenario allows either method to be used by the *Device In Testmode* and *Device Under Test* because it is not an interoperability concern which method an implementation prefers.

The *Connection-mode Echo Test Application* is designed as a service that first accepts a connection request on the local service access point bound to the service name `urn:nfc:sn:dta-co-echo-in`, and then issues a connection request to the remote service access point that is identified by the service name `urn:nfc:sn:dta-co-echo-out`. The steps below verify this behavior and serve as a *start of test* sequence.

1. Request a *data link connection* with the *Connection-mode Echo Test Application*. It is an implementation choice of the *Device In Testmode* whether the remote service name `urn:nfc:sn:dta-co-echo-in` is first resolved into the service access point address and followed by a CONNECT PDU to that address, or a CONNECT PDU is send to the service discovery component with the service name `urn:nfc:sn:dta-co-echo-in` provided as a parameter.
2. Verify that the connect request is confirmed by the *Connection-mode Echo Test Application* with a Connection Complete (CC) PDU. If the CONNECT PDU was sent to the service discovery component verify that the CC PDU source service access point (SSAP) address is different from 1.
3. Wait until the *Device Under Test* requests a *data link connection* with the service access point that is bound to the service name `urn:nfc:sn:dta-co-echo-out` on the *Device In Testmode*. It is an implementation choice of the *Device Under Test* whether it first resolves the service name into a service access point address and then sends a CONNECT PDU to that address, or sends a CONNECT PDU with the service name provided as a parameter to the service discovery component on the *Device In Testmode*.
4. Confirm the connect request with a Connection Complete (CC) PDU.

The *data link connection* requested by the *Device In Testmode* is further referred to as the *outbound data link connection*. The *data link connection* requested by the *Device Under Test* is further referred to as the *inbound data link connection*.

2.5.2 Connection Termination

A *data link connection* may at any time be terminated from either side of the connection. The party wishing to terminate must send a Disconnect (DISC) PDU to the remote service access point of the *data link connection*, which the receiver answers with a Disconnected Mode (DM) PDU. The *data link connection* is then terminated and applications on both sides notified.

1. Perform *Connection Establishment*.
2. Terminate the *outbound data link connection*.
3. Verify that the *Connection-mode Echo Test Application* confirms termination of the *outbound data link connection*.
4. Wait until the *Connection-mode Echo Test Application* terminates the *inbound data link connection*.
5. Confirm termination of the *inbound data link connection*.

2.5.3 Guaranteed Information Size

The guaranteed information size for an outbound Information (I) PDU is 128 octets. Although implementations should support a larger number of information octets, an application designed to work with a variety of peer devices must be able to function even if only the guaranteed information size is available.

1. Perform *Connection Establishment*.
2. Send a service data unit of 128 octets over the *outbound data link connection* to the *Connection-mode Echo Test Application*.
3. Verify that receipt of the service data unit is acknowledged by the *Device Under Test*.
4. Verify that the *Connection-mode Echo Test Application* returns the service data unit over the *inbound data link connection* after the echo buffer delay time.
5. Perform *Connection Termination*.

2.5.4 Maximum Information Size

The maximum information size of an outbound Information (I) PDU is determined by the *data link connection* MIU value that the remote service access point transmitted during connection establishment. The purpose of this scenario is to verify that the *Device Under Test* accepts an Information PDU with a number of information octets equal to the MIU of the remote service access point on the *outbound data link connection*. Note that in order to run this test, the *Device In Testmode* must provide the same MIU on the *inbound data link connection* because otherwise the *Connection-mode Echo Test Application* will not be able to return the service data unit.

1. Perform *Connection Establishment*.
2. Send a service data unit with **N** random octets on the *outbound data link connection* to the *Connection-mode Echo Test Application*, with the value of **N** being equal to the remote MIU of the *outbound data link connection*.
3. Verify that receipt of the service data unit is acknowledged by the *Device Under Test*.
4. Verify that the *Connection-mode Echo Test Application* returns the service data unit over the *inbound data link connection* after the echo buffer delay time.
5. Perform *Connection Termination*.

2.5.5 Sequence Number Cycling

Information (I), Receive Ready (RR), and Receive Not Ready (RNR) PDUs carry send and receive sequence numbers for the purpose of flow control and to acknowledge when the information field of an I PDU has been dispatched to the service user. Sequence numbers start at zero for a new *data link connection* and then increment with **(N + 1) mod 16**. The purpose of this test scenario is to cycle through all possible sequence number values and verify the *Device Under Test* behavior.

1. Perform *Connection Establishment*.
2. Send 17 service data units with 128 octets each on the *outbound data link connection* to the *Connection-mode Echo Test Application*.
3. Verify that all service data units are acknowledged by the *Device Under Test* and successively returned on the *inbound data link connection*.

4. Perform *Connection Termination*.

2.5.6 Receiver Busy Condition

The purpose of this test scenario is to verify that the *Connection-mode Echo Test Application* busy state is signaled by the *Device Under Test* when it is no longer able to retrieve service data units from the LLC. The condition is forced by the *Device In Testmode* when it stops retrieving service data units on the *inbound data link connection* (the *Connection-mode Echo Test Application* will then not be able to return all service data units to the *Device In Testmode*).

1. Perform *Connection Establishment*.
2. Send one service data unit over the *outbound data link connection* to the *Device Under Test*.
3. Verify that the same service data unit is received on the *inbound data link connection* from the *Device Under Test* after the *Connection-mode Echo Test Application* buffer delay time.
4. Stop retrieving service data units from the *inbound data link connection* (the LLC layer will still receive Information PDUs until the Receive Window (RW) is exhausted).
5. Continuously send service data units over the *outbound data link connection* to the *Device Under Test* until the remote Receive Window (RW) is exhausted and verify that the *Connection-mode Echo Test Application* has entered the busy state and caused its LLC to send Receive Not Ready (RNR) PDUs on the *outbound data link connection*.
6. Resume retrieving service data units from the *inbound data link connection* and verify that all service data units previously sent over the *outbound data link connection* are received in order. This may take two or more times the *Connection-mode Echo Test Application* buffer delay time depending on the *outbound data link connection* remote Receive Window (RW) and the *Connection-mode Echo Test Application* echo buffer capacity.
7. Verify that *Connection-mode Echo Test Application* has left the busy state and caused its LLC to send a Receive Ready (RR) PDU on the *outbound data link connection*.
8. Perform *Connection Termination*.

CHAPTER 3

Small Data Exchange

NFC isn't a fast data carrier - it's beauty lies in the explicitness of interaction that a user conciously performs when touching two devices. But 424 kbps physical speed and a transmit time of about 1 second are good enough to transfer about 20 kilobyte of user data with the Simple NDEF Exchange Protocol (SNEP) or other protocols that run on top of the Logical Link Control Protocol (LLCP).

3.1 SNEP Default Server

The SNEP specification defines both a protocol and a *Default SNEP Server* that can be connected to on services access point address 4 or with the service name `urn:nfc:sn:snep`. The *Default SNEP Server* is a mandatory service on NFC Forum compliant devices.

The *Default SNEP Server* provides a simple inbox into which remote clients can drop NDEF messages. What then happens with these messages is not defined but the most natural processing in application hosting devices such as smartphones is to see if there is an application installed that can handle the message type that was received.

Interoperability Test Requirement

The interoperability test scenarios require that a *Default SNEP Server* implementation either dispatches or makes visible the NDEF message received such that the success of receiving can be clearly identified.

Because the *Default SNEP Server* is a mandatory component on every NFC Forum Device, the maximum NDEF message size that is guaranteed to be transmittable is only 1024 octets. However, as application hosting devices have quite substantially more memory and processing power, larger transmission sizes are desirable and achievable.

Interoperability Test Requirement

The interoperability test scenarios require that a *Default SNEP Server* implementation supports NDEF message sizes of at least 20480 byte (20 KByte).

3.1.1 Connect to the Default Server

The *Default SNEP Server* is a well-known service with the service access point address 4 and service name `urn:nfc:sn:snep`. The NFC Forum Device Requirements mandate that an NFC Forum Device always has that service available. A SNEP client can choose to either connect directly to the destination address 4 or use the well-known service name.

The SNEP protocol uses the LLCP connection-oriented transport type facility, thus a client first has to establish a *data link connection* with the server. As part of the connect procedure, the server will indicate its *data link connection* receive window and maximum information unit size. To achieve data throughput that allows 20 KB to be transferred within a time that is convient for the user to hold devices in proximity, the the maximum information unit size should be the largest possible multiple of 248 octets (to match the NFC-DEP capacity for a single information packet) and the receive window should be 2 or more to allow the sender to continue while the earlier LLCP Information (I) PDU is still to be acknowledged.

Interoperability Test Requirement

The interoperability test scenarios require that a *Default SNEP Server* implementation supports a *data link connection* receive window of 2 or more and a maximum information unit size of 1984 octets.

---

1. As a choice of the *Device In Testmode* send either a CONNECT PDU to service access point address 4 or a CONNECT PDU with the service name `urn:nfc:sn:snep` to service access point 1.
2. Verify that the *Device Under Test* replies with a Connection Complete (CC) PDU.
3. Verify that the CC PDU contains a Receive Window (RW) parameter with a value of 2 or more.
4. Verify that the CC PDU contains a Maximum Information Unit Extension (MIUX) parameter with a value that results in an MUI of 1984 octets.
5. *Disconnect from the Default Server*

3.1.2 Disconnect from the Default Server

Once a client has established a connection with the server and potentially transmitted an NDEF message it should disconnect in order to allow the server free its resources.

1. *Connect to the Default Server*
2. Send a Disconnect (DISC) PDU on the *data link connection* with the *Default SNEP Server*.
3. Verify that the *Device Under Test* confirms termination of the *data link connection* with a Disconnected Mode (DM) PDU and reason code 0.

3.1.3 Sequential Connects

Although most current usage scenarios involve just the transfer of a single NDEF message, a *Default SNEP Server* should be prepared to receive another connection request once a client terminated a previous *data link connection*.

1. *Connect to the Default Server*
2. *Disconnect from the Default Server*
3. *Connect to the Default Server*
4. *Disconnect from the Default Server*

3.1.4 Unfragmented Message

A SNEP client uses the PUT command to place an NDEF message into the *Default SNEP Server* inbox. If the NDEF message plus the SNEP protocol header fit into a single I PDU, the server can immediately reply with the *Success* response.

1. *Connect to the Default Server*
2. Send a PUT request with a single record text message that contains the string "NFC Interoperability Test Scenarios" in English language encoding.
3. Verify that the server replies with a *Success* response and that the text is shown on the *Device Under Test*.
4. *Disconnect from the Default Server*

3.1.5 Fragmented Message

If the NDEF message plus the SNEP protocol header does not into a single I PDU, the client must send the message as a sequence of fragments. The server must wait until the last fragment is received before sending the *Success* response. To avoid that the client transmits more data than the server can handle, the first fragment contains, as part of the SNEP request header, the total NDEF message size and the server must reply to the first fragment with a *Continue* or *Reject* response message depending on whether it can receive that amount of data. If the client receives a *Continue* response it sends the remaining fragments without further confirmations.

As stated before, the interoperability test scenarios require that a *Device Under Test* implements the *Default SNEP Server* to accept an NDEF message of up to 20480 octets total size.

1. *Connect to the Default Server*
2. Send the first fragment of a PUT request with an NDEF message of 20480 octet total size.
3. Verify that the server replies with a *Continue* response.
4. Send the remaining octets of the NDEF message.
5. Verify that the server replies with a *Success* response.
6. *Disconnect from the Default Server*

CHAPTER 4

Connection Handover

To transfer larger amounts of data between two NFC devices an alternative carrier connection such as Bluetooth or WiFi can be identified and initiated with NFC.

The NFC Forum Connection Handover specification defines the framework that allows to negotiate an alternative carrier for further data transfer. The device that intends to share data sends a Handover Request message to the service `urn:nfc:sn:handover` of the peer device. The request informs about the alternative carrier(s) that the requester has available. The peer device determines if any of the alternative carriers matches a local carrier and returns a Handover Select message with one or multiple matching alternative carriers. That information allows the requester to determine its final candidate, if any, and use that carrier for the actual data transfer.

Connection Handover can also be performed if one of the devices does not have a P2P cabable NFC implementation but uses an NFC Tag. In this case the requester acquires connection handover select information by reading the tag without prior sending information about its local alternative carrier technologies. The main difference is thus that the selector can not adapt its response to the capabilites of the requester and, if the tag is not internally connected to the host processor, it can also not adapt to dynamic properties of the selectable carriers.

The default data format for an NFC Tag for connection handover is an NDEF message that is a Handover Select Message which allows multiple carriers or configurations to be included. A *Simplified Tag Format* can be used if only one carrier is available and additional features of the *Handover Tag Format* are less important than tag size.

Version 1.3 of the NFC Forum Connection Handover specification introduced a three party model where a third device mediates connection handover between two other devices. This allows to establish an alternative carrier connection between two devices that can not, or not easily, be brought into near field communication distance. The mediation process is basically to acquire alternative carrier information from one device in a first conversation and then negotiate a suitable technology with the other device in a second conversation.

Connection handover using NFC has become an important enabler of easy-to-use spontaneous data sharing between many types of consumer electronics devices. It is thus an important goal that any NFC-enabled device with alternative carrier technologies implements and utilizes connection handover in an interoperable way.

Interoperability Test Requirement

The interoperability test scenarios require that an NFC Device with alternative carrier technologies such as Bluetooth or Wi-Fi implements the NFC Forum Connection Handover specification.

Terminology:

Negotiated Handover When two devices run in NFC P2P mode, negotiated handover is used by the device that intends to transfer content to propose alternative carriers and learn which ones match the capabilities of the other device.

Handover Tag Format A device using the handover tag format presents alternative carrier information in the form of a Handover Select Message. This is especially useful if the device has multiple carriers or carrier configurations available. Additionally, if the tag is internally connected to the host processor it may present carrier power states matching reality.

Simplified Tag Format A device using the simplified tag format presents a single alternative carrier in one NDEF record that marks both start and end of the NDEF message.

Handover Requester The device that intends to send data over an alternative carrier and thus transmits a Handover Request Message to an NFC Peer Device or reads alternative carrier information from an NFC Tag.

Handover Selector The device that replies a Handover Select Message to an NFC Peer Device or supplies alternative carrier information on an NFC Tag.

Handover Mediator

A device tthat acquires alternative carrier information from two other devices to select a suitable alternative and request that a connection be established.

4.1 Common Procedures

The Connection Handover specification defines a framework that allows two implentations to exchange information about alternative carriers and perform a selection process. The protocol follows a client-server model and does not require the server to keep state information between successive client requests.

Connection handover protocol messages are exchanged over an LLCP *data link connection* and thus require connection setup before the handover message exchange. The service access point of a remote handover server application is determined by the NFC Forum registered service name `urn:nfc:sn:handover`.

For performance reasons, the *data link connection* should allow LLCP Information (I) PDUs to utilize at least the full transport capacity of an NFC-DEP MAC layer PDU.

Interoperability Test Requirement

The interoperability test scenarios require that a connection handover implementation provides a data link connection MIU of at least 248 octets.

4.1.1 Establish Connection with Server

The purpose this test scenario is to verify that a handover server implementation accepts a *data link connection* request from a remote client and provides a reasonable *data link connection* MIU. The reason why this is separate from *Terminate Connection with Server* is to be re-usable in other test scenarios.

1. Establish a *data link connection* with the `urn:nfc:sn:handover` service on the *Device Under Test*. It is an implementation choice of the *Device In Testmode* whether the service name is first

resolved into the service access point address and followed by a CONNECT PDU to that address, or a CONNECT PDU is send to the service discovery component with the service name provided as a parameter.

2. Verify that the remote *Maximum Information Unit* MIU of the *data link connection* is 248 or more octets.
3. Perform *Terminate Connection with Server*

4.1.2 Terminate Connection with Server

The purpose this test scenario is to verify that a handover server implementation allows the client to orderly terminate the *data link connection*. The reason why this is separated from *Establish Connection with Server* is to be re-usable in other test scenarios.

1. Perform *Establish Connection with Server*
2. Send a Disconnect (DISC) PDU to terminate the *data link connection* with the connection handover service on the *Device Under Test*.
3. Verify that the *Device Under Test* replies with a Disconnected Mode (DM) PDU.

4.1.3 Sequentially Establish Connection

The purpose of this test scenario is to ensure that a handover server implementation returns to the connectable state after the client has closed the *data link connection*.

1. Perform *Establish Connection with Server*
2. Perform *Terminate Connection with Server*
3. Perform *Establish Connection with Server*
4. Perform *Terminate Connection with Server*

4.1.4 Multiple Messages from Client

The purpose of this test scenario is to ensure that the handover server implementation keeps the *data link connection* and responds to handover messages until the client terminates. This allows a handover requester to ask for only a specific subset of alternative carriers in a first message to impose a strong preference on the handover selector. It also allows a handover mediator to send a handover initiate message after receiving a handover select message over the same *data link connection*.

1. Perform *Establish Connection with Server*
2. Send a handover request message with a single alternative carrier of type `urn:nfc:ext:nfc-forum.org:x-unknown-carrier-type-1`.
3. Verify that the *Device Under Test* returns a Handover Select Message with an empty alternative carrier selection.
4. Send a handover request message with a single alternative carrier of type `urn:nfc:ext:nfc-forum.org:x-unknown-carrier-type-2`.
5. Verify that the *Device Under Test* returns a Handover Select Message with an empty alternative carrier selection.

6. Perform *Terminate Connection with Server*

4.1.5 Accept Connection from Client

The purpose of this test scenario is to ensure that a handover client implementation on the *Device Under Test* establishes the *data link connection* with an MIU of at least 248 octets.

1. Activate the local handover server and wait for a *data link connection* request from the *Device Under Test*.
2. Verify that the remote *Maximum Information Unit* MIU committed with the *data link connection* request is 248 or more octets.
3. Confirm the *data link connection*.
4. Receive and reply connection handover protocol messages until the client terminates the *data link connection*.

4.1.6 Handover Request Collision

The connection handover protocol is run by a local client against with a remote server on behalf of an, explicit or implicit, user intention for data sharing. If both devices intend to share data, the local connection handover clients will both connect to the remote servers to send a handover request message and attempt to perform the role of a Handover Requester. In order to proceed with connection handover, one device must switch role to become the Handover Selector and avoid that both devices eventually try to establish the alternative carrier connection. In case that one device has received the handover request message early enough to stop sending, that device will simply switch role and become the Handover Selector. However, if both devices have sent the handover request message, the conflict is resolved by comparing a local and a remote random number that have been mutually exchanged with the handover request messages.

If the sent and received random number are equal, both devices need to again send a handover request message with a newly generated random number. Otherwise, the random numbers are compared to determine the roles, using both the numerical value represented by all 16 bits of the random number and the value of the least significant bit in spearate comparisions. If the least significant bit of both numbers are equal, the device that sent the numerically greater random number continues as the Handover Selector. If the least significant bit of both numbers are different, the device that sent the numerically lower random number continues as the Handover Selector.

1. Activate the local handover server and wait until *data link connection 1* has been established by the *Device Under Test*.
2. Perform *Establish Connection with Server* to get *data link connection 2*.
3. Wait to receive a Handover Request Message from the *Device Under Test* over *data link connection 1*.
4. Send a Handover Request Message with the same random number as received from the *Device Under Test* (using *data link connection 2*).
5. Verify that the *Device Under Test* sends a further Handover Request Message with a different random number (using *data link connection 1*).

6. If possible, send a Handover Request Message with a random number that assigns the role of Handover Selector to the *Device Under Test* (using *data link connection 2*). Otherwise return to step 4.
7. Verify that the *Device Under Test* closes *data link connection 1* and returns a Handover Select Message over *data link connection 2*.
8. Perform *Terminate Connection with Server* to close *data link connection 2*.

4.2 Bluetooth Simple Pairing

Version 2.1 + EDR (BR/EDR) of the Bluetooth Core Specification introduced Secure Simple Pairing with four association models: *Just Works*, *Numeric Comparision*, *Passkey Entry* and *Out Of Band*. In version 4.0 the security model for Bluetooth Low Energy (LE) was added for similar association models except *Numeric Comparision*.

The general phases to achieve an association between two Bluetooth devices are (1) Device Discovery, (2) Bluetooth Connection, (3) Security Establishment, and (4) Device Authentication. Device Discovery can be achieved in-band or out-of-band using NFC and basically makes the Bluetooth address known to the other device. If the out-of-band communication transmitted security information, that information will then be used during Device Authentication to achieve secure pairing with protection against man-in-the-middle attacks without requiring the user to input a passkey or confirm that numeric values shown are equal.

An illustration of the Secure Simple Pairing Association Models can be found in the Bluetooth Core Specification Version 4.1, Volume 1, Part A, Section 5.2.4.5, Figure 5.2.

The data format for Bluetooth Out Of Band pairing for Bluetooth BR/EDR and Bluetooth LE is defined within the Bluetooth specification. For NFC transmission it becomes the payload of an NDEF record termed *Bluetooth BR/EDR Carrier Configuration Record* and *Bluetooth LE Carrier Configuration Record*. The NFC Forum and Bluetooth SIG *Bluetooth Secure Simple Pairing using NFC* Application Document contains implementation hints and recommendations on optional data elements that should be included in the out-of-band data when used with NFC.

Bluetooth BR/EDR Carrier Configuration Record

The record type is the Internet Media Type "application/vnd.bluetooth.ep.oob" and the payload is the Bluetooth BR/EDR out-of-band configuration data.

The configuration data must include the *BR/EDR Bluetooth Device Address* and may include any number of *Extended Inquiry Response* (EIR) data elements.

For negotiated connection handover both devices have the ability to transmit the *Simple Pairing Hash C* and *Simple Pairing Randomizer R* that allow the receiver to then authenticate the device during Bluetooth link setup.

Interoperability Test Requirement

The interoperability test scenarios require that Bluetooth BR/EDR Carrier Configuration data includes the Simple Pairing Hash and Randomizer if negotiated handover is performed.

The Bluetooth Device Name allows a graphical user interface to better represent the other device in informational messages that may be shown while connecting or especially if the connection could not

be made (for example if the other device kept the Bluetooth radio deactivated and thus a device name could also not be learned through Bluetooth inquiry).

Interoperability Test Requirement

The interoperability test scenarios require that Bluetooth BR/EDR Carrier Configuration data includes the Bluetooth Local Name (Device Name).

Bluetooth LE Carrier Configuration Record

The record type is the Internet Media Type "application/vnd.bluetooth.le.oob" and the payload is the Bluetooth LE out-of-band configuration data.

The configuration data must include the *LE Bluetooth Device Address* and *LE Role* and may include any number of other *Advertising and Scan Response Data* (AD) elements.

For negotiated connection handover both devices have the ability to transmit the *Temporary Key TK* that allows the receiver to authenticate the device during Bluetooth link setup.

Interoperability Test Requirement

The interoperability test scenarios require that Bluetooth LE Carrier Configuration data includes the Temporary Key TK value if negotiated handover is performed.

The Bluetooth Device Name allows a graphical user interface to better represent the other device in informational messages that may be shown while connecting or especially if the connection could not be made (for example if the other device kept the Bluetooth radio deactivated and thus a device name could also not be learned through Bluetooth inquiry).

Interoperability Test Requirement

The interoperability test scenarios require that Bluetooth LE Carrier Configuration data includes the Bluetooth Local Name (Bluetooth Device Name).

List of Test Scenarios

- Negotiated Handover with Bluetooth BR/EDR Device
- Negotiated Handover with Bluetooth LE Device
- Static Handover with Bluetooth BR/EDR Device
- Static Handover with Bluetooth LE Device

4.2.1 Negotiated Handover with Bluetooth BR/EDR Device

1. Perform *Link Activation* and *Establish Connection with Server*
2. Send a Handover Request Message with a Bluetooth BR/EDR Carrier Configuration Record that includes the Bluetooth Device Address, Simple Pairing Hash C, Simple Pairing Randomizer R, Class of Device, Local Name and appropriate Service Class UUIDs.
3. Verify that the *Device Under Test* returns a Handover Select Message with a Bluetooth BR/EDR Carrier Configuration Record that includes the Bluetooth Device Address, Simple Pairing Hash

C, Simple Pairing Randomizer R, Class of Device, Local Name and appropriate Service Class UUIDs.

4. *Terminate Connection with Server*
5. Verify that a Bluetooth connection can be made to the *Device Under Test*.

4.2.2 Negotiated Handover with Bluetooth LE Device

1. Perform *Link Activation* and *Establish Connection with Server*
2. Send a Handover Request Message with a Bluetooth LE Carrier Configuration Record that includes the Bluetooth Device Address, LE Role, Temporary Key TK, and Local Name.
3. Verify that the *Device Under Test* returns a Handover Select Message with a Bluetooth LE Carrier Configuration Record that includes the Bluetooth Device Address, LE Role, Temporary Key TK, and Local Name.
4. *Terminate Connection with Server*
5. Verify that a Bluetooth connection can be made to the *Device Under Test*.

4.2.3 Static Handover with Bluetooth BR/EDR Device

Handover Tag Format

1. Read the NDEF Message from the NFC Tag presented by the *Device Under Test*.
2. Verify that the NDEF Message is a Handover Select Message with a Bluetooth BR/EDR Carrier Configuration Record that includes the Bluetooth Device Address, Class of Device, Local Name and appropriate Service Class UUIDs.
3. Verify that a Bluetooth connection can be made to the *Device Under Test*.

Simplified Tag Format

1. Read the NDEF Message from the NFC Tag presented by the *Device Under Test*.
2. Verify that the NDEF Message is a single Bluetooth BR/EDR Carrier Configuration Record that includes the Bluetooth Device Address, Class of Device, Local Name and appropriate Service Class UUIDs.
3. Verify that a Bluetooth connection can be made to the *Device Under Test*.

4.2.4 Static Handover with Bluetooth LE Device

Handover Tag Format

1. Read the NDEF Message from the NFC Tag presented by the *Device Under Test*.
2. Verify that the NDEF Message is a Handover Select Message with a Bluetooth LE Carrier Configuration Record that includes the Bluetooth Device Address, Generic Access Profile Role, Appearance, Flags and Local Name.

3. Verify that a Bluetooth connection can be made to the *Device Under Test*.

Simplified Tag Format

1. Read the NDEF Message from the NFC Tag presented by the *Device Under Test*.
2. Verify that the NDEF Message is a single Bluetooth LE Carrier Configuration Record that includes the Bluetooth Device Address, Generic Access Profile Role, Appearance, Flags and Local Name.
3. Verify that a Bluetooth connection can be made to the *Device Under Test*.

4.3 Wi-Fi Protected Setup

CHAPTER 5

List of Interoperability Test Requirements

5.1 Peer Communication

- The interoperability test scenarios require that NFC devices implement at least LLCP Version 1.1. (ref)
- The interoperability test scenarios require that NFC devices have a Link MIU of 248 octets or more. (ref)
- The interoperability test scenarios require that NFC devices send a WKS parameter during link activation. (ref)
- The interoperability test scenarios require that if an LTO parameter then the resulting remote Link Timeout value does not exceed 1000 milliseconds. (ref)
- The interoperability test scenarios require that NFC devices support both connection-less and connection-oriented transport type and send an an OPT parameter during link activation. (ref)
- The interoperability test scenarios require that NFC devices send a SYMM PDU no later than 10 milliseconds after a PDU was received and no other PDU became availble for sending. (ref)
- The interoperability test scenarios require that NFC devices do not increase the time between receiving and sending a SYMM PDU before at least a consecutive sequence of 10 SYMM PDUs has been received and send (5 per direction). (ref)
- The interoperability test scenarios require that NFC devices implement and use frame aggregation. (ref)

5.2 Small Data Exchange

- The interoperability test scenarios require that a *Default SNEP Server* implementation either dispatches or makes visible the NDEF message received such that the success of receiving can be clearly identified. (ref)
- The interoperability test scenarios require that a *Default SNEP Server* implementation supports NDEF message sizes of at least 20480 byte (20 KByte). (ref)
- The interoperability test scenarios require that a *Default SNEP Server* implementation supports a *data link connection* receive window of 2 or more and a maximum information unit size of 1984 octets. (ref)

5.3 Connection Handover

- The interoperability test scenarios require that Bluetooth BR/EDR Carrier Configuration data includes the Simple Pairing Hash and Randomizer if negotiated handover is performed. (ref)
- The interoperability test scenarios require that Bluetooth BR/EDR Carrier Configuration data includes the Bluetooth Local Name (Device Name). (ref)
- The interoperability test scenarios require that Bluetooth LE Carrier Configuration data includes the Temporary Key TK value if negotiated handover is performed. (ref)
- The interoperability test scenarios require that Bluetooth LE Carrier Configuration data includes the Bluetooth Local Name (Bluetooth Device Name). (ref)
- The interoperability test scenarios require that a connection handover implementation provides a data link connection MIU of at least 248 octets. (ref)
- The interoperability test scenarios require that an NFC Device with alternative carrier technologies such as Bluetooth or Wi-Fi implements the NFC Forum Connection Handover specification. (ref)

CHAPTER 6

License